Framework for Computer Programming in Preschool and Kindergarten

Tese apresentada por Leonel Caseiro Morgado à Universidade de Trás-os-Montes e Alto Douro para obtenção do grau de Doutor em Informática, sob a orientação da Professora Doutora Maria Gabriel Bulas Cruz, Professora Associada do Departamento de Educação e Psicologia da Universidade de Trás-os-Montes e Alto Douro e do Doutor Kenneth M. Kahn, da empresa Animated Programs, sediada em São Francisco, Califórnia, E.U.A.

Acknowledgments

When I embarked on the journey of developing this thesis and its supporting research, I could hardly imagine the effort and commitment it would require. It wouldn't be easy, that much I was fully aware, but only now I can I look back and realize the many obstacles, difficulties, frustrations, and rocky roads it would imply. Fortunately, I can also now realize the many joys of discovery, of achievement, of child-play, of making blunders and correcting them. And even though learning has always been a pleasure for me, this thesis has also been a personal, intimate reminder of that.

The completion of this thesis is due to many people, who have supported me in countless ways, both directly with the research and with this thesis, and indirectly in providing my with the supportive environment and frame of mind to overcome the hardships and enjoy the satisfactions.

Obrigado, Prof. Bulas Cruz, por me ter expressado a sua confiança na minha capacidade para avançar com este trabalho, antes ainda de eu próprio ter contemplado a possibilidade de o fazer. Deu-me sempre todas as condições para poder continuar a trabalhar sistematicamente nesta obra, mesmo perante outras prementes exigências profissionais e académicas que eu igualmente não podia descurar. E obrigado também por me recordar regularmente da necessidade de concluir este trabalho, mantendo-me em boa rota.

Thank you, Ken, for having been available from the very start to be my thesis advisor, always ready to comment and counsel, always patient to analyze my ideas, no matter how far-fetched, and scrutinize all of them in detail, pointing out issues and requiring my clear explanations and demanding clear thought-out ideas. And all this with a terrific turnaround time! Thank you for all the bug-sorting, the analysis of crash dumps and continual development of ToonTalk features.

Obrigado, Prof.ª Maria Gabriel, por me ter ajudado, um informático, a traçar uma rota através do pensamento e prática na educação de infância, sempre pronta a aconselhar-me e a esclarecer-me, a apoiar-me no relacionamento com educadores de infância, com os jardins-de-infância e com as ideias educativas. Obrigado por me ter facultado o apoio do projecto ICEI e pela confiança, encorajamento e apoio no desenvolvimento da minha actividade lectiva na Licenciatura em Educação de Infância. E por sempre me centrar nas questões essenciais à investigação em curso, ajudando-me a evitar questões menores nas quais facilmente eu me podia ter enleado.

Obrigado, Rosa, por constantemente partilhares comigo a tua paixão, inspiração, saber e experiência em educação de infância, tanto com computadores como sem eles. E por partilhares o caminho comigo durante todos estes anos, dando-me a insubstituível confiança no futuro.

Obrigado Zoé, por me teres dado imensas alegrias durante o meu último ano de trabalho nesta tese, e por me lembrares constantemente, enquanto gatinhavas pela sala ou te atiravas à minha cadeira, que precisava de concluir esta tese para te poder dar toda a devoção que mereces.

Obrigado mãe, obrigado pai, por sempre me terem dado não só o apoio para ir em frente, como também por sempre deixarem incondicionalmente por minha conta a decisão sobre o meu caminho. E por me terem desenvolvido o sentido da responsabilidade, do esforço e do empenho.

Thank you Yishay and Lennart, for your comments on the cookbook. Thank you Mikael, for your comic-strip ideas. Thank you, Playground and WebLabs teams, for letting me share your efforts for the continual development of ToonTalk and programming-based education, the latest ToonTalk versions and bug fixes, and other technical support.

Por fim, obrigado educadoras, pelo acolhimento que sempre tiveram pela minha investigação, obrigado profissionais do projecto ICEI, pelo empenho que cada um colocou; obrigado, meus alunos de educação de infância, pelo contacto rico com as vossas perspectivas do mundo e pelos vossos esforços no desenvolvimento de actividades educativas, em particular à Irina e à Liliana pelo empenho em desenvolver a vossa actividade, thank you Frank Berthold, for your reports on N.

Por fim, um obrigado muitíssimo especial a vós, crianças que participastes com alegria!

Abstract

This thesis aims to be a contribution to the integration of activities with computer programming into preschool education contexts. It is based on experiences and observations of children aged 3, 4, and 5 years olds programming computers, using an animated programming language, ToonTalk, held between 2000 and 2004 in seven different settings, typically in weekly session held over the course of several months in each year, involving approximately 150 children and 10 educators, including early childhood teachers and computer-activities teachers.

Background information on the use of computers and computer programming with preschool children are included, as well as an extensive survey on existing computer programming environments for children. I also provide introductory information about the fields of computing and programming, for the benefit of readers from outside the computing field; for the benefit of readers external to the educational field, and introduction to preschool education is provided.

The research results are provided in various forms: sample activities for integration of specific computer-science concepts into preschool education contexts; an analysis of hurdles faced by preschool children in the process of programming, and suggestions of approaches to help overcome them; an analysis of issues faced by educators in the process of using programming; and finally, a framework for integration of computer-programming activities in preschool education contexts, to assist preschool teachers and educators wishing to employ this resource.

Hopefully the insights from this thesis may prove also helpful for the computer scientist in research on the field of human-computer interaction, particularly regarding the use of computers in the development of cognition.

Resumo

Esta tese pretende ser uma contribuição para o enquadramento das actividades de programação de computadores no contexto da educação pré-escolar. Baseia-se em experiências e observações de crianças de 3, 4 e 5 anos a programar computadores, utilizando uma linguagem de programação animada: ToonTalk, realizadas entre 2000 e 2004, em sete ambientes de investigação diferentes, essencialmente através de sessões semanais ao longo de vários meses de cada ano, que envolveram cerca de 150 crianças e 10 educadores, incluindo educadores de infância e animadores infantis de informática.

São apresentadas as áreas de utilização da informática e da programação de computadores com crianças em idade pré-escolar, bem como um vasto levantamento dos sistemas actualmente existentes para programação de computadores por crianças. Como apoio a leitores exteriores à área da informática, forneço introduções aos campos das computação e da programação; como apoio aos leitores exteriores ao campo da educação, forneço uma introdução à educação pré-escolar.

Os resultados da investigação são apresentados sob diversas formas: exemplos de actividades para integração de conceitos específicos da informática em contextos de educação pré-escolar; uma análise das dificuldades enfrentadas pelas crianças desta faixa etária no processo de programação, e sugestões quanto a abordagens que apoiem a ultrapassagem dessas dificuldades; uma análise das dificuldades enfrentadas pelos educadores no processo de uso da programação; e por fim, um enquadramento da integração de actividades de programação de computadores em contextos de educação pré-escolar, que visa servir de apoio aos educadores de infância e outros profissionais de educação que pretendam utilizar este recurso.

Espero igualmente que as percepções e discernimentos incluídos nesta tese possam revelar-se úteis aos cientistas informáticos, na investigação no campo da interacção humano-computador, muito em particular relativamente à utilização de computadores para desenvolvimento da cognição.

Short summary

Having picked up this thesis, I assume the reader has already developed an interest, or at least some curiosity, for the fields of research and practice of computer programming with young children. My own engagement with this field came from my personal interests and motivations, and these are inseparable from my personal history and background on the use of computers. **Chapter 1** deals with my personal view of the three areas that meet in this thesis: teaching & learning, computers & programming, and computer programming in education.

Before delving into the particulars of computer programming proper, I wish to share with the reader some or my overall understanding of the field, its main issues, and its history and evolution, and this is the purpose of **Chapter 2**. Those acquainted with the history and evolution of computers and computer programming may wish to skip section 2.2, *Brief history of computer programming*; for other readers, this section isn't absolutely essential, but I hope it dispels potentially confusing ideas of computers as calculators, by providing an enjoyable narrative pathway of how machines developed for calculations transformed into machines for working with ideas. For all readers, section 2.1, *Body & Mind: hardware and software*, aims to provide the perspective I employed when discussing programming, and the background relationship I have considered when speaking of software and hardware. In section 2.3, *Computer programming in our society*, I present instances and uses of programming by people other than those whose job descriptions mentions "programmer", and even some cases which aren't often regarded as programming at all. To conclude this chapter, section 2.4, *Computer programming in education*, presents the major approaches in the use of computers in the field of education, so that all readers can be acquainted with the position of programming itself in this context.

The central issues of computer programming are the focus of **Chapter 3**. It presents to all readers the two areas of programming employed in this thesis, which are not usually discussed in the educational field. Section 3.2.1 presents the notions behind programming several actions to take place concurrently, and section 3.2.2 may prove particularly interesting to computer scientists, for it presents the field of constraint programming, which is not one of the most common computer science topics. Section 3.3, *Programming languages*, should not be disregarded by education professionals, for it addresses several central issues in the use of programming with children. Particularly important in this section is the discussion on the relationship between abstract and concrete concepts, in section 3.3.3. It also contains a lengthy survey of computer-programming systems devised for children to use (section 3.3.4), and an extensive description of the language ToonTalk, employed in the field work of this thesis, which is not found elsewhere in any single document. This chapter also presents one of the first results of my research, in section 3.4, in the guise of a "cookbook" of activities for preschools, each focused on a specific topic of computer programming. Thus, these technical concepts of programming are not simply presented, but exemplified in their applicability in preschool settings.

Chapter 4 includes a first section, introducing the field of preschool education and its background; I hope that educators from other educational backgrounds benefit from this selection of perspectives on the specific issues of theory and practice in preschool education, as should computing professionals less acquainted with preschool education theory and practice. The second section, *Computers in preschool*, provides the thought and background on the use of computers in preschool, and computer programming in particular. Of particular importance as a theoretical background is the presentation of the educational thought of Seymour Papert, in section 4.2.2.

The final chapters deal with the actual research description and the core results (apart from section 3.4). **Chapter 5** details the progress and settings of the research and data collecting, including references to the appropriate annexes, where accounts, reports, and other raw data is presented. In **Chapter 6**, I provide an analysis of the problems and difficulties faced by both children and their teachers in the use of programming, as well as the results of different approaches used against those problems and difficulties.

Lastly, **Chapter 7** aims to support the immersive integration of computer-programming in the typical contexts of preschool education. Initially, in sections 7.1 and 7.2, an overview on strategies for including computer activities in general is provided, situating programming within this strategy. Finally, section 7.3 presents several progressive approaches to the use of computer programming with children aged 3, 4 and 5, in preschool education settings, with examples based on ToonTalk but also explained in terms of other computer environments for children.

I hope to have provided not just an useful and informative read, but also a pleasant one.

Table of Contents

Table of Acronyms

Table of Figures

Table of tables

1. Thesis overview

1.1. Of teaching and learning

The nurturing and rearing of our children is one of mankind's greatest tasks. It is so, biologically, for the obvious reason that children carry the genes of our species (egotistically, our own); but they also carry our history – our heritage. They are our link with immortality, just like we are a link between our children and our ancestors.

While this can be said of any living organism (or in fact, with some literary or philosophic license, of any entity[1]) our role of sentient species imbeds us with a particular relationship amongst ourselves: we are teachers and learners (Freire, 1997, pp. 25-26). Each generation thus has a legacy to hand over – teach – to the next. However, new knowledge is also generated – new learning takes place – and the body of human knowledge increases.

The traditional way to interpret this flow of knowledge is to see teaching as transmission of knowledge; and learning, as reception of that knowledge. Current learning theories dispute this interpretation, most notably by noting that human beings build ("construct") their own mental representations of knowledge[2], and also by constantly acquiring and memorizing new information[3].

This traditional interpretation eventually led to the creation of educational systems – schools (see section 4.1.1).

While in present-day schools we can find a large variety of teaching styles and educational philosophies (see section 4.1), the overall idea of the school system is still to **transmit** (or at least convey) specific knowledge and skills to students.

However, a notably different educational system exists, "within the system", so to speak: I am referring to Early Childhood Education (preschool and kindergarten). The educational system for children aged 3, 4 and 5 (an historical evolution of preschool and kindergarten education is presented in section 4.1.1).

In preschool and kindergarten, the major aim is not to convey specific information, but rather to support the children's personal and social development (Ministério da Educação, 1997): a constant struggle to help each child achieve its highest potential. This powerful educational aim is even seen as potentially benefiting older learners (Resnick, 1998).

Having decided to embark on a research that would involve educational settings, these particular features of preschool education were much closer to my personal feelings, intuitions and interests. I find particularly engaging the following:

- **the openness of goals**: in preschool, children and teachers are not faced with a set of concepts that must absolutely be approached, often with somewhat pre-determined pace and style;
- **the openness of children**: younger children are generally seen as being more open to new concepts, tasks and learning; while the somewhat informal learning setting of preschool (compared to elementary school) may very well be part of this general impression, the children's social and mental development are likely involved; whatever the actual background of such intuitive feeling on my part, it meant for me that I felt a greater, deeper connection between child and educator when working with children in this age group (3-5 year olds);

[1] *Cf.* "*As the ancient mythmakers knew, we are the children equally of the sky and the Earth*" (Sagan, 1983, p. 264).
[2] This was first theorized by Jean Piaget (Spodek & Saracho, 1994, pp. 73-78; Fosnot, 1996) but is also part of the educational theories of Paulo Freire (1997) and Leo Vygotsky (Oliveira, 1997; Papalia *et al.*, 1999, p. 339). More information is available in section 4.1.
[3] The information-processing theory was initiated by George Miller (1956) and developed by several researchers, such as Katherine Nelson (an excellent description of the information-processing approach to the development of cognition can be found in Papalia *et al.*, 1999, pp. 328-333). In this thesis, more information is presented in section 4.1.

- **the feeling of freedom, of no ceiling**: this is perhaps a consequence of the two previous features; but it has nonetheless a distinct, broader impact on my motivation to do research work in these educational settings: untethered by mandatory curricula and formal, content-based evaluations, my research work could proceed with fewer constraints in terms of time and content; I would be less tempted to "push" children, or to back off from an experimental pathway: my research would benefit, I believed, from a most precious commodity: time to mature.

1.2. Of computers and programming

I would be somewhat betraying this background description of my motivations for research if I were to present computer programming as a purely technical skill I had eventually acquired, omitting its profound significance in my life. I believe that much of my mental development, thought patterns and styles – possibly even my worldview – may have been affected by it. I see these impacts as having been immensely positive, and this provides me with perhaps my most valuable asset for conducting research in a field, fighting frustration and hardships: a profound conviction on its significance – finding myself wrong satisfies a personal need for understanding of knowledge as great as finding renewed confidence and conviction.

This lengthy recollection of my personal early experiences with computer programming will allow me to present a few ideas that frame my research.

My first contact with anything I can relate to computer programming came from a machine seen even in popular culture as a bane to understanding: the videocassette recorder (VCR). My father owned an electric appliances store in a small town, and sold VCRs. Part of the service provided to the customers included delivery to their homes, installation, and an explanation of its operation. Since this was an expensive piece of equipment, the more features that were presented the better, so programming a recorder to start working at a given time was always part of these "explanations". At the time, many instruction manuals came without a Portuguese version, and this simple fact was the initial cause of my connection with programming.

I had recently begun learning English in school, so my father asked me to try and figure out how the VCRs worked, so that I could go with him to customers' homes and explain it. For me, a VCR was just another appliance, so I basically approached it naturally and learned the programming style from the user manuals. Sometimes I would be presented with an old VCR without any manual, and had to figure out its operation by trial, error, and inference from what I already knew. I can't recall my actual age at the time, but since I started learning English when I was 9 years old, this was probably when I was 10 or 11.

My recollection of contact with videogames comes from about the same time, most notably the old PONG-style home gaming machines (Wolf, 2003; Winter [1], 2004) that were connected to television sets (Winter [2], 2004). And soon after, I came aware of the existence of computer programming, when my brother and I received a game-playing system called Philips Videopac G7000 – in the USA, it was marketed as Odyssey² (Cassidy, 2004). It was only intended for gaming, but browsing the list of game cartridges there was one for programming (anon., 2004). It announced the possibility of creating and running our own programs. I ended up never getting my hands on that cartridge, but programming was becoming an enticing mystery.

The next crucial step was a brief encounter (one-evening long) with a Sinclair ZX81 computer (Barber, 2004) some customer had ordered. He was going to pick it up the following day, and my father asked if I could learn whatever I could in a few hours, in case the customer had any questions. This was too great a challenge, and I didn't make much out of that computer time. But it did remain in my memory as a mysterious, magical moment with a completely new thing: the prompt, blinking, displaying strange commands as I clicked the keys (in the ZX81 editing environment, each key would display a full command). Within a year or so, in a similar situation, my father did

Figure 1 – Sinclair Research ZX81 computer
From: http://www.microhobby.com/010203/ord/zx81-002.jpg

more than put me in brief contact with a customer's computer: when someone ordered a Zilog Z80-based Sinclair ZX Spectrum (VV.AA., 2004) with 16KB RAM, he bought a ZX Spectrum with 48KB RAM and a few games and offered it to me.

Again, I started playing games – I had to issue a programming language command to load them from tape: `Load ""`. But then my family went to a book fair, where we bought two computer-programming books: the Portuguese versions of *Programming your ZX Spectrum* (Hartnell, 1982) and *Creating Adventure Programs on Your Computer* (Nelson, 1983). Being able to make my own games was source of motivation for learning how to program. I was either 12 or 13 years old. And I can never forget the excitement of having made my first program – at the time, two simple commands to print a few blocks together, inked green, to display what I called "a green house".

Figure 2 – Sinclair Research ZX Spectrum computer
From: http://appel.rz.hu-berlin.de/Zope/cm/cmKat/Sinclair.JPG (enhanced)

From then on, through that ZX Spectrum computer, I ended up exploring several areas of programming using the BASIC programming language. But the desire to understand how "real" games where made even led me to a brief reading on the concept of assembly-language programming, a few years later. I even resorted to an advanced computer-science concept unknowingly – using overlays to run programs bigger than the available RAM.

In retrospect, I believe that I was lucky enough to benefit from a situation where three current educational ideas were very much present. These are presented here in a simplistic view, but will be explained in more detail and background later on.

The **first idea** is that the most significant moments of learning computer programming didn't involve structured "teaching" or structured "learning": I was going on my own, driven mostly by coming up with ideas about what would be "fun" to try, but also about how to assist me (my biggest project in those years was creating a system for simplifying the tedious process of character creation in role-playing games). Each achievement was felt as a tremendous success, each failure was either a challenge to overcome or simply forgotten, since there was so much to explore[4].

The **second idea** is that I didn't manage to program a thing until I got hold of both a book structuring the knowledge about programming AND a book making clear that pretty fast I could be doing something worth my troubles: programming adventure games[5].

The **third idea** is that my efforts weren't disregarded as useless or serving as a mere method of keeping me occupied or "studying": from the very first moment, I was doing something useful, my tasks were valued: I was helping my father do something he had trouble doing himself. Later on, even if just to amuse me, each program had a purpose beyond mere "training"[6].

These ideas permeate my research as background beliefs: learning needs an open environment, where children can follow their ideas and interests, not just someone else's; children need some structure and support that helps them realize what they can achieve and how; children take their actions seriously and adults should respect that (Santos, 1999), to the point of involving them in activities that are useful and meaningful for all, not just for "training" or "practicing".

[4] "*The best learning takes place when the learner takes charge*" (Papert, 1980, p. 214).
[5] "(…) *an intellectual culture in which individual projects are encouraged and contact with powerful ideas is facilitated*" (Papert, 1999, p. XV).
[6] "*We believe in making learning worthwhile for use now and not only for banking to use later*" (Papert, 1999, p. XVI).

But looking at these settings in my personal account, I can also point out that they couldn't have occurred much earlier in my life. And the reason for that isn't a matter of intelligence or cognition, but a very real barrier: I needed to know at least a bit of English to start, and that learning only started in October of 1980, when I was 9 (almost 10). It is reasonable to say that if the instructions of VCRs and the BASIC instruction set were available in Portuguese, I probably would have been able to learn how to program earlier: numerous accounts and research of programming with young children demonstrate that young children can program[7].

Since that time, numerous barriers to computer use – and programming – have fallen: programming can be done in a child's mother tongue; computer use has been greatly simplified due to new software and hardware interfaces (GUI, mice, etc.); the production of physical results has been greatly simplified and enhanced (better and cheaper printing or robotics). A decisive moment for initiating this research was when I realized that two such barriers had been broken.

The first such barrier is that programming can now be done entirely without writing or reading of words. Several programming systems, described in chapter 3.3, allow the programmer (and children) to issue commands and create programs by selection and organization of visual or physical elements. The tearing down of this barrier meant that children could try to program before they were able to comfortably read or write commands.

The second barrier is that programming can now be performed in connection with concrete concepts and elements, most notably using techniques of programming by demonstration, which I describe in section 2.3.2. The tearing down of this barrier means that children could try to program early in their mental development.

The realization that this latter barrier had been at least partly overcome was a decisive moment for me, leading to the initiation of this research: I grew curious to know whether very young children could or could not start to program computers.

[7] Papert's books and papers contains numerous accounts, mostly of mid-elementary school age, between 9 and 12 years old: *e.g.*, fourth-graders (1993, pp. 68-69; *id.*, ch. 6, "An Anthology of Learning Stories", pp. 106-136); a fifth-grader (1980, p. 100); a sixth-grader (*id.* p. 118). Reports for first graders are also available (*e.g.*, Degelman *et al.*, 1986), for all years of elementary school (Papert, 1993, pp. 75-76), and even a few for children as young as four (Perlman, 1974). More recently, the Playground project involved children aged 6 to 8 in programming activities in different programming environments (Playground Project, 2001).

1.3. Of computer programming in education

I have presented my motivations and background for conducting research in the field of computer programming in preschool education. However, some extra background should, I believe, be presented now, before entering the more information-rich chapters in this thesis.

When I started looking for previous research in this field, I became aware that previous researchers, most remarkably Alan Kay (Kay, n.d.), Seymour Papert (Papert, 1980, 1993 & 1999; Papert & Harel, 1991) and Andrea diSessa (diSessa, 2000), have built a corpus of knowledge providing a broad and powerful idea: computer programming can be used as a new literacy (diSessa, 2000: chapter 1, pp.1-28, "Computational Media and New Literacies – The Very Idea"), a new tool allowing people to think and reason in novel ways[8]. As these researchers have remarked, and as some non-science writers have imagined, this opens tremendous potential for change in our society (*e.g.* Kuttner & Moore, 1943; Bork, 2000). In educational terms, it is not the programming by itself that produces the change, but its use as part of a different way of learning – or even a different way of facing life (Clements and Meredith, 1992; Papert, 1998, 1999).

And yet, while research work under this framework has been extensively conducted with older children, accounts of programming with pre-literate children are scarce, as I'll document in the coming chapters. Also scarce are accounts of programming in the educational environment of preschools and kindergartens (Clements, Nastasi & Swaminathan, 1993; Iglesias, 1999).

My goal for this research was thus to perform groundwork exploration of the possibilities and hurdles involved in conducting computer-programming activities with 3, 4, and 5-year old children, in the environment of preschools and kindergartens. I hope to have contributed to further build up the knowledge in this field, so that children can benefit from this novel thinking tool early in their lives.

[8] "*seeing ideas from computer science not only as instruments of explanation of how learning and thinking in fact do work, but also as instruments of change that might alter, and possibly improve, the way people learn and think*" (Papert, 1980, pp. 208-209)

2. Computer Programming: conceptual foundations

2.1. Body & Mind: hardware and software

For the uninitiated, computer programming might seem an esoteric topic. It usually brings echoes of technical wizardry, of complex mathematics and reasoning, and – an important perspective for people involved in humanistic areas – of being a cold, machine-like activity, with little or no human warmth.

Programming isn't alone in this unwelcoming public image. Science, for instance, suffers from the same social bias, and this image of low-popularity may indeed impact both children's and adults' perception of their relevance.

> *« In both theory and practice, science in this century has been perceived as a noble endeavor. Yet science has always been a bit outside society's inner circle. The cultural center of Western civilization has pivoted around the arts, with science orbiting at a safe distance. When we say "culture," we think of books, music, or painting.* (…) *Popular opinion has held that our era will be remembered for great art, such as jazz. Therefore, musicians are esteemed. Novelists are hip. Film directors are cool. Scientists, on the other hand, are ...nerds. »*
>
> *(Kelly, 1998)*[9]

And yet, many people can probably recall events where friends with technical computer interests meet, often by chance, and frequently spark up "strange", technically-oriented conversations. By recalling how intense such conversations are – sometimes lasting for hours – people may also recall having themselves said or heard statements such as "they only talk about computers all the time". Such passionate conversations don't match the idea of computer use and computer programming as cold, machine-oriented activities. Were they so, how could they engage their practitioners in such committed conversation? The alternative is to consider a different possibility: that programming also appeals to one's emotional self, not just the analytical self. This will be rendered clearer after I discuss the presence and use of programming in the current society, in section 2.3.

Therefore, before discussing computer programming in more detailed fashion, its occurrence, features and varieties, I propose an introductory look at the concept itself. What *is* computer programming? To answer, I need to present and explain the computer itself, viewed as the combination of "hardware" and "software".

When thinking about general-purpose computers, it is very common to find people that associate the ideas of computers and tools, to the point that saying "the computer is a tool" is commonplace (Kay, 1993a). And yet, while one can definitely think of the computer as a tool, it is not just any ordinary tool. It can be used as leverage to achieve a goal, such as composing a text or performing a calculation – and in this sense it is similar to a hammer or a Swiss-army knife. But it can be used on its own terms, as when one is playing a game, customizing the user environment or trying out a computer-programming idea – and in this sense it is similar to a rock, a cloud, or an idea: something with which one plays for delight, contemplation or inspiration. Possibly a better word for it would be "instrument" – after all, one plays musical instruments, not musical tools. In the remainder of this section, I still use the common idea "computer as a tool", but the word "tool",

[9] On the lighter side, being "nerd" in current days has acquired some glamour. Kelly says, in this same essay: "*Call it nerd culture. For the last two decades, as technology supersaturated our cultural environment, the gravity of technology simply became too hard to ignore. For this current generation of Nintendo kids, their technology is their culture. When they reached the point (as every generation of youth does) of creating the current fads, the next funny thing happened: Nerds became cool. Nerds now grace the cover of Time and Newsweek*" (Kelly, 1998). A more recent account of this new social status of the "nerdish" tastes is given by The Guardian's newspaper headline after two highly successful movies in the "Lord of the Rings" trilogy: "*We are all nerds now*" (Brooks, 2003, citing interviewee).

here, should be interpreted in its most broad sense. After all, a musical instrument is a musician's tool.

I'll consider a very simple tool: a wooden stick. Usage of such a tool is documented, for instance, among apes: chimpanzees can insert a grass stem or a wooden stick in a termite's mound, wait a while and carefully remove it, in this way attaining a meal of termites (Estes, 1991). What makes a tool out of the wooden stick? Chimpanzees don't carve sticks (although they must select them carefully and remove any leaves), so the sticks are just as nature-made as anything can be. It's the way in which they are used that turns the sticks into tools. This usage may be referred to as "knowledge", "method" or "following operating instructions". But it isn't a physical property of the stick: it is intangible.

The computer is also based on physical, touchable materials. The most obvious are those composing the external elements: metal and plastic cases, buttons, and lights. But the internal materials are more important: the electronic circuits made of metal and semiconductors. All of these are the tangible parts of the computer. They are usually called "**hardware**".

However, by itself, hardware doesn't allow us to accomplish much. Nothing appears on the screen, no response occurs to our actions. Therefore, the only ways in which the computer could be used as a tool, were it only composed of hardware, would be as a stand to reach higher, as a counter-weight, and similar improvisational ends to which the physical computer can be put to use.

The intangible portion is what renders the computer a tool, in the sense it is more commonly thought of. Consider the wooden stick again. It is easy to consider equivalent "methods" for a computer, in terms of "usage knowledge". It must be connected to electric power. There is a power button to turn it on. There are keys on a keyboard that can be tapped. And one could assemble a long list of such pieces of knowledge, but the most crucial parts would be missing: what is supposed to happen when the human finger hits a key? When the mouse is moved? In fact, what is supposed to happen when the computer is turned on? Something must render "real" non-physical objects such as the mouse cursor, the menus, and the icons. Before a common word processor, spreadsheet or even game can be used, its existence must somehow emerge from the physical parts.

By comparing the computer and the wooden stick as general tools, a central differing aspect is that the computer has an intermediate layer, between the physical tool and the human knowledge of its operation. The computer operates by executing instructions. When it is turned on, its circuits feed the computer's "brain" (the processor) with the instructions necessary to display some information on the screen. Then the processor is instructed to activate storage devices, such as disk drives, CD drives or flash memory drives, in search of further instructions. The execution of those instructions presents to the human user a set of objects which can be used to "manipulate" the computer further: the mouse cursor, the insertion prompt, menus, icons, and all the interface elements most people are acquainted with. The same happens if the computer is interacting with another machine, rather than a human: some interface device needs to be commanded and employed.

These instructions and the constructs that they produce are called "**software**". The concept of software thus includes all the applications that humans employ to use the computer: word processors, spreadsheets, games, presentation builders, databases, Web browsers, etc.

This viewpoint on the computer, as the combination of hardware and software, has been seen as a metaphor for being as a combination of body and mind. We can see the hardware as the computer "body" and software as the computer "mind". This metaphor has been used extensively, but perhaps a nice example of its appropriateness is its use in debate over the philosophical problem known has "*the mind-body problem*" (Fodor, 1981; Dunhio, 1991; Harrison, 1992).

The computer-tool, therefore, is not a traditional tool, such as the wooden stick: its "mind" (software) allows it to assume completely different functional forms, rendering it useful for distinct tasks. Using a word processor, it is a writing tool; with a game, it's an entertainment tool; using a

screen-painting program, it becomes a creativity tool or an illustration tool; and so on. Its greatest power, seen as a tool, is precisely this ability: it can turn into a myriad of specific-purpose tools. It is a tool without a purpose: a generic tool, which must metamorphose into a specific tool, in order to be used. For this reason, it can more aptly be named a **metatool**.

> *«* (…) [The] *protean nature of the computer is such that it can act like a machine or like a language to be shaped and exploited. It is a medium that can dynamically simulate the details of any other medium, including media that cannot exist physically. It is not a tool, although it can act like many tools. It is the first metamedium, and as such it has degrees of freedom for representation and expression never before encountered and as yet barely investigated. Even more important, it is fun, and therefore intrinsically worth doing. »*
>
> *(Kay 1984)*

In a broader sense, **programming** is the process by which the computer-tool is transformed into another tool, which in turn can be used to achieve a specific goal[10,11]. Its understanding, be it conscious or unconscious, is therefore crucial to being able to employ this machine effectively.

[10] An interesting corollary of this idea is that a computer can be programmed to simulate the operation of another computer, *i.e.*, turning a general computer-tool into another general computer-tool. This is what occurs, for instance, when implementing a programming language.

[11] This definition is similar to Dijkstra's "*We can view the program as what turns the general-purpose computer into a special-purpose symbol manipulator, and it does so without the need to change a single wire*" (Dijkstra, 1989, p. 1401).

2.2. Brief history of computer programming

2.2.1. Historical overview

Telling the history of computer programming is helpful in understanding the actual actions and methods it employs today – and also in envisaging where it all may lead. Since this history covers many years and elements, this overview aims to present its overall framing in this thesis, as a support to the reader. The following sections expand these concepts.

This account of computer programming history aims to present the evolution of two ideas: what is a **computer** and what is a **programming language**, the technical concept around which this thesis revolves. This evolution, in a somewhat similar way to biological evolution, did not follow a straight path, but was rather the result of many separate contributions by different individuals. Some of these fed the inspiration of further developers, leading to newer evolutionary forms, and eventually to modern-day computers; some other contributions simply represented evolutionary paths that extinguished themselves, and which, being discovered later, have their relevance in providing a picture of how different history might have turned.

Computers evolved from machines intended to support or replace the human process of doing calculations. Initially, each tool or machine would seek to accomplish this by focusing on one specific process of calculation (some simple, some complex): add numbers; compare values; calculate the value of a polynomial; etc. As these machines grow more complex and powerful, the aim of their developers starts to be the encompassing of more than one process of calculation, to try and solve as many different calculation problems as possible. The history of this evolution is told in sections 2.2.2 and 2.2.3.

A crucial point is reached when the computer comes into existence as a machine capable of following general procedures for calculation, rather than general calculations using a limited set of procedures. This achievement was done both conceptually (in mathematical notation) and physically (by building actual machines). This is accounted in chapters 2.2.4 and 2.2.5. Machines with this ability are seen as the first computers, in the sense most currently used today.

In this thesis, a further crucial evolutionary point is considered. The computer machines mentioned in the previous paragraph often achieved the goal of following general procedures by reorganization of their physical elements. This means that they had provisions for being turned into different machines by their human operators, for each procedure. (For instance, the ENIAC cabling was re-wired in order for a new program to run.) In the evolutionary line that originated modern computers, this crucial point was achieved when computers became able to store internally the procedures they were to follow, interpreting them afterwards without need for human intervention. This is explained in chapter 2.2.4.

Programming languages, like human languages, are sets of codes that can be combined using specific grammars, in order to convey to a computer the procedures it is intended to follow. Initially, computers were not so much "programmed" (in the current sense) as "set-up" or configured: programming was basically the reorganization of the computing machine's components. As described for the evolution of computers, these machines eventually became able to interpret sets of procedures and follow them without need for human intervention.

Free from being framed by the actual physical organization of machines, the methods for setting the machines with procedures evolved, aiming to provide humans with easier and broader ways to express their aims and intentions. In this way, programming languages were created, from the very start as a means of expression and communication. The history of this evolution is told in sections 2.2.5 and 2.2.6.

2.2.2. The first programmable computer device

Computer programming, as defined at the end of section 2.1, absolutely relies on the computers' ability to operate in diverse manners, *i.e.*, to metamorphose into different tools. The emergence of computers with this ability thus marks the birth of programming. In this sense, it may seem odd to refer to the existence of non-programmable computers. This oddity is however demystified by recalling that computers originated from the desire to achieve a simpler task: automate tedious calculation or, more precisely, automate the execution of an algorithm (Brookshear, 2003, pp. 2-9). The very verb "to compute" originated from the Latin *computare*, which simply means "to calculate" (Cawley, 2004). In fact, originally the term "computer" was not applied to machines but rather to humans whose task was to perform tedious calculations (Ceruzzi, 1983).

The origins of instruments to support calculations are usually traced as far back as the use of primitive abaci, or counting boards, in the ancient world. The abacus, in various forms and under various names, has been used worldwide, and is still used today (Brookshear, 2003, p. 5; Sadiku & Obiozor, 1997; Williams, 1990, pp. 7-15).

However, physical aids to calculation did not start with the abacus: they are most likely to have been in existence as long as mankind has been able to count. One can point out, for instance, using one's fingers, other parts of the body, or grouping small pebbles (Williams, 1990, p. 4, "Counting"). The various number systems and other methods of representation of quantities throughout the world are also physical aids for calculation (Williams, 1990, pp. 4-7). Throughout history, various methods and instruments for improving the precision, speed, and ease of calculation were devised, culminating in the widespread current-day desktop or handheld calculator (a detailed history can be found in Williams, 1990, pp. 16-33).

These computing devices devoted to the execution of algorithms, both ancient and modern, did not and do not possess the ability to metamorphose into different tools: they can't be programmed. For instance, the abacus allows its user to change the placement of beads in order to store different values. But knowledge of how to use it remains entirely in the user's mind: there are no intermediate components – no software. And after learning how to use the abacus, a person can only use it to perform the same kinds of tasks (arithmetic calculation): knowledge of how to use an abacus does not allow the user to employ it for different purposes. Of course, a user can always invent a novel way to use the abacus, just as someone can invent a novel way to use a wooden stick. But that's entirely new knowledge – it is not a metamorphosis achieved through the exclusive use of existing usage knowledge.

So, in order to find the first examples of programming, a different branch must be followed on this "evolutionary tree". The metamorphosing ability depends on the existence of a software layer in the calculation/computation instruments, as explained in section 2.1. In order for this layer to emerge, it is necessary that the computation instrument does more than just reorganize information present in the original numbers (data): the supplied data must be reinterpreted and converted, originating new patterns and results, new information. This in turn implies that the instrument must possess a mechanical nature, an internal functioning and operation separated from the entered data and from the methods of entry and operation. Michael Williams (1990, p. 34) provides a very specific starting point, for this branching in the evolutionary tree of computing:

> *"Though the various analog instruments were capable of performing a great deal of useful arithmetic, the story of devices that ultimately led to fully automatic computation really starts with the invention and development of mechanical devices to perform the four standard arithmetic functions."*

He locates in the 17th century the first such mechanical machines for automation of calculations, devised by Wilhelm Schickard (1592-1635), Blaise Pascal (1623-1662), and Gottfried Wilhelm Leibniz (1646-1716). The principles demonstrated by these machines were further

developed in the 19th century and early 20th century, most notably by Charles Xavier Thomas de Colmar, in 1820, Frank S. Baldwin, in 1873, and Otto Steiger, in the early 1890s, eventually leading to the large variety of mechanical calculation machines of the 20th century (Williams, 1990, pp. 34-57), such as the mechanical cash registers that are still in our living memory.

All these machines were based on rotating gears or drums, for input of calculation data and output of results (Williams, 1990, pp. 34-49). Again, like the abacus, they could not be programmed, but rather only used to execute specific calculation algorithms.

However, yet another branch of this evolutionary tree appeared in the 19th century: Charles Babbage (1792-1871) invented a machine called the Difference Engine, and in 1822 produced a working demonstration model; in 1830 he produced an improved design (Bromley, 1990). A major novel aspect in both versions of the Difference Engine was that they "*could be modified to perform a variety of calculations*" (Brookshear, 2003, p. 6). But still, they were "*only directly capable of handling polynomials*" (Bromley, 1990, p. 75); they were "*not a* (sic) *general purpose machines. They could process numbers entered into them only by adding them in a particular sequence* (Science Museum, 2004, par. 2), like the aforementioned machines of the 17th century.

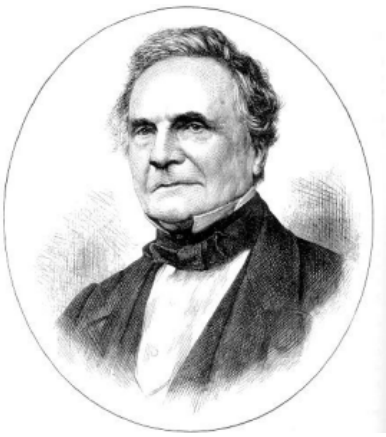


Figure 3 – Charles Babbage, 1791-1871
From: http://www.fourmilab.ch/babbage/figures/babbage.jpg

These were Babbage's first constructs, but he then invented a machine, the Analytical Engine, whose basic plans laid formulated by 1834 (Bromley, 1990, p. 76). This machine could accept instructions in the form of holes in paper cards – a form of data input for machines which had been invented by Joseph Jacquard in 1801, to determine the operation of a loom (Brookshear, 2003, p.7; Bromley, 2003, p. 86). Had the Analytical Engine been built, it would have been the first programmable computer (Science Museum, 2004, par. 2; Brookshear, 2003, p. 6).

> *« The sequence of operations ordered by the barrels included what today we call "loops" and by alternate sequences dependent on arithmetic results that arose during calculations (today known as "branching"). »*
>
> *(Bromley, 1990, p.76)*

In fact, programs were devised for it, even thought the machine itself was never built: Augusta Ada Byron King, Countess of Lovelace, under close guidance of Charles Babbage, included several notes in her English translation of a document written in French describing Babbage's designs (Byron-King, 1843; Menabrea, 1842). Those notes contained "*examples of how the Analytical Engine could be programmed to perform various tasks*" (Brookshear, 2003, p. 6). Augusta Ada Byron King (also known as Lady Lovelace or Ada Lovelace) is for this reason often referred to as "the world's first programmer". However, this happens not to be so, as is pointed out by Bromley and others:

Figure 4 – Augusta Ada Byron King, 1815-1852
From: http://www.cs.kuleuven.ac.be/~dirk/.image/ada_1838.jpg

> *« All but one of the programs cited in her notes had been prepared by Babbage from three to seven years earlier. The exception was prepared by Babbage for her, although she did detect a "bug" in it. Not only is there no evidence that Ada Lovelace ever prepared a program for the Analytical Engine but her correspondence with Babbage shows that she did not have the knowledge to do so. »*
>
> *(Bromley, 1990, p. 88-89)*

> *« Babbage, not Ada, wrote the first programs for his Analytical Engine, although most were never published. »*
>
> *(Kim & Toole, 1999, p. 76)*

Irrespective of this authorship discussion, which is detailed by Green (2000), the programs contained in her notes are the first known examples of computer programs. And it was her writing that brought Babbage's achievement to a wider audience.

> *« And whereas Babbage's groundbreaking work formed the basis of Ada's notes and her thinking, her lucid writing revealed unique insight into the significance and the many possibilities of the Analytical Engine. »*
>
> *(id., ibid.)*

I'll look into the evolution of programming proper in section 2.2.5.

2.2.3. The electric and electronic programmable machines

Babbage's work was not completed nor pursued; thus, modern computers did not stem from it, but rather from the evolution of the other aforementioned mechanical calculation machines of his time. While in the early 19th century the only source of power for automating the operation of such machines was the steam engine, that same century saw the development of electric engines. And the knowledge and technology that led to the creation of the electric engine also made possible for calculation machines to be made using electric circuits instead of "*cams, pins, gears and levers*" (Ceruzzi, 1990, p. 200).

A major change brought by this replacement of mechanical elements by electric circuits was that these are much easier (and flexible) to rearrange (Ceruzzi, 1990, p. 203):

> *« (...) a crucial advantage over mechanical systems, in that their circuits could be flexibly arranged (and rearranged) far more easily. One could arrange relays on a rack in rows and columns, and connect them with wires according to what one wanted the circuit to do. One could further reconfigure a relay system using a switchboard, with cables plugged into various sockets by a human operator. »*

This new feature was central in enabling an evolution of technology, eventually leading to the creation of computers in a modern sense.

The bridge between mere calculators and computers was erected between the mid-1930s and the mid-1940s. During these years, the work of several pioneers led to the creation of machines where elements and features of modern computers were developed, coming all into place in 1945, with the creation of a machine called the ENIAC, that made the final link between the primitive specialized calculating machines and modern computer machinery (Ceruzzi, 1983, p. 8; Ceruzzi, 1990, p. 236). Six central pillars for that bridge were developed within those years.

The **first** such **pillar** is due to Konrad Zuse, from Germany. Between 1935 and 1941, he developed three calculators, each a step forward. The first, called the Z1, was still an entirely mechanical calculator, where a few novel ideas where implemented; he improved his designs to build a machine called Z2, which already employed electric relays in its operation; and then the Z3, the first computer in his series of machines that was centered on electric circuits (Zuse, 1999). "*A person entered numbers and operations on a calculator-style keyboard, and answers were displayed on small incandescent lamps*" (Ceruzzi, 1990, p. 206). This machine represented a tremendous advance, and several crucial technical developments were part of it (Zuse, 1999). The advances developed in these machines did not feed into the development of modern computer architecture due to Zuse's isolation in Germany during World War II and the years that followed it. The following quote provides a nice example of such advances.

Figure 5 – Konrad Zuse, 1910-1995
From: http://www.rtd-net.de/Zuse.GIF

> *« The most unusual feature was undoubtedly the mechanical binary cells that made up the memory. The memory has 64 words with 32 bits (Z1 and Z4). These devices were completely different from mechanisms in contemporary cash registers or desk-top calculators. The elements could be used not only for storage, but also for calculation, for example for address coding. A relay memory would have required about 2500 relays, which would have more than doubled the size and the weight of the Z4 computer. »*
>
> *(Zuse, 1999, "Part 11: Some Conclusions")*

The operation of the machine was generic, since it depended on the actual instructions fed to it. But these instructions only allowed for arithmetic-based and algebra-based computations, there were no instructions that specifically would allow alternative paths to solve a problem:

> *« (…) the conditional branch instruction is missing. In his papers between 1936 and 1945, Konrad Zuse described many scientific and numerical problems he wanted to solve with his machines, but none of these problems required the use of the conditional branch. »*
>
> *(Zuse, 1999, "Part 4: The Z2 and Z3")*

Due to this limitation, Ceruzzi (1990, p. 205), considered it "*perhaps the world's first general-purpose, sequential calculator*". It was more than just that, as was shown in 1998 by a final, if somewhat late recognition of the Z3's potential, when Raul Rojas "*formulated the proof that the Z3 was a truly universal computer in the sense of a Turing machine*" (Zuse, 1999, "Part 4: The Z2 and Z3"). (The concept of a Turing machine is presented at the end of section 2.2.4.) So, in fact, the only element missing from the Z3 to render it an actual computer was the stored-program concept (Zuse, 1999, "Part 10: Konrad Zuse and the Stored Program Computer"), as will be explained in section 2.2.4. The same can be said of Konrad Zuse's later machines, such as the Z4 (which already incorporated a conditional branching instruction) and the Z5. But the Z3 marked Konrad Zuse's major contribution for the development of computers.

The **second pillar** is represented by George Stibitz, from the USA. In 1937, he built a fully electric device, using relays: it could add two binary digits (Ceruzzi, 1990, p. 208). Working at the Bell Labs, in 1939 Stibitz developed a device called the Complex Number Computer, which could perform arithmetic operations on complex numbers[12], something that required the execution of sequences of several arithmetic operations. Subsequent, more flexible, machines were developed by Stibitz from 1943 onward, the first of these being called the Model II (Ceruzzi, 1990, pp. 210-211):

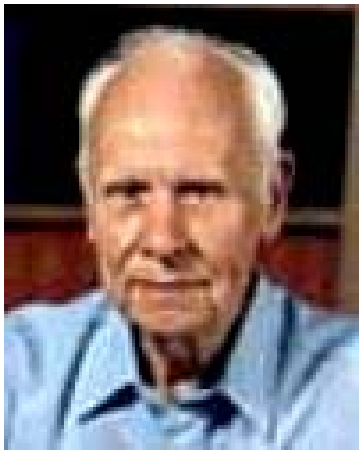


Figure 6 – George Stibitz, 1904-1995
From: http://www.todayinsci.com/S/Stibitz_George/StibitzGeorgeThm.jpg

> *« It mainly solved problems related to directing antiaircraft fire, which it did by executing a sequence of arithmetic operations that interpolated function values supplied to the machine by paper tapes. Like the Complex Number Computer, it was a special-purpose machine; however, its arithmetic sequence was not Relay Calculators permanently wired but rather supplied by a "formula tape" cemented into a loop. Different tapes therefore allowed one to employ different methods of interpolation. The Model II could not do much besides interpolation,* (…) *»*

The following machines by Stibitz, called Model III (1944) and Model IV (1945) were a bit more sophisticated, "*having the ability not only to perform interpolation but also to evaluate* (…) *ballistic equations describing the path of the target airplane and of the antiaircraft shell.* (…) *paper tape directed which* (…) *functions the machine was to evaluate. Thus, the Models III and IV were the first of the Bell Labs digital calculators to have some degree of general programmability, although neither was a fully general-purpose calculator*" (Ceruzzi, 1990, p. 211).

[12] A "complex" is a construct of a pair of numbers: one of the numbers is called the "real" part of the complex; the other is called the "imaginary" part. To write these numbers, usually the letter "*i*" (sometimes "*j*") is used to indicate the imaginary part. Thus a complex whose real part is 3 and whose imaginary part is 4 is written as 3+4%*i* (or simply 3+4*i*). This construct is called a "complex number", because if one considers *i* to be a number, whose value is $\sqrt{-1}$, then all arithmetic used for real numbers is also valid for "complex numbers" such as 3+4*i*.

Third pillar: Howard Aiken, from the USA. Aiken also sought to improve the existing calculators of his time, to render them more useful for scientific applications, particularly in order to enable them to perform iterative resolutions (Ceruzzi, 1990, p. 213). His first plans are from 1937, and were further developed between 1941 and 1942 in cooperation with an IBM team led by James W. Bryce, in New York. In 1943, this team ultimately designed and built the machine devised by Aiken, which was called the ASCC (Automatic Sequence Controlled Calculator), but later became known as the Harvard Mark I (Ceruzzi, 1990, pp. 214-218). "*Aiken's "Mark I" was the first large computing device to be made public (in 1944), and as such its name is appropriate – it marks the beginning of the computer age, despite the fact that it used mechanical components and a design that soon would become obsolete*" (Ceruzzi, 1983, p. 7).

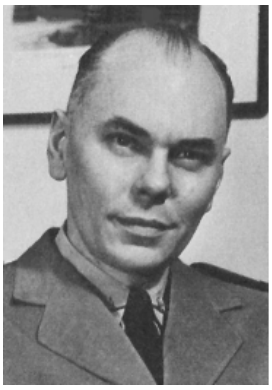


Figure 7 – Howard Aiken 1900-1973
From: http://virtualmuseum.dlib.vt.edu/Images/howard_aiken.jpg

At the **fourth pillar**, one can place John V. Atanasoff, also from the USA. The previous contributors I mentioned focused their development both on earlier mechanical technology and the novel electric circuits. Those electric circuits, however, were mainly based around a type of electric component called "relay", which contained moving parts. Faster computations would require circuits whose behavior was based only on electronic components. According to himself, Atanasoff's decision on this regard, which was to do computations and data storage entirely in electronic components, dates from 1937, although he only built the first circuits in 1939 (Ceruzzi, 1990, p. 227). His machine was developed between 1939 and 1942, and focused on the resolution of linear systems of equations, but it was never refined beyond the point where occasional malfunctions would not occur (Ceruzzi, 1990, p. 230). Also, it was not possible to alter the sequence of operations that the machine went through (Ceruzzi, 1983, p. 111).

Figure 8 – John V. Atanassof 1903-1995
From: http://www.propertyinbulgaria.net/Assets/Images/Atanassoff.jpg

The **fifth pillar** is Helmut Schreyer, a colleague of Konrad Zuse, in Germany. Zuse was developing his electrical calculators, but Schreyer's plans, like Atansoff's, were to rely rather on electronic components. His work on computing took place only between 1941 and 1943, but unlike the contributors mentioned previously, he focused on using the electronic components to conduct logical operations[13], rather than arithmetic ones (Ceruzzi, 1990, pp. 231-232).

Figure 9 – Helmut Schreyer 1912-1984
From: http://www.atariarchives.org/deli/time_line_22.jpg

The **sixth and last pillar** is called *Colossus* and is also centered on logic operations. Colossus was the name of ten machines designed and developed during World War II at Bletchley Park[14], for the Department of Communications of the British Foreign Office, to help British intelligence services in the decoding of German messages. Max H. A. Newman was the proponent of a first design, called Heath Robinson (after a British cartoon character). "*Installed in June 1943, Heath Robinson was unreliable and slow, and its high-speed paper tapes were continually breaking, but it proved the worth of Newman's method*" (Copeland, 2000).

[13] Logical operations are conducted on logical values (typically, TRUE and FALSE) rather than on numerical values. Schreyer's initial circuit could perform AND, OR and NOT operations on three separate inputs.
[14] A Victorian estate about fifty miles north of London (Ceruzzi, 1990, p. 232).

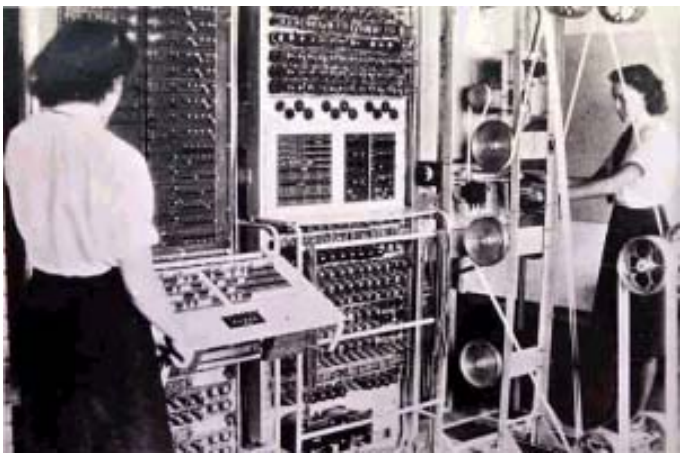


Figure 10 – Colossus
From: http://www.computersciencelab.com/ComputerHistory/HtmlHelp/Images/Colossus.gif

Alan Turing, a mathematician who was also involved in the wartime code-breaking effort of the British Foreign Office, and had earlier written a fundamental paper with the mathematical analysis of a general-purpose computational machine (as I explain in sections 2.2.5 and 2.2.6), suggested that Newman contact another person working for the Foreign Office, engineer Thomas Flowers (Copeland, 2000). Flowers designed "*a fully electronic machine with a similar function to Heath Robinson.* [He] (…) *received little official encouragement* (…) *but proceeded nonetheless, working independently at the Post Office Research Station at Dollis Hill. Colossus I was installed at Bletchley Park on 8 December 1943*" (*id.*). Its job was to compare two paper tapes, one with an encrypted German message, and the other with a previously-deducted mathematical supposition about what the encoding on that message might be. This comparison would yield a "clue" about the exact code used on the encrypted message, and that clue would lead to another tape and so forth, until the message was decoded (Ceruzzi, 1990, p. 233). The main feature of the original Colossus was its operation at the speed of five thousand characters per second, relying on electronic components for both calculation and also to read in data at high speed, keeping synchronized the two different paper tape sources. A second version, built in 1944, brought with it another important development, "*consequent upon a key discovery by cryptanalysts Donald Michie and Jack Good*" (Copeland, 2000): it "*contained circuits that could automatically alter its own program sequence. If it sensed a potentially meaningful pattern, a circuit permitted it to redirect its stepping through the internal table in such a way that would more quickly find a path to a solution*" (Ceruzzi, 1990, p. 234-235). The Colossus machines, however, despite all these advances, could only deal with the specific problem of decoding encrypted messages, and only messages encrypted in the style employed by the German military at the time of World War II. They "*did not perform ordinary arithmetic, nor* (…) *solve other logical problems not cast in the same mold as those for which* [they were] *designed*" (Ceruzzi, 1990, p. 235).

Figure 11 – Maxwell Herman Alexander Newman (1897-1984) and Thomas H. Flowers (1905-1998)

From: http://www-groups.dcs.st-and.ac.uk/~history/BigPictures/Newman.jpeg and http://wwwusers.brookes.ac.uk/03088589/images/tf1.jpg

The technological developments mentioned in this section, and their associated conceptual advances, put the bridge in place from old-style calculators to modern-day computers and computing. But while each of these machines brought an important element, they remained separate abilities. The first machine to put them together and thus considered as the last element, the link between calculators and computers was the ENIAC computer, built in 1945 at the Moore School of Electrical Engineering at the University of Pennsylvania, under the efforts of John Mauchly and John Presper Eckert (Ceruzzi, 1983, p. 107; Ceruzzi, 1990, p. 236). The name stands for Electronic Numeric Integrator And Computer (Ceruzzi, 1983, p. 127).

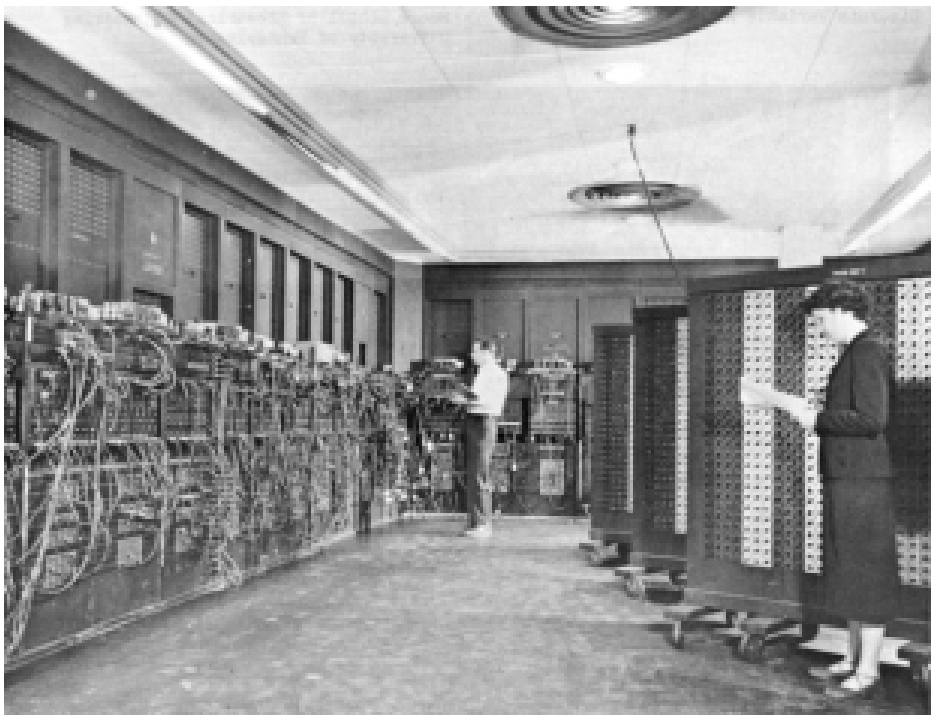


Figure 12 – ENIAC

From: http://www.computersciencelab.com/ComputerHistory/HtmlHelp/Images/eniac1.jpg

The ENIAC also brought forth some novel ideas; it was not a mere contraption of all previous developments[15]. The most important: it was programmable. The sequence of operations of the ENIAC could be altered by plugboards (Ceruzzi, 1983, p. 111), something the Colossus also achieved, but in a limited way (and the Colossus was under military secrecy at the time, thus the ENIAC was developed independently). But perhaps its greatest contribution was the fact that it worked, thus being the first machine to prove **publicly** that electronic devices could be used reliably for computations (Ceruzzi, 1983, p. 111; Ceruzzi, 1990, p. 236). Without using electronics, extremely high calculation speeds could not be achieved, and computers could not evolve from calculation aides to the novel machines they became.

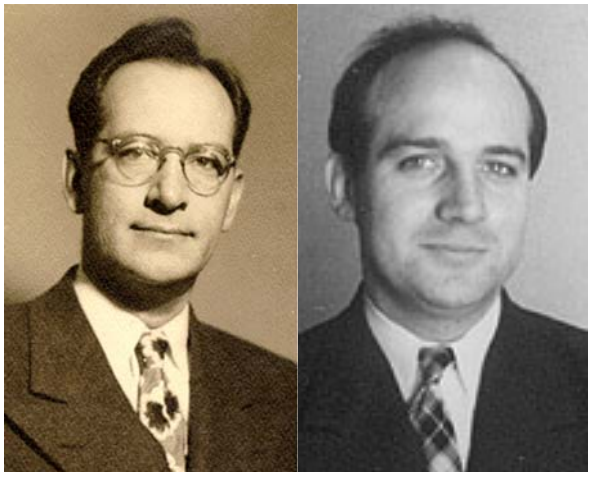


Figure 13 – John Mauchly & John P. Eckert (1907-1980) and (1919-1995)

From:
http://www.library.upenn.edu/exhibits/rbm/mauchly/img/maupor.gif
and
http://www.luckbealady.com/images/ENIAC/Portrait_at_Moore.JPG

[15] In fact, it was even a step back in one aspect: it embraced a decimal notation for recording numbers internally, instead of the much faster binary notation already developed by then and still in use today.

« Charles Babbage first conceived of the idea of an automatic digital computer. John V. Atanasoff, together with Helmut Schreyer and perhaps others [most notably Konrad Zuse]*, first conceived of an electronic digital computer. John Mauchly and J. Presper Eckert, as leaders of a skilled team at the Moore School, invented the first working electronic numeric digital computer. »*

(Ceruzzi, 1983, p. 111)

« It deserves its fame not for its electronic circuits, which several other machines already had, nor for its architecture, which although ingenious and full of promise was rejected by subsequent designers. One should rather remember the ENIAC as the first electronic machine to consistently and reliably solve numerical problems beyond the capability of human and in many cases relay computers as well. »

(Ceruzzi, 1990, p. 236)

« It was not fully a " "general-purpose computer" – for example, it could not solve large systems of linear equations as Atanasoff's machine was designed to do. But its ability to be reconfigured to perform an almost limitless sequence of steps, including iterative loops of operations, sets the ENIAC apart (...)*, and places it astride the categories of "calculator" and "computer." »*

(Ceruzzi, 1990, p. 241)

2.2.4. The appearance and evolution of computers

In spite of all the advances, all the machines mentioned in the previous section could not be considered computers as described in section 2.1, able to metamorphose into different tools. They still missed a final element. Those early machines would be supplied data to work with, and would produce data in return (calculation results). But their operation was strictly dependent on their physical organization. Even the ENIAC, which could be programmed by reorganization of cables on plugboards, was not actually interpreting instructions: in fact, the unplugging and plugging of cables meant that each time it was actually being physically dismantled and rewired into a different, special-purpose, machine (Ceruzzi, 1983, p. 121).

A crucial evolution was thus the moment from which machines could receive and operate on programs as they would on data. Modern computers do just that: when they execute an instruction, they simply proceed to reorganize data, by changing electric current and charge in electronic devices. Computer software, as seen in section 2.1, comes into existence from this very organization of electric currents and charges in circuits. But there is no physical change other than that: the physical machine and its elements remain always the same. This final element, defining the first computers, is called **the stored-program concept** (its history and a different perspective on its relevance are presented in Ceruzzi, 1983, pp. 134-141).

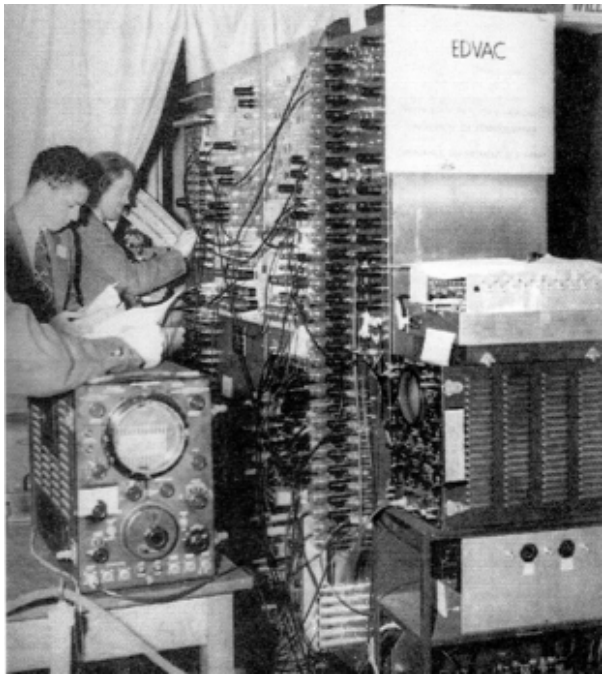


Figure 14 – EDVAC
From: http://tecweb.tol.itesm.mx/A00742413/electronic.jpg

The first machine designed to possess this ability was ENIAC's successor, called EDVAC (Electronic Discrete Variable Automatic Computer), whose design was complete in 1945 (von Neumann, 1945). The EDVAC is also significant in the history of computing because it was the machine where two concepts met, which would dictate the evolution of further computer machines ever since: the mathematical analysis of Alan Turing and the technological computational model (a.k.a. architecture) of John von Neumann.

Alan Turing had, nine years before the development of the EDVAC, in 1936, performed the mathematical analysis behind the very concept of computing, where he established the equivalence of data and instructions (Turing, 1936). He didn't use any machine for his analysis; instead, he conceived a mathematical abstraction of one, what has seen been known as a "Turing machine". In this machine, there isn't any distinction between the instructions and the data: the machine simply receives symbols, and **internally** reacts to them, regardless of whether one sees those symbols as "instructions", as "data", or as any combination of them.

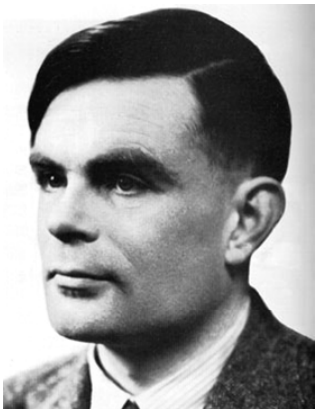


Figure 15 – Alan M. Turing
1912-1954
From: http://aima.cs.berkeley.edu/graphics/turing.jpg

Turing's contributions, from the point of view of software and programming languages in particular, are described in section 2.2.6. From the hardware point of view, his "machine" was simply a long strip of paper, potentially infinite in size, where only one single space at a time would be available for reading, writing, or erasing. This "Turing machine" would also have the ability to move forward and backward along the tape and place itself, internally, in one of a finite number of

states. The functioning of this machine was entirely determined by the combination of the symbol it would read from the paper strip with the current state of the machine; this combination would result in moving the tape to the left or right, write a new symbol to the tape, or erase a symbol (Mitchell, 2003, p. 14; Turing, 1936).

> *« Turing* [provided] (…) *for the first time a logical rigorous definition of what it means to compute a solution to a problem by some definite process. That definition had* [to] *be good enough to convince the mathematical logicians, who were a skeptical lot, to begin with. It did not really have to convince those who worked with computing machines: they went merrily on with their computing, oblivious to the debate over the logical fundamentals. But as stored-program electronic computers with enormous power and speed appeared after the war, such a definition was needed. Turing's has served that need ever since. »*
>
> *(Ceruzzi, 1983, p. 136)*

While Turing provided the mathematical proof for the feasibility of modern computers, his actual impact on the technical development of the first computers with stored programs was important but less influential on future designs. Turing was working during the Second World War for the British Foreign Office, assisting in the code-breaking effort, but was not actively involved in the Colossus project, beyond what I mentioned on p. 37 (Napper, 1999); he did participate in further computing projects in Britain and in 1945, a few months after the original EDVAC report (von Neumann, 1945), Turing wrote a "*relatively complete specification of an electronic stored-program general-purpose digital computer*" (Copeland, 2000), with details on engineering, components, and "*specimen programs in machine code*" (*id.*). A computer was being built according to his specification, but he left the company in charge midway through the project. Afterwards, he was involved in the development of other British computers, but these designs would eventually die out, and therefore have not been a significant influence on modern computers (*id.*).

In modern computers, the technological model/architecture is due to the Hungarian mathematician John von Neumann (b. János Neumann). This was the architecture developed for and employed in the EDVAC, and is still the major computer architecture in use today. Basically, it consists of a processing unit, connected to a memory unit. The processing unit collects one instruction from the memory at a time, then collects any necessary data for that instruction, also from the memory unit, and processes the data (von Neumann, 1945).

Figure 16 – János Neumann 1903-1957
From: http://www.cs.cmu.edu/~mih aib/whoswho/john-von-neumann.gif

This architecture's assumption on instructions being executed one at a time created a bottleneck on processing power, which has been dubbed "*the von Neumann bottleneck*" (Backus, 1978); it renders difficult any attempt to distribute the workload by various processors, working in concert, or to handle different workloads at the same time. Several techniques have been developed since, to overcome some of these limitations, but there has also been considerable effort on the proposal of novel computational architectures (Gruau *et al.*, 2004; Tosic, 2004). However, these have yet to reach adequate levels of technical efficiency and reliability, as well as economical feasibility (Johnston *et al.*, 2004).

Many computing projects took place after the EDVAC; valves where entirely replaced by transistors, and these in turn by integrated circuits, allowing the full processing power of a computer to be harbored in a chip – a microprocessor. These advances tremendously reduced the design, manufacturing, and maintenance costs of computers. Originally hand-made in labs, by the mid-1970s computers besting the computational power of the computer mastodons of the 1940s and 1950s were already being sold as do-it-yourself kits (Kopplin, 2002). "*By the 1990s a university student would typically own his own computer and have exclusive use of it in his dorm room*" (*id.*).

2.2.5. The first programs

As mentioned at the end of section 2.2.1, the first computer programs were devised for Babbage's Analytical Engine. Entering a program in the Analytical Engine would consist of adequate arrangement and placement, into columns of operating discs, of punch-cards for operations and storage of intermediate values[16]; thus, the programming process would be a manual, paper-based planning activity, taking place beforehand. Keeping in mind that the overall goal was to calculate a mathematical function, this planning had to be done in two phases. Firstly, the mathematical expression to be calculated would have to be transformed into an arithmetic-based procedure, using the actual means for transformation of values available in the Analytical Engine: adding, subtracting, multiplying, dividing, and storing intermediate results. Secondly, it would be necessary to define the control methods for determining the actual operations to be made during the calculation process, based on the intermediate results stored (Byron-King, 1843, Note A & Note B; Bromley, 1990, pp. 87-89).

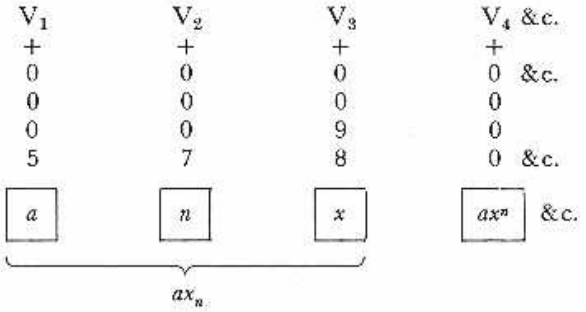


Figure 17 – Preparing input and output for the Analytical Engine

From: Byron-King, 1843, Note B

During the planning procedure, the programmer of the Analytical Engine would have to consider several specific operating details, such as the actual storage location of values, for instance. This would be done resorting to tables, representing features of the machine.

DIAGRAM BELONGING TO NOTE D

Number of Operations	Nature of Operations	Variables for Data						Working Variables									Variables for Results	
		$^1V_0$	$^1V_1$	$^1V_2$	$^1V_3$	$^1V_4$	$^1V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	$^0V_{14}$	$^0V_{15}$	$^0V_{16}$
		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		m	n	d	m'	n'	d'										$\frac{dn'-d'n}{mn'-m'n}=x$	$\frac{d'm-dm'}{mn'-m'n}=y$
1	×	m	.....	.....	.....	n'	.....	mn'										
2	×	.....	n	.....	m'	.....	.....	.....	$m'n$									
3	×	.....	.....	d	.....	.....	.....	.....	.....	dn'								
4	×	.....	0	.....	.....	.....	d'	.....	.....	.....	$d'n$							
5	×	0	.....	.....	.....	.....	0	.....	.....	.....	.....	$d'm$						
6	×	.....	.....	0	0	.....	.....	.....	.....	.....	.....	.....	dm'					
7	−	.....	.....	.....	.....	.....	.....	0	0	.....	.....	.....	.....	$(mn'-m'n)$				
8	−	.....	.....	.....	.....	.....	.....	.....	.....	0	0	.....	.....	.....	$(dn'-d'n)$			
9	−	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	0	0	.....	.....	$(d'm-dm')$		
10	÷	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	$(mn'-m'n)$	0	.....	$\frac{dn'-d'n}{mn'-m'n}=x$	
11	÷	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	0	.....	0	.....	$\frac{d'm-dm'}{mn'-m'n}=y$

Figure 18 – Analytical Engine program

From: Byron-King, 1843, Note D

[16] "(…) *these columns are designated* (…) *the Variables. The origin of this appellation is, that the values on the columns are destined to change, that is to vary, in every conceivable manner.* (…) *The columns are called Variables on a ground wholly unconnected with the analytical distinction between constants and variables.*" (Byron-King, 1843, Note B).

For instance, in the table presented in Figure 17, on the previous page, the variables V_1, V_2, and V_3 have been loaded with the values 5, 7, and 98. The notes inside the squares below these columns tell that these are values for the algebraic elements *a*, *x*, and *n*. Finally, the variable V_4 is to receive the result, and the notation tells us that the original elements are to be combined into the expression ax^n. (The values in variable V_4 could be referred to just as those in the other variables, and thus employed in other calculations.)

The rest of the program would specify the sequence of operations to perform in order to achieve the desired results. For instance, in the ax^n example there would have to be 6 multiplications (to produce 98^7), and a seventh multiplication (to multiply the previous result by 5, the value of *a*). The tables necessary for this kind of elaborate specification could easily grow quite large, and the programmer had to keep in mind the actual way in which the machine would interpret the control cards, rather than just the mathematical notation, as is demonstrated by Figure 18, on p. 42, which presents the program for the calculation of the following two simple functions, *x* and *y*:

$$x = \frac{dn' - d'n}{mn' - m'n} \qquad y = \frac{d'm - dm'}{mn' - m'n}$$

Since the actual instructions were not stored in variables or otherwise accessible to the program execution itself, in effect this programming of the Analytical Engine was an advanced planning methodology for what I referred to, in section 2.2.1, as the reorganization of the physical elements of the computer. In other words, the control and variable cards were the last pieces of the machine: programming was the planning of their proper placement, as if one would be, each time, re-assembling a bit of the machine.

This style of programming, *i.e.*, using some planning for determining the sequence of operations to be conducted by the specific elements of the machine, was also a fundamental consideration of the programming process of early computers such as the ENIAC (*vd.* p. 38):

> *« To reprogram the ENIAC you had to rearrange the patch cords that you can observe on the left in the prior photo* [(*Figure 12*, on p. 38)]*, and the settings of 3000 switches that you can observe on the right.* (…)
>
> *Circumference = 3.14 * diameter*
>
> *To perform this computation on ENIAC you had to rearrange a large number of patch cords and then locate three particular knobs on that vast wall of knobs and set them to 3, 1, and 4. Reprogramming ENIAC involved a hike. »*
>
> *(Kopplin, 2002)*

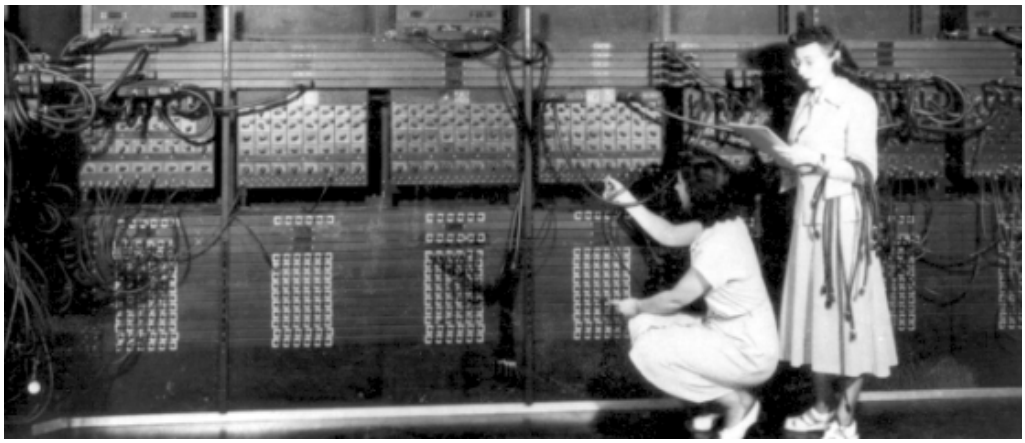


Figure 19 – Reprogramming ENIAC
From: http://www.computersciencelab.com/ComputerHistory/HtmlHelp/Images/eniac4.gif

This planning approach was only replaced by one more focused on requesting a machine to conduct specific operations, without having to physically change, when computers were developed

with the ability to accept instructions and interpret them, rather than be re-wired. The EDVAC, as mentioned in section 2.2.4, was the first computer that employed this "stored-program" concept, its instructions able to be read and interpreted without human intervention. Zuse's Z3, Z4 and Z5 did not store programs, but could read their instructions from tape, effectively achieving the same purpose, albeit in slower fashion.

Since computers' memory – a condition that lasts up to present day – only stores numbers[17], the instructions themselves were also stored as numbers. So, entering instructions into memory implied deciding which numbers to enter, in which memory address. This could be done by different methods, depending on the actual machine, from setting hand-switches on and off (to represent a 0 or a 1 and thus any number, in binary notation), to using punched cards.

This tedious process meant that to add two numbers, for instance, one representing the price, and the other the shipping charge, this would involve:

1. copying one number from its location in memory to a machine register (say, register number 5);
2. copying the other number from its location in memory to another machine register (say, register number 6);
3. adding the contents of both registers and storing the result in yet another register (for instance, register number 0);
4. copying the result in the third register to the location in memory where the calculation result is to be stored.

Not only this is more complex than writing on paper "total=price+shipping charge", but the whole process had to be translated into numbers – usually in binary format. Brookshear (2003, p. 80) provides a sample translation of such a sequence of instruction into a machine, which I employ here, just adding the translation of the instructions into binary format:

Instruction (binary)	Instruction (hexadecimal)	Meaning
1010101101100	156C	Load register 5 with the bit pattern found in memory at address 6C.
1011001101101	166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
101000001010110	5056	Add the contents of register 5 and register 6 and leave the result in register 0.
11000001101110	306E	Store the contents of register 0 in the memory cell at address 6E.

Table 1 – Machine programming in binary and hexadecimal

Each 1 or 0 would represent a hand-switch to be turned on or off; or a hole to be punched (or drilled) in a card or not. Such method for entering code was obviously not only tedious but error-prone, so frequently the initial code entered into a machine was to allow the operation of a more human-adequate input device, such as a keyboard.

> *« Using the Z3 to solve a problem involved first of all writing out the sequence of commands to perform arithmetic operations and send to or retrieve data from storage. This sequence was then punched into a filmstrip, using an eight-hole pattern. Once this code was prepared, initial values were entered into the keyboard in floating-point, decimal form; the machine then*

[17] Alan Turing, in his theoretical analysis of 1937, didn't restrict the symbols processed by computing machines to be only numbers (Turing, 1937).

converted the numbers into binary, carried out the sequence, and displayed the result on the lamps after reconverting it back to decimal notation. »

(Ceruzzi, 1990, p. 206).

One advantage of entering decimal numbers with a keyboard is that it's much less error-prone to use decimal (or hexadecimal) notation than it is to enter long sequences of 0s and 1s.

Before moving on to the next section, I'd like to present a personal note. My first contact with this reality was in September 1985, when I bought my first computing magazine, a British magazine called *Your Spectrum* (Figure 20). At the end of it, there was a program by Stuart Jamieson (Jamieson, 1985), for a game called "Mac Man" (a Pac-Man[18] clone, Figure 21). This was the exciting presentation of the process of entering hexadecimal instructions and data:

Figure 20 – Your Spectrum issue 12
From: ftp://ftp.worldofspectrum.org/pub/sinclair/magazines/YourSpectrum/Issue12/YRCover12.jpg

Figure 21 – Mac Man playing screen
From: http://www.users.globalnet.co.uk/~jg27paw4/pourri/mac_man.gif

> *« Before giving you the details of the game (Only hermits won't know what it's all about! Ed.), here's how you get that code into your Speccy[19]. First type in the machine code loader and SAVE it to tape. Next type in the Hex loader program and RUN it. It'll accept eight bytes at a time (without spaces) and then ask for a checksum (which is given after the eight Hex pairs). You'll then be asked to SAVE the code after the short loader program. This done, reset the Spectrum, rewind the tape to the beginning, enter LOAD "" and press the Play button on your cassette machine. It's as easy as that! »*
>
> *(Jamieson, 1985)*

Piece of cake, indeed! I tried four times to enter the code into my ZX Spectrum, each time carefully entering each number – but still something would fail, and the program wouldn't run! And the magazine printing of code wasn't that perfect, so I couldn't be too sure I was reading the numbers right, and I eventually gave up (on Mac Man, not on the Spectrum or on machine code).

Here's a sample of the code to be entered in that "hex loader" program. Thanks to the Internet, I now have access to it without having to enter it all from printed matter!

```
26490 3E 07 32 48 5C 32 8D 5C =566     26570 68 75 21 28 68 01 0E 01 =414     26650 C9 01 20 00 09 06 20 7E =407
26498 32 8E 5C 32 8F 5C 32 90 =763     26578 C5 E5 C5 06 08 C5 CD 01 =1040    26658 17 77 2B 10 FA C9 20 4D =761
26506 5C AF D3 FE 21 00 40 11 =846     26586 68 C1 10 F9 C1 E1 7E 23 =1141    26666 41 43 4D 41 4E 20 57 52 =553
26514 01 40 36 00 01 00 18 ED =381     26594 E5 CD 36 69 E1 3E F7 DB =1346    26674 49 54 54 45 4E 20 42 59 =575
26522 B0 21 00 58 11 01 58 36 =457     26602 FE CB 47 28 10 3E BF DB =1056    26682 20 53 54 55 41 52 54 20 =547
26530 07 01 00 03 ED B0 CD C2 =823     26610 FE CB 47 28 08 C1 0B 78 =900     26690 4A 41 4D 49 45 53 4F 4E =598
26538 67 3E F7 DB FE CB 47 C8 =1359    26618 B1 20 D5 18 CD C1 C9 21 =1078    26698 20 7F 31 39 38 34 20 2E =451
26546 3E BF DB FE CB 47 CC 78 =1324    26626 1F 50 CD 0E 68 21 3F 50 =610     26706 20 47 55 49 44 45 20 4D =507
26554 69 06 C8 76 10 FD 18 CC =926     26634 CD 0E 68 C9 06 08 C5 E5 =964     26714 41 43 20 4D 41 4E 20 52 =498
26562 11 CD 76 0E EE 06 27 CD =842     26642 CD 1B 68 E1 24 C1 10 F6 =1052    26722 4F 55 4E 44 20 54 48 45 =567
```

(Jamieson, 1985, 5% of total code)

[18] An excellent history and description of the original Pac Man game and its sequels can be found on the Web (Robertson, 2002).
[19] The name by which ZX Spectrum fans caringly referred to their computers.

2.2.6. Programming languages are developed

A major limitation of early computer programming, as described in section 2.2.5, was its rigidness: a sequence of instructions could be devised and entered into the machine, and was executed – several times, if necessary – until the desired results were complete. This sequence could not include optional methods of resolution, alternatives to account in mid-calculation for unexpected data. As Ceruzzi points out, it's easy to code a computer in order to evaluate an expression like "$x^3 + 4x^2 + 3$ *for successive values of x from 1 to 10* (…)*: the programmer simply specifies the sequence of arithmetic for that formula and arranges to have it accept the successive values of x as input*" (Ceruzzi, 1983, p. 143).

This approach is fine for some problems, but not for others. For instance, what about if one wants to compute an employee's weekly payroll check (*id.*, *ibid.*)? The formula depends on whether the employee worked overtime or not, whether such overtime work has been approved by the appropriate manager, which in turn requires knowledge of who was the manager in charge at that time and date, where such approval may be recorded, etc. In fact, it would be troublesome to come up with a solution other than specifying a different formula for each and every situation. The algebraic notation used in the formula above simply isn't adequate to describe all the necessary tests and decisions that must be taken.

This was recognized as early as 1945 by Konrad Zuse and independently in 1947 by John Mauchly, "*Thus is created a new set of operations which might be said to form a calculus of instructions*" (as cited by Ceruzzi, 1983, p. 143). Konrad Zuse did present a sketch of his ideas for such a calculus in 1945, calling it "*Plankalkül*" (*id., ibid.*; Bauer & Wössner, 1972), which was finished in 1946, but was not published until 1972 (Zuse, 1999, "Part 5: Konrad Zuse's Plankalkül Programming Language"). The Plankalkül can be seen as the first example of what is now called a **programming language**.

However, Plankalkül was never implemented. The first steps from ancient programming of arithmetic formulas towards modern programming languages came from a different source: the evolution of numerical stored-programs, as described at the end of section 2.2.5. It started with the recognition that instead of using a table of numerical instructions and their meanings, such as the one presented in that section (and partly included here, for convenience), programmers could simply use a textual mnemonic, to assist them when planning the programs on paper. The following table presents a sample list of mnemonics for the instructions from the previous section. They add up two values stored in memory location and store the result in another memory location[20] (Brookshear, 2003).

Instruction (hexadecimal)	Explanation	Mnemonic
156C	Load register 5 with the bit pattern found in memory at address 6C.	LD R5, Price
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.	LD R6, ShippingCharge
5056	Add the contents of register 5 and register 6 and leave the result in register 0.	ADD R0, R5 R6
306E	Store the contents of register 0 in the memory cell at address 6E.	ST R0, TotalCost

Table 2 – Machine programming with assembler mnemonics

[20] The programmer would note down the actual memory addresses represented by the textual labels.

> *« When these techniques were first introduced, programmers used such notation when designing a program on paper and later translated it into machine-language form. It was not long, however, before this translation process was recognized as a procedure that could be performed by the machine itself. Consequently, programs called* ***assemblers*** *were developed to translate other programs written in mnemonic form into machine-compatible form.* (…) *Mnemonic systems for representing programs were in turn recognized as programming languages called* ***assembly languages****. »*
>
> *(Brookshear, 2003)*

These assembly languages were thus the first programming languages, but they possessed a major limitation: they were completely tied to the hardware architecture of the machine where they were intended to be run. One could not pick up an assembly-language program intended for a hardware architecture and input it in an assembler program for another hardware. The next step in the evolution of programming languages was the realization that the programmer could program independently of the hardware architecture, at least for expressing simple ideas such as Price + Shipping Charge = Total Cost.

As is often the case, the mathematical background supporting this process of automatic conversion from human-readable form to machine-readable form had been developed much earlier. And once more, the people that took this missing step did it independently. Heinz Rutishauser (who worked with Konrad Zuse), recognized such need in his 1951 lecture "*Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen*" (Bauer, 2002; Ceruzzi, 1983). And Rutishauer was rediscovering a concept first formulated by Alan Turing in 1936. Before general-purpose computing machines existed, Turing mathematically described and analyzed the functioning of such a machine, and due to this achievement general computing machines are also frequently referred to as "Turing machines".

> *« It is possible to invent a single machine which can be used to compute any computable sequence. If this machine* $\mathcal{U}$ *is supplied with a tape on the beginning of which is written the S.D* [(standard description)] *of some computing machine* $\mathcal{M}$*, then* $\mathcal{U}$ *will compute the same sequence as* $\mathcal{M}$*. In this section I explain in outline the behaviour of the machine. The next section is devoted to giving the complete table for* $\mathcal{M}$*. »*
>
> *(Turing, 1936, pp. 241-242)*

Turing's conception and its analysis are a mathematical proof of the possibility of programming the resolution to a problem, not caring about the technological specification of the machine where that resolution will be executed: it suffices that the resolution first feeds the actual machine with the specification of the virtual machine it was conceived to run upon.

In other words, it means that instead of worrying about issues such as "in which electronic register or memory address should I save the values I want to add" or "which electronic register or memory address will receive the resulting values", a programmer can think in terms such as "*TotalCost* will receive the result of adding the values *Price* and *ShippingCharge*" – a considerable advantage, allowing the programmer to focus more in the problem at hand, and less on the machine that will execute the program intended to solve that problem.

In practice, this is not absolutely so, because real machines have physical limitations, while Turing's conception may not have[21], and therefore a professional computer program, no matter how

[21] "*If a computing machine never writes down* [on the tape] *more than a finite number of symbols of the first kind, it will be called* ***circular****. Otherwise it is said to be* ***circle-free****.* (…) *A sequence is said to be computable if it can be computed by a circle-free machine. A number is computable if it differs by an integer from the number computed by a circle-free machine*" (Turing, 1936, p. 233).

abstract, isn't absolutely independent of the hardware it is running on: it must account for hardware limitations or failures (mostly, how much memory is available, how fast the machine is, and whether input and output devices are operating as expected). But to a great extent, such considerations are remitted to a different concern area; programming becomes a translation into a foreign language: a programming language. It ceases to be a matter of converting a problem into the computing machine's parts, devices and remaining paraphernalia.

Programming languages have thus since evolved from the initial assembly languages[22], in order to rise in abstraction from the physical properties of the machines they eventually run on, trying to be closer to the programmer's mind frame when dealing with a problem. But different people think differently; and different problems often require different solving methods, or are easier to understand and tackle under different perspectives. For this reason, the evolution of programming languages was not one of sequential improvement and refinement, but rather a biological-like evolution, branching from the machine languages of the physical machines to a multitude of languages at various levels of abstraction and with different background philosophies or paradigms.

While providing a full historical account of programming languages is beyond the scope of this thesis, I believe a short review of their evolution and varieties will benefit the reader, by providing a background of the most famous languages and programming models.

The first widely-used programming language the was Fortran, created between 1954 and 1957 "*by an IBM team lead by John W. Backus*" (VV.AA., 1998, ch. 1-1), which introduced **expression abstraction**. This means that programmers performing arithmetic calculations could use their familiar mathematical notation, rather than select and carefully order the specific machine-instructions and determine from where in the machine the values should be retrieved or where the results should be stored. The advantages of such a step, for instance, can be as clear has being able to specify a multiplication, without having to worry whether the machine even had a multiplication instruction at all (Burns & Davies, 1993, pp. 2-3)! This approach is known as the **imperative paradigm**, since a program is composed, essentially, by "orders", each following the previous one in sequence, with the entire flow of execution controlled by various instructions[23] (Baranauskas, 1995).

Several other languages were created following the imperative paradigm, trying to address some of Fortran's shortcomings, proposing novel developments and techniques, or simply inspired by it. One is the extremely popular BASIC language (mentioned in section 87, p. 87), but essentially, besides Fortran there are two other main branches to this family. One of these branches originated in 1958 with the language IAL[24], soon renamed Algol (Naur, Backus *et al.*, 1960), which improved data abstraction, allowing better data organization and processing; from this evolutionary line were born famous and widespread languages such as Pascal[25], C[26], and Ada[27], among others. The other branch is represented by verbose languages, which attempt to be close in syntax and readability to the English language, by avoiding the use of mathematical notations or similar operators and using extra, redundant text (this approach has also been used by child-oriented languages GRAIL and HANDS, described on pp. 157-158). These were focused on the processing and retrieval of information such as business records, rather than numerical calculations. It is best

[22] Which are still used today, but only when performance requirements or low-level hardware management concerns require the programmer to think in hardware-specific terms.
[23] These included instructions to "force" the execution to proceed from a specific point in the list of instructions, a method which later was mostly abandoned (Dijkstra, 1968), and for organization of the flow in a structured form, such as subprocedures (sets of commands with a specific purpose, that can be called upon to complete that purpose, before proceeding) and cycles (repeating a set of instructions a number of times or subject to specific conditions).
[24] International Algebraic Language (Perlis & Samelson, 1958).
[25] Wirth, 1993.
[26] Ritchie, 1993.
[27] Gargaro, 1993.

known by the widely used language COBOL, created in 1959 (Sammet, 1978), which was inspired by the language B-0 (later known as Flow-matic). This, in turn, was created in 1956 by Grace Hopper (Gorn, 1957), culminating a development initiated in 1951 with the A-0 compiler, a project that also produced a language released in 1957, under the name Math-matic (Sammet, 1972). Another early verbose language was the APT language, developed in 1957 for use in industrial machines with numerical controls (Ross, 1978). There have been many specialized languages for machine control and information processing, most notably the current standard language for database querying, SQL[28].

A distinct approach was employed in the Lisp programming language, developed between 1956 and 1962 (McCarthy, 1979). This approach, known as the **functional paradigm**, is entirely centered on the manipulation of functions, seen as expressions which receive parameters and manipulate them, producing a result. This was in contrast with the approach used in Fortran, which basically was a set of commands for manipulating stored values, rather than for organization of code itself – those would later be introduced to imperative languages by the Algol branch, mentioned in the previous paragraph. In a functional language, functions can be passed as parameters to other functions, and in fact even values are seen as the results of functions that provide always the same value: the entire language is composed by simple operators and a notations for defining functions in terms of other functions and the basic operators (McCarthy, 1979). Many technical contributions originated in functional languages and then were incorporated into imperative languages, such as conditional execution[29] (Graham, 2002).

Among several other languages based on the functional paradigm of Lisp, one finds Logo, with a long history of educational programming use, which is described on pp. 139-146. Other famous languages that follow this paradigm are ML[30], Erlang[31], and Haskell[32]. Yet, possibly the most widely used functional languages are those of a specific type (one might almost call it a subparadigm), called **data-flow** (Johnston *et al.*, 2004, p. 9). These languages allow the programmer to conceive the program as a graph, with data flowing between nodes, for processing. The programmer defines those processing "nodes" for data, the actual processing methods and syntaxes varying (inside a node, a pure data-flow language would only employ primitive instructions such as arithmetic or comparison operations, but actual data-flow languages frequently employ techniques from other programming paradigms[33]). It is also the programmer's task to define the connections between the various nodes, specifying how the data flows from one node to the next; at each node, the computation waits until all data sources are available, and then its processing takes place. Possibly the earliest such language was TDFL, from 1975 (*id.*, p. 10), but their popularity is due to the various data-flow visual programming languages developed since the 1980s. Examples of such languages include LabVIEW (*vd.* Figure 74, on p. 125), its children-oriented derivative RoboLab (p. 162), and the language used in the MindRover game (p. 188); but their widespread use is due to the fact that this is also the paradigm employed in the programming of business spreadsheets[34] (Kay, 1984), such as Microsoft Excel (Microsoft, 2004a).

A radical departure from both the imperative and the functional paradigms is to program without specifying any manipulation of data at all, neither by direct orders, nor by definition of functions. Such a departure was originated by the Prolog language, created by Philippe Roussel and

[28] Hoffman, 2001.
[29] Determining a condition and from it determine whether a set of instructions should be executed – or which set, if several sets are available.
[30] Harper, 2005.
[31] Armstrong, 1997.
[32] Hudak & Fasel, 1992.
[33] "*Most dataflow architecture efforts being pursued today are a form of hybrid, although not all* (…). (…) *it seems that dataflow languages are essentially functional languages with an imperative syntax*" (Johnston *et al.*, 2004, pp. 2, 10).
[34] Even disregarding the fact that the data-processing instructions in spreadsheet cells are textual, one might still consider these languages only halfway visual: the editing of nodes is visual, and sometimes their selection too, but the arcs of the dataflow graph are not represented, being defined by way of textual references in the nodes' code.

Alain Colmerauer between 1970 and 1972 (Colmerauer & Roussel, 1996): it is called the **logic paradigm**. Languages under this paradigm allow the programmer to specify logic statements, asserting the validity of relationships between the data. Such a program can be read as a set of declarations, but it can also be executed. The execution is not specified by the programmer, but rather imposed by a pre-defined resolution model: for instance, from a given state the system can choose one from a variety of other states for which all the logic statements are still valid (Shapiro, 1989). This approach to programming has been described in more detail in section 3.2.2, in the context of a specific variety of logic programming languages relevant to this thesis (concurrent constraint programming languages).

An important concept that is found across languages classified under any of the previous paradigms is that of **object-orientation** (its presence across languages following diverse paradigms might allow its classification as a meta-paradigm). The basic concept is that programming is done focused on the behavior of *objects* communicating with each other, rather then a set of instructions or functions doing transformations on data. These *objects* are "*collections of operations that share a state*" (Wegner, 1990, p. 8), those operations determining how objects communicate with each other (*id.*, *ibid.*) – another, more usual way of putting it, albeit lacking some generality, is to say that an object is a set of both some data and all the procedures necessary for its manipulation. The first object-oriented language was Simula, developed between 1961 and 1964 at the Norwegian Computing Center, in Oslo, Norway (Nygaard & Dahl, 1978). It was an inspiration for the successful Smalltalk language, developed between 1971 and 1983 by Alan Kay (Kay, 1993b), from which the child-programming language Squeak evolved (*vd.* p. 146).

Object-orientation provides several advantages to the development and maintenance of large programs. Since an object includes both a group of data and the procedures for its manipulation, it's easier to make sure that the manipulation is consistent, and at least partly independent from the object's implementation details: the other objects only depend on the existence of the manipulation procedures to communicate with an object, they can, to some degree, ignore the actual way in which the internal data is stored, its internal structure, or even if it is actually stored somewhere or calculated when requested. I say "to some degree" because often some of these details carry with them specific assumptions that must be taken into account in the overall planning of a program.

Several languages under the imperative and functional paradigms have been adapted or developed from the ground up to incorporate object-oriented features, such as the popular evolutions of the C language, such as C++[35], Java[36], and C#[37]. Object-orientation has also been incorporated in logic programming languages, such as Concurrent Prolog and Vulcan (Kahn *et al.*, 1986).

[35] Stroustrup, 1993.
[36] Chandra & Chandra, 2005.
[37] *Id.*

2.3. Computer programming in our society

2.3.1. Pervasive programming

From the previous sections, computer programming surfaces as it is more commonly seen: a highly technical activity profoundly linked with the computing machines and their technical use. But that is because the history of programming presented in section 2.2 provides just an historical view of the technical evolution. As programming evolved, so did its role, impact and importance in society.

Nowadays, programming is no longer done only by highly-skilled people that understand the machines' internal behavior and organization. In fact, the evolution of computing machines and programming languages and methods did increase the internal complexity of computer programming – just as a person's speech usually increases in complexity with age, as education and experience increase. But it also allowed the creation of simplified forms of programming – programming that can now be done with little regard for the technical specifications and circuit features of the underlying machines.

> *« Setting an alarm clock is probably the most common form of programming. »*
>
> *(Shneidermann, 2001)*

Shneidermann's remark above is a good example: in a traditional mechanical alarm clock, one would rotate a small pointer (the "alarm" pointer) to the position where the pointer for the hours would eventually be when the alarm was to go off. Notice that this would only be the precise time when one wished an exact hour, such as 7:00 or 8:00. For something like 7:15, one would place the pointer more-or-less ¼ of the way between the markings for 7:00 and 8:00, because that's were the pointer for the hours would be at that time. In other words: the setting of the clock had to take in mind its internal operation. If one now considers a modern digital alarm clock, there is no such concern. The only required operation is the setting of the exact alarm time, in minutes and seconds, and placing the alarm switch at the "buzzer" position: the user doesn't have to worry about the microchips' timings, voltages, the signaling used in digital electronics, etc. (One now has to worry about power failures, though.) So, the digital alarm clock is conceptually simpler. However, for many people it's more obvious to simply turn a wheel until a pointer reaches a position than to coordinate three buttons: depressing the "Alarm" button and keeping it depressed, while pressing the "Hours" and "Minutes" buttons, sometimes understanding that keeping these depressed may result in faster change of values after a short while, etc. And that, after all this, a specific lever must be set in a "ring" position, typically labeled "radio" or "buzzer".

In this example there are moments of greater simplicity (setting the exact time, not caring about the internal workings), but also moments of greater complexity, due to the increased ability to specify a precise time and some automated behaviors. This is quite common: the simplicity – or freedom – regarding machine-related concerns in turn allows programming to develop as a mental construction, which has its own complexities.

The alarm clock is far from being the only programmable device in current usage: people program a VCR to record a specific channel, at a specific time, for a specific duration; microwave ovens and washing machines can have specific sets of behaviors (often called "programs") available for the user to employ, requiring an understanding of their behavior over time (although these are not programmed by the user, which only needs to be aware of how the program sequence will evolve), and some other parameters on these machines (such as power or water temperature) usually affect the overall results. Temperature controls, in particular, are commonly programmed in the guise of thermostats, in washing machines and refrigerators, for instance. Even bread toasters require the selection of timing controls to adjust their mode of operation. One can even go beyond technological devices to find other human activities similar to programming. For instance, a football

coach plans specific tactics for free kicks and corner kicks, and explains them to the players. Each player is expected to act in a predetermined fashion and react according to predefined rules. The activity of the coach while planning the tactic can be seen as "programming" it; explaining it to the players and practicing it can be seen as "coding" the program; and the use of the tactic in actual gameplay is akin to seeing a program in execution.

Simple as they are, these examples prompt us to understand that programming permeates society in several, almost camouflaged forms. But the most powerful forms of programming occur in modern general-purpose computers, and – as was put forward in section 2.1 – this specific area is the one approached by this thesis.

> *«* (…) *all computer users may now be regarded as programmers, whose tools differ only in their usability for that purpose. »*
>
> *«* (...) *almost all major software applications now include programming languages. »*
>
> *(Blackwell, 2002)*

It's the amount of programming that takes place in everyday life that often eludes users. Generally, computer users are divided into two groups: **end-users**, people whose main concern is the use of some application, without resorting to programming (or at least without being aware of their use of programming techniques), and **professional programmers**, whose main professional concern is the development of computer applications for themselves or other people (Blackwell, 2002). However, this typical division excludes many people that knowingly employ various forms of computer programming, some more advanced than others, and yet whose main job is not necessarily the creation of computer programs: people that use it in the course of their professions, people that employ it as a means of expression, or simply as a hobby. For this reason, the varieties of programming and programming techniques employed by end-users are presented in the following section, and then section 2.3.3 is devoted to the presentation of some cases of usage of those advanced forms of computer programming in modern society by people that are not typically seen as professional programmers.

2.3.2. End-user programming

> *« End-user programming will be increasingly important in the future. No matter how successful interface designers are, systems will still need to be customized to the needs of particular users. »*
>
> *(Bryson et al., 1996)*

End-users, as defined in the previous section, usually do many tasks which can be considered – and often are – as programming: "*providing programmability is important to support end user customization. Software supplied by vendors rarely does exactly what the customer requires, so providing features that allow the typical user to change the software is quite desirable*" (Myers, 1993).

Examples of such use are numerous. Allen Cypher (1993) sorts them in four categories: Preferences, Scripting Languages, Macro Recorders and Programming by Demonstration. But I find this list of categories somewhat limiting, so I'll first present several examples that don't quite fit in them.

Perhaps the most striking form of programming (used both by end-users and professionals), in its widespread use, is represented by **spreadsheet programs** such as Microsoft Excel (Microsoft, 2004a). Spreadsheet users regularly create formulas, which often depend on the results of other formulas, and thus end up producing a specific application, to suit a specific purpose (Jones *et al.*, 2003).

Creating **Web pages** can also incorporate some degree of programming. I am not referring to the most obvious case, when the page builder employs some programming language to define interactive behavior; rather, I am considering the fact that a Web page will look different to different viewers, or even to the same viewer in different instances (Blackwell, 2002). Several situations can cause these changes, but the most obvious are the different sizes in the dimensions (width and height) of the browser Window. In my personal experience working with professional graphic designers, this fact was something that troubled them: the need to understand and consider the dynamic behavior of a Web page, while still doing just graphic design, not HTML coding (Bulas-Cruz *et al.*, 1998; Bulas-Cruz *et al.*, 1999).

Even on everyday activities such as word processing, programming can be found. Beyond the most basic text-entry and text-formatting skills, one of the most helpful techniques people use in word processing applications is the **definition of paragraph and text styles** (Daffron, 1999). By saying that a selected paragraph should be considered, for instance "My Heading Style", one can later, at a different paragraph, simply select "My Heading Style" from the list of styles and immediately have the new paragraph follow the graphic style of the original paragraph. Programming is relevant because styles don't limit themselves to acting as shortcuts for application of graphic formatting: one can include rules such as "style for the following paragraph is". This can immensely speed up, for instance, the process of entering a bulleted list like this:

▪ *First element (away from the previous text)*	Style "Bullet list first line"
▪ *Middle element 1 (near other lines)*	Style "Bullet list body"
▪ *…*	Style "Bullet list body"
▪ *Middle element n*	Style "Bullet list body"
▪ *Last element (away from the following text)*	Style "Bullet list last line"

Table 3 – User programming with word processing styles

The user simply selects "Bullet list first line" as the desired style and starts typing. At the end of the line, by pressing Enter, the user can proceed and type the text for the second bullet; however, the style was automatically changed to "Bullet list body", which has no distance between paragraphs. After typing the text for all middle bullets, the user simply selects the style "Bullet list last line" and writes the last line, which has some distance to the following, plain text paragraph. By

pressing Enter, the user can simply continue typing away, since the text style was automatically changed to "plain text" or "body text", or whatever style the user decided would be more common for that situation. (Myers, 1993, p. 499, makes a passing reference to styles in Microsoft Word as a programming technique.)

Another important technique in word processing applications is the use of **regular expressions for searching and replacing**. Consider this situation: one wants to replace, say, all instances similar to "auto-correction", "auto-summarizing", *et cetera*, by "automatic correction", "automatic summarizing" and so on. In a large document, this can be a time-consuming task. But experienced users know they can do this in one single command, by entering specific expressions in the search and replace boxes of a dialog. For instance, in Microsoft Word (Microsoft, 2004b) this replacement can be done by using this set-up:

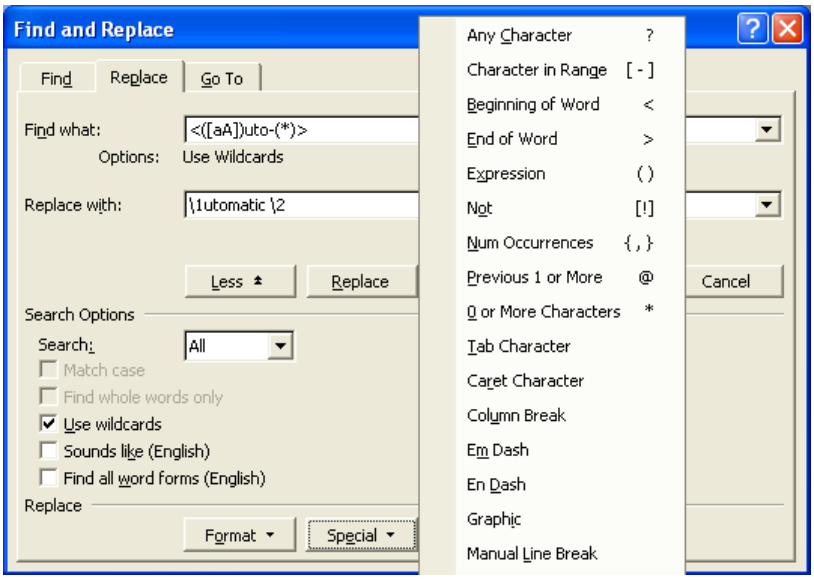


Figure 22 – Finding and replacing with regular expressions

The expressions in Figure 22 mean:

> ***Find what:*** *beginning of word; either letter "a" or letter "A"; letters "uto-"; any number of letters; end of word.*
>
> ***Replace with:*** *either "a" or "A", whatever was found; the letters "utomatic" followed by a space and whatever was found after "uto-".*

Yet another case can be found in e-mail filtering systems, either against junk mail (a.k.a. "spam") or for automatically organizing one's messages as they arrive; these systems need to be tailored by the user, specifying rules and behaviors (see Figure 23).

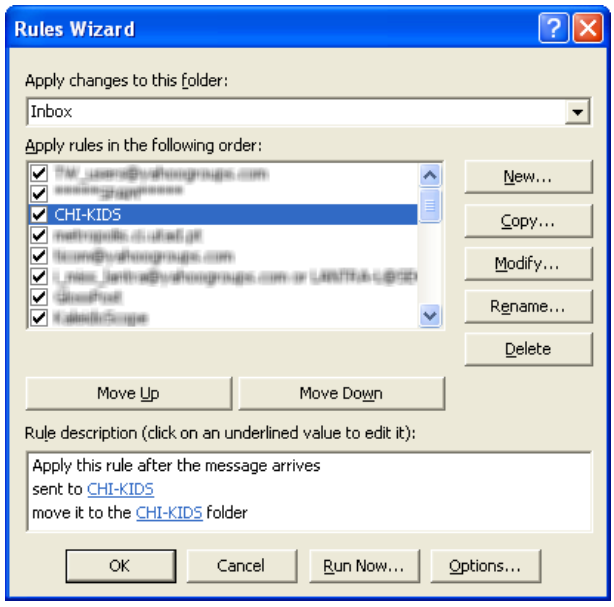


Figure 23 – Defining e-mail rules

Considering now Cypher's categories, I'll start with the creation of **macros**, which is available in programs such as the word processor Microsoft Word (Microsoft, 2004b). A macro is a sequence of commands recorded by the user as actions. Rather than learning some programming language, the user will simply select a menu or button command to "Start Macro Recording", and then act as intended. As Cypher (1993) puts it, « *these recorders are a basic implementation of "Watch What I Do". (…) All the user's actions*

are saved as a sequence, and the user can then invoke a "Redo" command to replay the entire sequence. »

The complexity of macro use arises that it is highly dependent from the context conditions. For instance, suppose you want to delete a long list of ">•" from an e-mail message, such as this one (I used "•" to indicate a space, to render this explanation simpler):

>•*Hello,*
>•*Is everything fine with you?*
...

If this message was very long, or if you needed to do this in several messages, you could record the following actions, starting with the cursor as represented below by the vertical bar:

1. Start Macro Recording

   |>•*Hello,*

2. Click the "Delete" key twice

   |*Hello,*

3. Press the "↓" key.

   Hello,
   |>•*Is everything fine with you?*

Typically, the user would now stop recording and assign this macro to a key such as "F8" or to a toolbar button labeled "Remove >". While time-consuming, it's easier to click consecutively on F8 or on a single button than it is to be moving the mouse and using a few keys for each line. (With a little bit of actual programming language use, even this much trouble could be avoided.)

Now, suppose that for some reason the original message had more than a single space at the start of each line:

>•*Hello,*
>••*Is everything fine with you?*
>••*I went to the game last night.*
...

Using the previous macro in this message would result in:

Hello,
•*Is everything fine with you?*
•*I went to the game last night.*
...

This happens because the recorded action was "press Delete twice", and it was only by coincidence that deleting two chars in the first message was the action that removed the prefixes.

An experienced user would have rather used "*press-and-hold Ctrl and press →. Then press Delete*". This would select all spaces from ">" to the beginning of the first word, therefore being applicable to larger number of situations. While this is usually called "advanced use", or something for this effect, such care in specification of actions, considering possible future situations, is a programming skill and activity.

Another of Cypher's categories, **preferences**, is extremely common: most applications have some kind of "options" or "preferences" section where users can customize the operation of the application. However, these often include a certain degree of programming. One such example can be found in modern word processing packages, where users can set a large number of options for automated processing of text. For instance, a user can set "(c)" to display as "©", besides many

other settings that will have an impact – sometimes not so obvious – in the overall functioning of the application (see Figure 24).

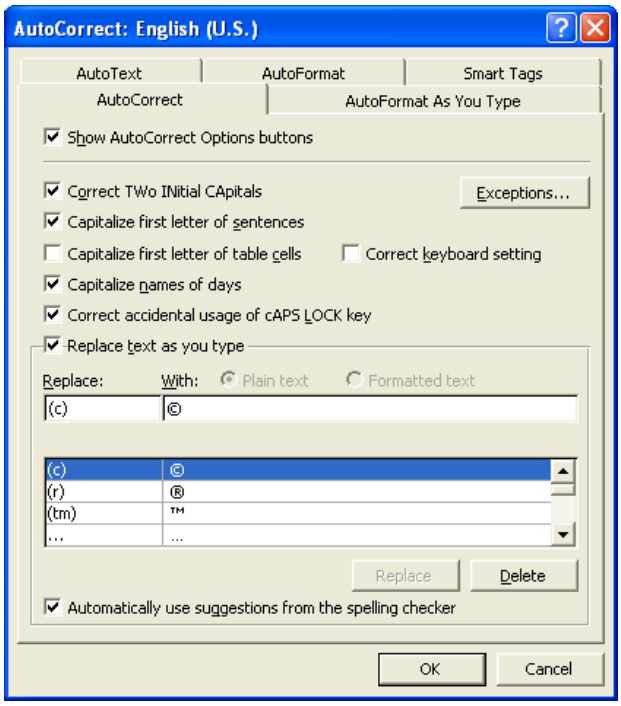


Figure 24 – Word processor preferences (automatic text)

Yet another relevant example of preferences is date formatting. In several applications, dates are stored in an internal numerical format, and the user employs "formatting strings" to specify the way in which a specific date is displayed or printed.

Examples include common spreadsheets, but also other programs such as the Scribe text formatter. In these programs, when "*specifying the way the date should be printed, the user can supply an example in nearly any notation for the particular date March 8, 1952*" (Myers, 1993).

The remaining two of Cypher's categories, Programming by Demonstration and Scripting Languages, are used both by end-users and by professional programmers.

One of those, however, is more common in the realm of end-user programming: **Programming by Demonstration**. This has been called "*macros on steroids*" (Lieberman, 2001, p. 2). Macros, as I described above, are very dependent on starting conditions, and can only reproduce the exact same behavior. However, as was then mentioned, when defining a macro the user is in effect saying to the computer "watch what I do". Programming by demonstration picks up from this basic macro concept and adds the possibility of generalizing the actions of the user, rendering them useful for a wider range of applications (*id.*, *ibid.*).

Sometimes, this is done by empowering the user with a way to specify what information he/she wants to generalize. This, in fact, is the approach used in ToonTalk, the programming environment employed in the field work of this thesis (*vd.* section 3.3.5), and also in Stagecast Creator, another programming environment for children (*vd.* section 3.3.4, pp. 151-155). Another promising approach employs techniques from artificial intelligence, attempting to achieve some level of automatic generalization of the user's intentions. This approach hasn't yet provided major commercial results, but there's an active community of researchers coming up with promising prototypes and small systems (Cypher, 1993; Lieberman, 2001).

The final category, **scripting languages**, is even more spread-eagled between end-users, advanced programmers, and professional programmers. Traditionally, these would be languages that allowed users to automate some common tasks done with an application, *i.e.*, provide a script of actions for the computer to repeat – in other words, more or less like writing down the steps of a macro, instead of using a "recording" procedure. However, in order to avoid the limitations of

macros, scripting languages typically provide several features that allow the person using them to enlarge their range of uses, such as conditions, parameters, and cycles. For instance, in an operating system text shell, such as the Unix shell or the command line prompt of Windows, an user can write down in a text file a sequence of commands for the system to interpret and reproduce; but the user can also add specific symbols and modifiers, so that when that text script is called up for execution, the user can include parameters, or have it conduct different operations under different conditions (*e.g.*, process the contents of different text files, as long as they follow a common format).

From the point of view of a common user, such a scripting language has virtually the same level of complexity as a full-fledged programming language. But from the point of view of an experienced programmer, there is a clear simplification: the scripting language is focused on changes to a specific environment, and possesses specific features or "hooks" to employ to that environment, simplifying the entire process of programming. But the distinction is fuzzy: for instance, the immensely popular game series Quake (Id Software, n.d.), where the player goes around three-dimensional worlds shooting monsters or other players, includes a scripting language, so that players can create new weapons and behaviors, effectively transforming the game. But the scripting language of Quake is an adapted version of the professional C programming language, called QuakeC (Hesprich, 1998) – not exactly what one would call a choice for typical end-users. Possibly a relevant contribution to this view is that anecdotal reports surface from time to time in Web forums, where players of various games with scripting capabilities mention having been introduced to advanced computer programming precisely via their interest in modifying a game by using its scripting language. In the scope of this thesis, such scripting languages are indeed considered entry-points or simplifications to advanced programming.

One might say that scripting languages are not doing a spread-eagle between end-users and advanced programmers, but rather being split between them in usage style: end-users using them simply to write down scripts that work more or less like macros, advanced programmers writing scripts using traditional programming techniques. This approach might yield an effective distinction in the case of early textual or symbolic scripting languages. But as user interfaces developed, the range of possibilities rendered available to users has similarly increased, allowing them to specify interactive automated behavior in several ways. This evolution led to the appearance of a range of applications that further blur this transition area between end-users and advanced programmers: **multimedia-authoring applications**.

Such applications allow end-users to create various interactive mini-applications, such as DVD menus, or interactive slideshows. Since the user is specifying the automation of behaviors in the scope of a specific application, I elected to classify these as fitting within the range of scripting environments. Taking Microsoft's PowerPoint (Microsoft, 2005) as an example, an user can create a slideshow presentation and specify if a slide should stay visible until the user clicks the mouse, or rather be replaced after a certain time has elapsed. And in this latter case, whether the mouse-clicks can be used to anticipate the replacement, or ignored. Further, the user can specify that an object in the presentation reacts to user clicks, by moving, disappearing, blinking, making a sound, or other possibilities (those actions can also occur when the mouse simply passes over the object). And these mouse events can work as hyperlinks, moving the slideshow to a specific slide, in effect allowing the creation of something that is more than just a slideshow.

Unlike other scripting languages, these capabilities are extremely simple to grasp and use, particularly so in view of the large range of results one can produce. For instance, since 2002 I have lectured an optional one-semester course, for first-year students of the Early Childhood Education baccalaureate, at the University of Trás-os-Montes and Alto Douro (Vila Real, Portugal); students without a technical background learn to take advantage of these capabilities. They have employed them in the creation of children tales where children can determine the outcome by selecting different options during the story; "explorable" stories, in which children can click on various objects or characters to get a detailed look, before proceeding with the story; simple interactive games, and even multimedia "zoos" and "farms".

2.3.3. Advanced programming

As mentioned in section 2.3.1, many people knowingly employ computer programming, some in a more advanced way than others, even though their main job description probably isn't as "programmers": people that use it in the course of their professions, people that employ it as a means of expression, or simply as a hobby. These people use specialized applications such as full-fledged programming languages, scripting languages or other advanced programming methods. The most obvious case of advanced programming, of course, is the programming that is done to create commercial computer applications, for use by others – those people whose job description reads "programmer".

A full survey of these uses is beyond the scope of this thesis, as is the exploration of the actual job of "programmer". This section simply aims to provide some examples, to assist the reader in constructing a finer picture of the use of programming in modern society, particularly where it is least expected; for instance, "*some industrial robots are trained by moving the robot's limbs through the desired motions. The robots will repeat these motions later*" (Myers, 1993). This is an example of macro programming[38] (a technique explained in the previous section) used in a professional manner, where the complexity and care that the end result imposes cannot be disregarded.

Taking the cue from this industrial example, I look into the field of computer-aided design (CAD) tools, which are commonly used by mechanical and civil engineers, but also by architects and other professional designers that need to produce precise drawings efficiently. Rather than simply use visual drawing techniques, CAD tools employed by these professionals routinely involve issuing textual commands, providing textual or numerical parameters, to control the behavior of the program (Figure 25). These professionals are already invoking commands from a library and providing them with parameters, not just numerical distances and angles but also operational modes; but they can also produce text files with several commands in sequence, to expedite common design tasks – a case of programming by scripting.

Command Sequence

Command: `SNAP`

```
Specify snap spacing or
[ON/OFF/Aspect/Rotate/Style/Type] <10.0000>:
```

(…)

The "Aspect" option can be used to vary the horizontal and vertical snap spacings independently.

"Rotate" is used to set the snap grid to any angle.

You can also set the snap style to either Isometric or Standard (the default) using the "Style" option. The Standard style is used for almost all drawing situations including detail drawings in Orthographic Projection. The Isometric style is specifically to aid the creation of drawings in Isometric Projection (…).

The "Type" option allows you to set the snap type to either Grid (the default) or to Polar. The Polar option can be used in conjunction with Polar Tracking so that Snap mode snaps along polar tracking angles rather than to the grid.

Figure 25 – AutoCAD manual entry
From: http://www.cadtutor.net/acad/acad2ki/aids/aids.html

At a more advanced level of graphics programming, Web graphic designers commonly need to program animated behaviors and user interactions, to make a Web site respond to the user's mouse movements and selections. Several tools allow them to do this, the most common being Macromedia's Flash (Macromedia, 2004a). These tools include components

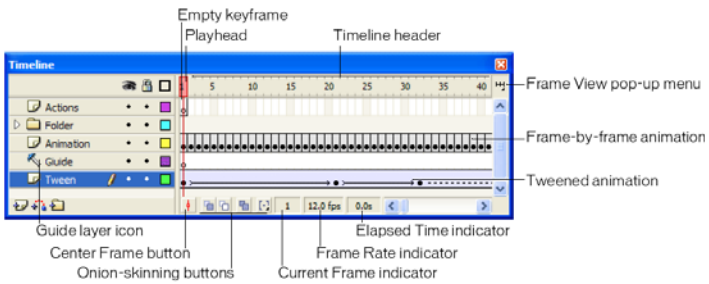


Figure 26 – Macromedia Flash timeline control
From: Macromedia, 2004, p. 24

[38] In this case, this classification is fuzzy; rather than reproducing the operator's movements precisely, the robots can use sensors to fine-tune of the original motions. Thus, one can state that the robot generalized of the original instructions, and this would be a case of programming by demonstration, rather than macro programming.

typical of end-user multimedia-authoring programs, such as timeline controls (Figure 26). However, for full control of the animations and user interactions, Web graphic designers need to use textual scripting languages: for instance, in Macromedia Flash, the scripting language is called ActionScript (Macromedia, 2004b), as shown in Figure 27.

To draw a shape:

1. Use `MovieClip.createEmptyMovieClip()` to create an empty movie clip on the Stage. The new movie clip is a child of an existing movie clip or of the main Timeline, as shown in the following example:
   ```
   this.createEmptyMovieClip ("triangle_mc", 1);
   ```
2. Use the empty movie clip to call drawing methods. The following example draws a triangle with 5-point magenta lines and no fill:
   ```
   with (triangle_mc) {
        lineStyle (5, 0xff00ff, 100);
        moveTo (200, 200);
        lineTo (300, 300);
        lineTo (100, 300);
        lineTo (200, 200);
   }
   ```

Figure 27 – Programming with ActionScript
From: Macromedia, 2004b, p. 217

Still in the area of graphics, many modern movies routinely employ computer-programming techniques as part of their development. For instance, in Disney's 1985 movie The Black Cauldron, "*computers made inroads in the manipulation of solid inanimate objects on the screen. The dimensions and volume of objects were fed into a computer and their movement was generated by programming*" (Magical Ears, n.d.). The professional process of creating computer animations for movies frequently involves a combination of use of specialized end-user animation tools and programming, to determine the specific rendering of visual features, such as surface textures under different lighting, blurring to improve the illusion of motion, and so on. Further, behaviors and movements for characters need to be modeled, and while this can be achieved by several traditional techniques, involving multiple images to convey the idea of motion, the use of procedural techniques is attaining large popularity.

> *« Current technology is not capable of generating motion automatically for arbitrary objects; nevertheless, algorithms for specific types of motion can be built. These techniques are called procedural methods because a computer follows the steps in an algorithm to generate the motion. Procedural methods have two main advantages over keyframing techniques: they make it easy to generate a family of similar motions, and they can be used for systems that would be too complex to animate by hand, such as particle systems or flexible surfaces. »*
>
> *(Hodgins et al., 1999, p. 6)*

A particular use of these techniques is the control of vast numbers of characters, where each computer-generated figure must display a singular, individualized behavior. One often-cited example are the crowd scenes in Walt Disney's "The Hunchback of Notre Dame" (*e.g.*, Hodgins *et al.*, 1999, p. 7), or more recently the huge armies in the three "The Lord of The Rings" movies of New Line Cinema (*vd.* Figure 28).

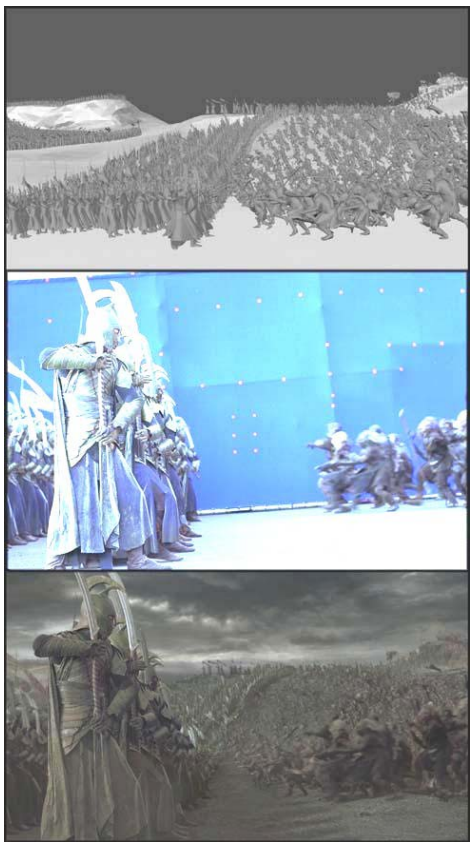


Figure 28 – Computer-generated characters combined with live images
From: http://www.warofthering.net/ahobbitstale/extras/extras_sfx1.jpg

In this last case, in particular, the large numbers of computer-generated figures in the armies were programmed not only to display individual behavior, but also to respond to the behavior of other nearby figures – to such an extent that sometimes complex group behaviors emerged:

> *« (…) we put a level of machine intelligence into each of the agents so they can act*

autonomously based on information they get -- sight and sound from the environment around them. We had a lot of issues with directing them. It turns out, when you create reasonably intelligent creatures, they do what they want to do. »

(Labrie, 2002, Chief Technical Officer of the company developing the special effects for the three movies)

Another example is the field of domotics, a.k.a. "house automation", using computer-controlled systems. When installing such systems, professional installers need to program the computer controller with several standard behaviors[39] according to the wishes of each household member, usually based on various parameters such as timers and sensors; the customer can also program several configurations himself or herself, *e.g.*: alarms to be issued over an Internet or cellphone connection, programming hours for automated turning on or off of different behaviors.

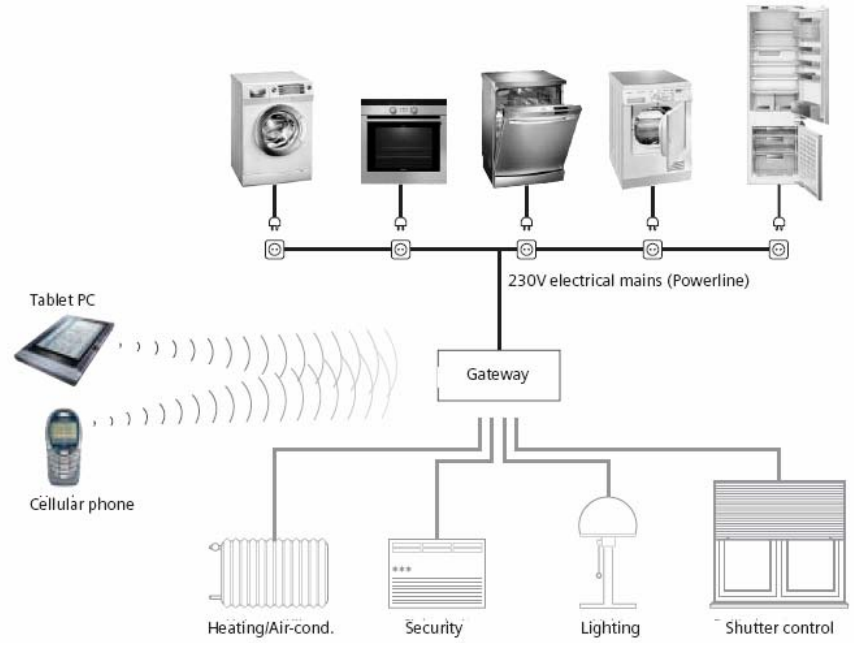


Figure 29 – Structure of a typical domotics system

From: http://www.servehome.com/pdf/serve-home-prospekt-i2-en.pdf, p. 7

These can all be achieved using automatic configuration mechanisms, or pre-programmed components, but installers (and users) also perform the programming themselves to fine-tune what is usually an expensive system. For instance, in the documentation of one of the available inter-operation standards for domotics mechanisms, one finds the following:

« The KNX Standard incorporates 3 different configuration modes :

The "S-mode" (System mode)

This configuration mechanism is meant for well trained installers to realise sophisticated building control functions. All "S-mode" components in an installation will be addressed by the common software tool (…) *for their planning, configuration and linking. With ETS each component can exactly be programmed, according to the specified requirements. The "S-mode" configuration has the highest degree of flexibility in functionality and in communication links.* (…) *»*

(Crijns, 2003)

[39] Typical examples are: electrical appliance control (pre-heat oven on certain conditions, start dishwashing cycle at specific hours), lighting control (on/off, intensity and mood), windows blinds and curtains, heating or cooling levels and profiles, automated hi-fi or home cinema, and operation of garden sprinklers (Smarthome, 2005).

Artists are also exploring novel approaches to the creation of art, by integrating computer programming – and even robotics – in their work. In particular, the field known as algorithmic art (Verostko, 2004) has long taken advantage of the possibilities brought about by computer programming in the production of art, its practitioners known as "algorists".

> *« As computers became more accessible to artists in the 1970's and 1980's some artists began to experiment with algorithmic procedure. The new technology offered them methods of working algorithmically that were unavailable before the advent of computers. By the 1980's a number of these artists were working with the pen plotter, a machine with a "drawing arm". By the end of the 1980's algorists like Harold Cohen, Mark Wilson, Manfred Mohr, Jean Pierre Hebert and this author had already achieved a mature body of work. Each in their own way had invented algorithmic procedures for generating their art. By doing so each created their own distinctive style. Clearly style and algorithm were linked in a very important way. »*
>
> *(Verostko, 2004)*

At this level, the specific applications of programming are diversified. For instance, Some artists use their algorithms to have computer plotters draw the result of their algorithms (Figure 30), sometimes replacing the typical print heads of plotters by brushes or other artistic tools (Figure 31).

Figure 30 – Roman Verostko, "Cyberflower V. 1, 2000", pen plotted drawing

From: http://verostko.com/shows/tempe/cyber-5-1-w.jpg

Figure 31 – Plotter with an oriental brush installed

From: http://verostko.com/pathway/brush11.jpg

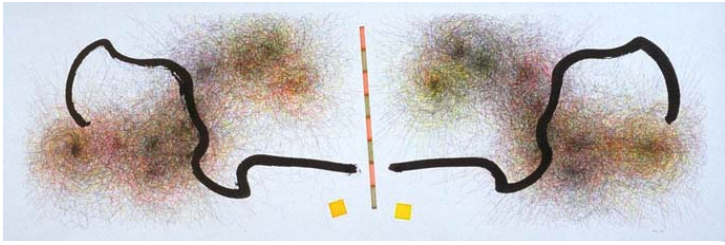


Figure 32 – Roman Verostko, "*Lung Shan II*", pen+brush plotted

From: http://verostko.com/pathway/brush11.jpg

Among other varieties, artists have explored the use of algorithms to render classical solid surfaces, fractal surfaces and landscapes; they have developed genetic pictures, "*obtained through a succession of mutations of an initial image*" (Vassallo, n.d.), and generated abstract scenes by combining algorithms and 3-D rendering programs (*id.*); an example of this last variety is presented on the right, in Figure 33.

Similar efforts have been done in the fields of music (Harry, 1995), and plastic arts, in this latter case both at the level of planning and of behaviors (*e.g.* DAM, n.d.).

Figure 33 – Robert H. Russ, "Whisper", 3-D rendered algorithm

From: http://www.sdsc.edu/~russ/art/whisper.jpg

One recent variety of this artistic exploration has been Leonel Moura's programming of robots to produce art, an approach entitled "Symbiotic Art" (Moura & Pereira, 2004). The robots and the evolution of a painting are shown in Figure 34.

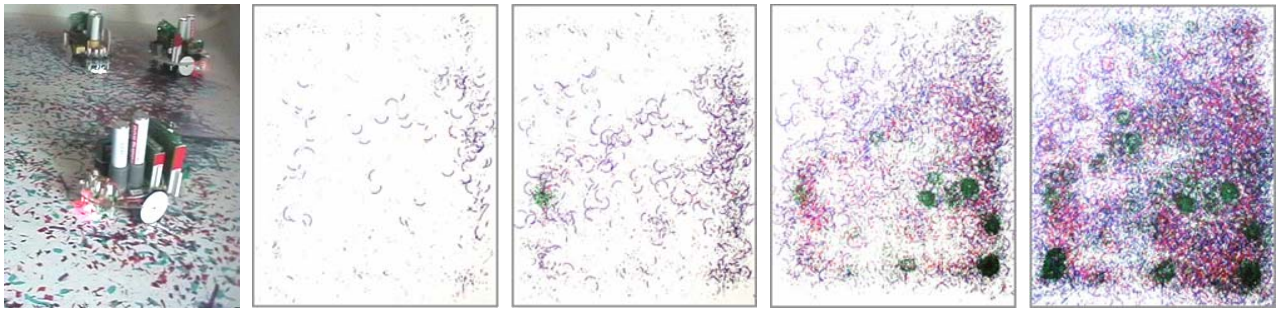


Figure 34 – Robots at work and painting in progress (30, 60, 120 and 240 minutes)
From: http://www.lxxl.pt/artsbot/atwork.jpg and http://www.lxxl.pt/artsbot/image005.jpg

Conversely, some people have started to inquire on the artistic potential of programming itself[40], as a novel form of human endeavor, both from the perspective of an elaborate craft (Fishwick, 2000; Hunt, 2001), and from that of an artistic form in its own right:

> *« (…) some programmers have adopted a literary approach to coding, describing carefully crafted code as "beautiful," "elegant," "expressive," and "poetic"; writing and reading programs as literary texts; and even producing hybrid artifacts that are at once poems and programs. »*
>
> *(Black, 2002, p. v)*

Returning to non-artistic examples, in various professional fields there are occasions when some time-consuming task lends itself adequate for the use of small customized programs, or there is an intention to provide computer-based support for its execution. Sometimes, however, people that are not professional programmers take up the task of programming. The circumstances leading to this can be varied, but to point just two: there may be no budget to hire the services of a professional programmer or software-development company; or special subtleties of the data to be processed may cause professional programmers unacquainted with it to misunderstand specifications, leading to the failure of the development project, and such a frustrating experience may cause the people involved to decide and program the software themselves.

Figure 35 – EuroPreArt sample data-entry form and Web results
From: http://www.europreart.net/eurodb1.gif and http://www.europreart.net/preart.htm

In cases such as these, professionals from various areas besides programming end up dusting off programming skills learned in undergraduate course or training sessions (or even deciding to outright learn programming from scratch), in order to produce themselves the necessary software. For instance, I once came across a program for management of a surgery waiting list, developed by the surgeon himself; and the development of a large on-line European Prehistoric Art Database, presented in Figure 35, has been conducted by one of its participant archaeologists (Arcà, 2002).

[40] As a final comment under this arts theme, it's interesting to note that Logo, the first computer programming language for children (vd. p. 139), owed much of its success to a subset of commands designed to allow children to perform drawings procedurally (vd. p. 144). Also in this thesis, the very first of my proposed usage typologies of computer programming in preschool education involves its use as a space for self-expression (vd. section 7.3.1).

While the programming of complex and demanding applications by non-professionals can be a bad option, likely to produce awkward contraptions suffering from well-known and easily-avoidable programming pitfalls, the use of programming to automate small repetitive tasks as they pose themselves – or even slightly before that, when one manages to anticipate them – can be a very good option, leading to increased productivity, and has been called **just-in-time programming**:

> *« the goal of just-in-time programming is to allow users to profit from their task-time algorithmic insights by programming. Instead of automating with software that was carefully designed and implemented much earlier, the user recognizes an algorithm and then creates the software to take advantage of it just before it is needed, hence implementing it just in time. It is worth emphasizing that the user's task could be from any domain (e.g. graphic drawing, scientific visualization, word processing, etc.) and that the algorithm to be implemented originates with the user. Obviously, a user with more programming experience will be able to envision a more complex algorithm than a novice user. »*
>
> *(Potter, 1993)*

Basically, programming just in time can be a good option when the time and effort invested in the programming diminishes the overall time and effort spent on the entire task (Figure 36). Such a decision is dependent on the programming skills of the individual, on the features of the programming tools available, and on the level of programmatic access to the data (*id.*).

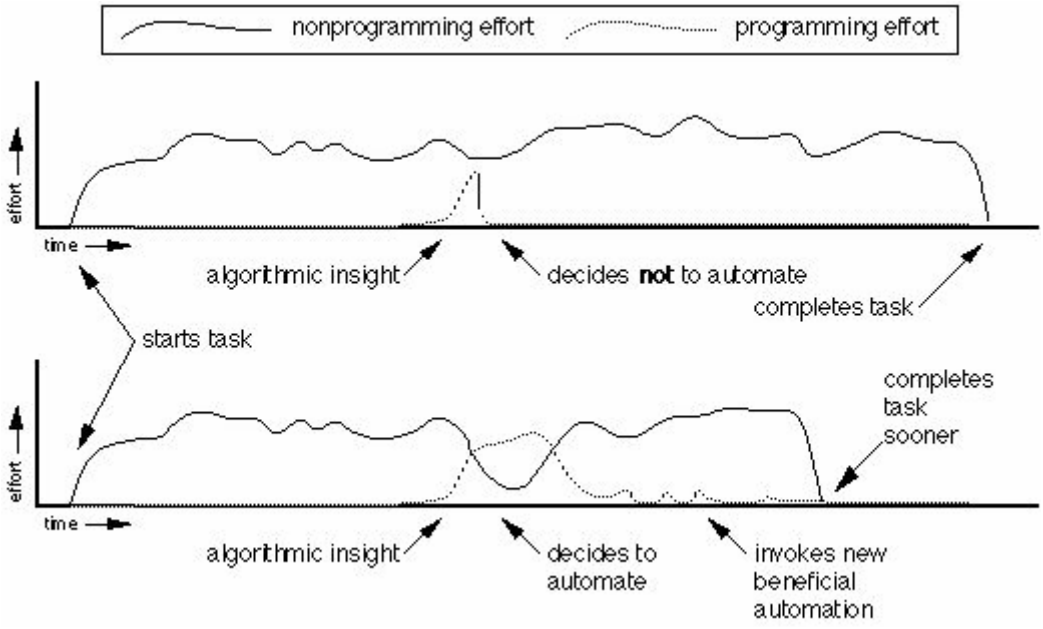


Figure 36 – Comparison of two tasks: with and without just-in-time programming
From: Potter, 1993 (version at http://www.acypher.com/wwid/Chapters/27JITP1.jpg)

To conclude, I'd like to briefly mention a group of professionals who are, possibly, the ones closest to professional programmers in their use of computer programming in their jobs. I am referring to people whose jobs require them to use programming virtually on a daily basis, employing programming skills identical to those of professional programmers: people who constantly analyze novel situations, for which no adequate computational tool is (or can be) available, other than a programming-based tool.

These professionals are found working in fields employing calculus, the creation and analysis of simulations, or the creation and analysis of decision-making scenarios. Such fields include engineering, biotechnology, financial analysis, mathematics, and scientific research, using programming-based tools such as Mathematica (Wolfram Research, n.d.), ChemCAD (Chemstations, n.d.), Simulink (The MathWorks, n.d.-1), and MATLAB (*id.*), among many others.

2.4. Computer programming in education

2.4.1. Introductory remarks on computers & education

Since this thesis focuses on preschool education, I won't be presenting information on the use of computers in high-school-level and college-level education, or professional training (although many remarks are probably valid for all ages). Instead, this section is centered on the use of computers in child education, *i.e.* preschool, elementary and primary education levels. These levels concentrate most of the background research and field reports publicly available, and form the overall historical scientific background. The information for all these levels is not discriminated here, since the specific use of computers and computer programming at the preschool/kindergarten level (3- to 5-year olds) is presented in detail in chapter 4.2, "Computers in preschool".

This said, I would like to remark that when one mentions the use of computers – and computer programming – in education, most listeners are thinking either about new subject matter to teach and learn, computer-related, or about software and hardware tools to help the teaching of traditional subjects (or both). This is to say, listeners usually think about how children can benefit from this technology and use it.

However, this technology impacts far more than just the children's experience: for its potential impact to be fulfilled, one must consider not just the children's point of view, but also the teachers' point of view. The most powerful uses of computers in education involve change, not just on the part of the students, but also for the teachers.

> *«* (…) [underlying theoretical] *issues do not fall solely in one domain but span across the traditional concerns of philosophers, psychologists, educators, and computer scientists. In fact, the new computer educator emerges as one with a foundation in each of these disciplines. »*
>
> *(Solomon 1986, p. 1)*

2.4.2. Framing computer use in education

The title of this section has an intentional double meaning. Here I'll present an overview of how computer have been used – and are still used – in educational settings. However, it should become clear for the reader of this thesis that I personally stand along with the defenders of the immense potential of computer use for the development of education. For this potential to be realized, computers must be used in ways that are not in widespread use today, lest their impact becomes negligible. Among resources on this subject, I point out Papert (1998), Jonassen (2000), and diSessa (2000, ch. 9, "Stepping Back, Looking Forward").

Several authors propose different classifications of the use of computers in education (Berg, 2003, p. 14; Hoić-Božić, 1997, ch. 3.1.1; Jonassen, 2000, pp. 3-9; Solomon, 1986, p. 8-13; Solomon, 1996, pp. 6-8). The encompassing view I present here groups educational computer use in four major models, which will be further explained in the coming sections:

- Learning about computers
- Learning from computers
- Learning with computers
- Learning about thinking, with computers

The first model, **learning about computers**, is the mere inclusion of the computer and its features (hardware, software) as new subject matter for teaching and studying. While knowing about a tool is useful to understand how to use it more efficiently, this approach is just aimed at making children better computer users; no change in learning methods is taking place. In this model, computer programming is learned for its own sake, since it's part of the "computer science" body of knowledge.

The second model, **learning from computers**, represents the kind of educational computer uses that most people think of when the expression "educational software" is used, and it has also been defined as using computers as "*interactive textbooks*" (Solomon, 1986, p. 8-10; Solomon, 1996, pp. 6-8). This model represents the use of information-rich software applications (CD-ROMs, Web sites, etc.) and other information-based multimedia applications. It also includes Computer-Aided Instruction (CAI), Computer-Aided Learning (CAL), and Computer-Based Training (CBT), using rote learning and drill-and-practice applications, tutorials, and intelligent tutoring systems (Hoić-Božić, 1997, ch. 3.1.1; Jonassen, 2000, pp. 4-7). Computer programming is scarcely used in this model (if at all).

The third model, **learning with computers**, is based around using computers as learning tools, just as one would use a sheet of paper, a pencil, paints, dry leaves, a microphone and a tape recorder, whatever. It is associated with the modern shift from teaching by instructionalism (where the teacher is the source providing all knowledge) to constructivist teaching (where the teacher acts as partner and coordinator, facilitating the mental construction and development of knowledge by the students). Another way of putting it is saying that the computer stops being a teacher and starts to become a learning partner (Jonassen, 2000, p. 7). It also includes the use of simulations and content-rich games, for children to explore, being thus introduced to new content in an interesting way (Hoić-Božić, 1997, ch. 3.1.1). In this model, computer programming is one amongst several ways in which computers can be used.

The fourth model, **learning about thinking, with computers** is similar to the third, but has a central concern in helping learners improve their learning processes. Its name may be misleading, for it seems to be less general than the previous models: it deals with "learning about thinking" …with computers. "So," one may ask, "is this model only suited to learn about thinking?" Indeed, that is the case. But as Papert once put it:

> *You can't think about thinking, without thinking about thinking about something.*
>
> *(Seymour Papert in Minsky, 1988, p. 6)*

In other words, if one learns about thinking, one does so by reasoning over one's own thinking about specific subject matters. Those subject matters weren't studied merely for their own sake: in the process of study, the learner gains insights into his/her own mental processes while working in that area. This, in turn, helps create powerful connections and personal approaches to what one is studying. Computer programming is central to this fourth model, since it allows children to teach the computer, and in doing so, by finding and correcting misinterpretations of their teachings ("bugs"), they have to consider their own thinking processes.

> *« Learning from bugs and discussing teaching strategies to enhance what the computer knows help children reflect on their own learning. »*
>
> *(Solomon, 1996)*

2.4.3. Learning from computers

This model of computer use in education is the oldest, both in application and in theoretical background. It is above all an extension to traditional educational methods, rather than a drastic departure. For this reason, this section also presents a brief history of computer use for educational purposes. But limited as it may seem now, this model did include serious innovation from the start and is still the most prevalently used today.

The overall logic of this model can be seen from how the basic ideas of using a machine to improve instruction were presented by Edward Thorndike, in 1912:

> *« If, by a miracle of mechanical ingenuity, a book could be so arranged that only to him who had done what was directed on page one would page two become visible, and so on, much that now requires personal instruction could be managed by print. »*
>
> *(Thorndike, 1912, p. 165, in McNeil, n.d.)*

Figure 37 – Edward L. Thorndike
1874 – 1949
From: http://www.psychology.uiowa.edu/Faculty/wasserman/Glossary/skinner.jpg

Thorndike acknowledged the advantages of teacher-based instruction, which for him were above all the greater empathy and customization made possible by human contact: he considered words to be more enticing than "*black marks seen*" (Thorndike, 1912, p. 161, in McNeil, n.d.), *i.e.*, book-based education. He also saw the following teacher's actions as an advantage of teacher-based education against book-based education: practice activities, questions, explanations, directions, etc. But he applauded the advantages of book-based study, since it "*allows a student to think at his own pace, get the facts over and over again as he needs, test himself point by point as he goes along, and make notes*" (Thorndike, 1912, p. 161, in McNeil, n.d.). However, he saw limitations in the use of books that sound extremely valid today:

> *«* [Textbooks] *commonly give the results of reasoning, and perhaps problems demanding reasoning, but they do not so manage the latter that the pupil is at each stage helped just enough to lead him to help himself as much as is economically possible...nor do they usually give work in deductive thinking so arranged as to stimulate the pupil to make and test inferences himself. »*
>
> *(Thorndike, 1912, pp. 164-165, in McNeil, n.d.)*

Thorndike proposed for books to include data, instructions on how to conduct experiments and work out problems over those data, and questions about inferences that could be extracted. The problem, according to him, was that students didn't follow written instructions and tried to read both the problems and their answers, rather than try to develop their own answers. It was to overcome this that he proposed the creation of a machine to control the pace of study, as quoted above, at the beginning of the current section.

Figure 38 – Pressey Teaching Machines
From: http://www3.uakron.edu/ahap/apparatus/images/photos/presseyteachingmachines(8-8-2).jpg

The first attempt to put Thorndike's ideas to practice occurred ten years later, in the early 1920s. An educational psychology professor, Sidney Pressey (1888 – 1979), developed a testing machine (*vd.* Figure 38), to help students practice and evaluate their knowledge (McNeil, n.d.; McDonald *et al.*, 2005). He wanted teachers to focus on "*inspirational and thought-stimulating activities which are, presumably, the real function of the teacher*" (Pressey, 1926, p. 374, in McNeil, n.d.). His machine "*resembled a typewriter carriage with a window that revealed a question"* (McNeil, n.d.). In order to answer, "(…) *the student refers to a numbered item in a multiple-choice test. He or she presses the button corresponding to the first choice of answer. If that is right, the device moves on to the next item; if it is wrong, the error is tallied, and the student must continue to make choices until the right response is made*" (Skinner, 1968, ch. 3). Pressey continued to develop his teaching machines until 1932, when the Great Depression caused the stalling of developments in educational technology, which only resumed after World War II (McDonald *et al.*, 2005).

However, even though the development of educational technology stalled in this period, it was a time that saw the development of crucial ideas of information dissemination, which would form part of modern educational technology and practice. They would only much later become reality with the appearance of personal computer technology and the Internet. One such idea was H. G. Wells' concept of a world-wide, permanently updated encyclopedia in contact with all knowledge-related institutions:

> *« Both the assembling and the distribution of knowledge in the world at present are extremely ineffective, and thinkers* (…) *are beginning to realize that the most hopeful line for the development of our racial intelligence lies rather in the direction of creating a new world organ for the collection, indexing, summarizing and release of knowledge* (…)*. These innovators* (…) *project a unified, if not a centralized, world organ to "pull the mind of the world together", which will be not so much a rival to the universities, as a supplementary and co-ordinating addition to their educational activities – on a planetary scale.* (…) *The whole human memory can be, and probably in a short time will be, made accessible to every individual.* (…) *It need not be concentrated in any one single place. It need not be vulnerable as a human head or a human heart is vulnerable. It can be reproduced exactly and fully, in Peru, China, Iceland, Central Africa, or wherever else seems to afford an insurance against danger and interruption. »*

The other idea was Vannevar Bush's conception of a machine and its operation, which would help human beings navigate more easily the vast amounts of information produced in the world. In doing so, he anticipated the use of computers as personal systems for management of information (at a time when their sole purpose was doing calculations):

> *« Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, "memex" will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.*

> *It consists of a desk, and while it can presumably be operated from a distance, it is primarily the piece of furniture at which he works. On the top are slanting translucent screens, on which material can be projected for convenient reading. There is a keyboard, and sets of buttons and levers. Otherwise it looks like an ordinary desk. »*
>
> *(Bush, 1945, p. 10)*

Another important contribution by Vannevar Bush in the same article is more closely related to education: he realized that reasoning based on one's own memorized collection of facts was no longer sufficient for tackling a society where the amounts of information are vast. A non-stated consequence of this view, in educational terms, is that being able to search, evaluate, analyze and judge information was no longer merely useful, but rather had become an absolutely crucial skill in today's world.

> *Presumably man's spirit should be elevated if he can better review his shady past and analyze more completely and objectively his present problems. He has built a civilization so complex that he needs to mechanize his record more fully if he is to push his experiment to its logical conclusion and not merely become bogged down part way there by overtaxing his limited memory. His excursion may be more enjoyable if he can reacquire the privilege of forgetting the manifold things he does not need to have immediately at hand, with some assurance that he can find them again if they prove important.*
>
> *(Bush, 1945, p. 13)*

After World War II, electronic computers had made their debut (cf. section 2.2.4), and were soon adapted for educational purposes. The first attempts to use computers to support learning were strongly connected to the behaviorist theories, particularly the theory of Operant Conditioning by B. F. Skinner (Kearsley, 2004). This theory is centered on the consequences to one's specific actions. Rather than analyze the response elicited by a specific stimulus, Skinner concentrated on the consequences of actions. According to his daughter, while working with rats, he "*discovered that the rate with which the rat pressed the bar* [near it] *depended not on any preceding stimulus* (…) *but on what followed the bar presses.* (…) *this kind of behavior operated on the environment and was controlled by its effects. Skinner named it operant behavior. The process of arranging the contingencies of reinforcement responsible for producing this new kind of behavior he called operant conditioning.*" (Vargas, n.d.).

Figure 39 – Burrhus Frederic Skinner, 1904 – 1990
From: http://www.psychology.uiowa.edu/Faculty/wasserman/Glossary/skinner.jpg

After developing his knowledge on the conditioning of animal behavior, he realized that the actions of human teachers were actively against the principles of operant conditioning: the amount of information presented to the students would require large behavioral changes at one single time, and the feedback provided to the students (evaluation of the answers of students in tests) occurred a long time from the presentation of the information (McDonald *et al.*, 2005). From these observations, he decided that operant conditioning could be used to improve teaching and learning.

Figure 40 – Skinner's teaching machine
From: http://www.americanhistory.si.edu/teachingmath/images/1999-01829.jpg

Skinner presented his ideas for better human learning in a 1954 article entitled "The Science of Learning and the Art of Teaching" (Reiser, 2001, p. 59). In order to use behaviorist theory in instruction, the behavior of each child had to be shaped according to each of the child's learning actions. The teacher would have to prepare the correct sequence of small-sized pieces of information, presented as problems, so that their correct answering could be rewarded with positive stimuli to the child[41].

But in a typical classroom of 20 to 30 pupils, there is little chance of a teacher doing this. So Skinner devised a teaching machine that would present problems for students to do (Figure 40). Such machines, the first of which was presented in March 1954 at a University of Pittsburgh conference (NMAH-BC, 2002), would provide information in small pieces and the feedback was present immediately after each answer, thus allowing the conditioning of the student. While this first machine was never actually used in a classroom, other machines were developed, both mechanical and using computers, and their use eventually resulted in what came to be called the **programmed instruction movement** (Vargas, n.d.). One of these first teaching machines was based on a computer: the IBM 650 (Figure 41 and Figure 42), "*IBM's first commercial business computer*" (Cruz, 2004).

> *« The IBM 650, a high-speed digital computer, interfaced with a typewriter, was used as a teaching machine. »*
>
> *(McNeil, n.d.)*

Due to the huge price of the IBM 650, which had been developed for business and scientific use, its educational use was obviously not widespread. But it did mark the start of educational computer use.

These behaviorist beginnings originated a long line of development of software programs for education, sharing the same philosophical background. Such software is simple to program and develop (Jonassen, 2000, p. 5), and even today many educational materials for personal computers employ these principles at some level.

Historically, a major figure involved in the development of such software was Patrick Suppes. In 1963, he and his colleagues at the Institute for Mathematical Studies in the Social Sciences (IMSSS), at Stanford University, began a research project on the use of computers in education, following behaviorist orientations. Four years later, in 1967, they established the Computer Curriculum Corporation (CCC)[42], aimed at marketing and distributing

Figure 41 – IBM 838 Inquiry Station for the IBM 650
From: http://www-1.ibm.com/ibm/history/exhibits/650/images/3403ph16.jpg

Figure 42 – IBM 650 Console Unit
From: http://www-1.ibm.com/ibm/history/exhibits/650/images/3403ph01.jpg

Figure 43 – Patrick Suppes 1922 –
From: http://www.booknoise.net/flickeringmind/characters/src/patrick.jpg

[41] An example program of self-instruction for classroom use for instructors, following this method, is available on-line, for Microsoft Windows personal computers, from the Web site of the B. F. Skinner Foundation, at this address: http://www.bfskinner.org/instruction/setup.exe.

[42] CCC was purchased by Simon and Schuster Publishing, which then became part of Viacom and was then sold by Viacom to Pearson plc (Viacom, 1998). Currently (July 2004), its know-how is part of Pearson's division named Pearson Digital Learning (http://www.pearsondigital.com).

educational material developed during the previous years, as well as creating new education materials (Solomon, 1986, pp. 16-30; Solomon, 1996, p. 31).

In the early and mid-1960s, the use of what we now call multimedia, such as graphics, sound, video, etc. was either not available on a computer screen or extremely costly. For this reason, the first materials employed only text. A sample interaction with a student would consist of a question such as (from Solomon, 1996, p. 32):

7 x 6 = _____

A correct answer would present the student with congratulations and a new exercise. A wrong answer would provide different feedback, which in the original CCC materials would be something like:

TRY AGAIN

The student would then be presented with the same exercise. Another wrong answer would result in the presentation of the result, and "*TRY AGAIN*". If the third attempt is also wrong, the learner moves to a different exercise, probably at a lower skill level.

While the graphical and sound environments underwent a huge evolution since the late 1950s/early 1960s, the basic philosophy remains the same: divide knowledge is small-sized bits, present them in a given sequence, and present questions to the student. If the student gets it right, he/she will move on to further, possibly more complex, knowledge bits. If the student gets it wrong, he/she will have to review or study previous knowledge bits and answer questions on them, until the system moves on. More recent versions of software using this philosophy might present a graphic scene and ask a student to select a missing object, or to put banknotes together to pay for a purchase, rather than ask to multiply 7 x 6, with sound cheers or visual cues. (Examples with screenshots can be found in Solomon, 1996, pp. 33-35 and 40-41.)

This category of applications, known as "**drill-and-practice**", or "**Computer-Aided Instruction**" (CAI) can produce measurable results, since it leads itself well to formal testing – in fact, testing is crucial in its design. Therefore, its impact on students' results in formal written examinations can be assessed. From the start, a central idea to the use of CAI applications in educational environments was the existence of a management system, in order for the educator/teacher to review the student's progress, which answers he/she got right or wrong, and so on. That information would allow a teacher to focus the efforts towards a student in face of the content areas where the student is showing lesser ease of progression. (As an example, the principles of the management system behind CCC applications are analyzed by Solomon, 1986, pp-16-30.)

It has been demonstrated that CAI use improves, at the very least, the test results of low-scoring students:

> *« The CCC curriculum has its clearest success with students who perform below their grade level according to their scores on standardized tests.* (…) *Evaluative studies conducted between 1971 and 1977 indicate a pattern* (…) *students show a grade place improvement of 1 to 1.5 at the end of their experience in mathematics and about 0.6 year in reading or language arts. »*
>
> *(Solomon, 1986, p. 25)*

There is, however, some controversy regarding its impact beyond the specific set of skills considered in the instructional content and its transfer or use in new knowledge areas:

> *« In 1982, an Educational Testing Service evaluation* (…) *found that students made significant gains on their computational skills. The reading and language arts materials did not reflect the same kind of increased*

> *improvement in test scores. The test score did increase in the first year, but in additional years there was no increase (and no decrease) in test scores. »*
>
> *(Solomon, 1986, p. 25)*

Criticisms to this method of computer-based education are usually based around the fact that "*behaviorist principles underlying drill and practice are unable to account for, let alone foster, the complex thinking required for meaningful learning required to solve problems, transfer skills to novel situations, construct original, ideas, and so on*" (Jonassen, 2000, p. 5). Its defendants argue that drill-and-practice fosters automaticity, and that for higher-order skills to be learned, the lower-level skills must be done automatically (Jonassen, 2000, p. 5; Solomon, 1986, p. 26). I believe that both arguments hold true: a certain level of drill and repetition is probably necessary for basic knowledge to become so embedded that one can move on to using it in more complex tasks. However, while drilling facilitates that embedding of knowledge, it's harder to see how it can help students transfer that knowledge to new situations and problems.

Drill-and-practice, however, isn't the only method of learning from computers in use today: most computer users are bound to have experienced at least another common method, usually called "**tutorial**". A classical tutorial would present information, ask questions (often with answers in multiple-choice style, to avoid language-interpretation problems). If the student gets it right, he/she would be rewarded (as in drill-and-practice). Wrong answers would cause the path of instruction to change, in order to provide remediation adapted to the error, and later the question that was wrongly answered would be presented again (Jonassen, 2000, pp. 5-6).

A specific kind of tutorial is modernly found when the aim is training a human on the use of some specific software application, or some situation that can be modeled or simulated in a computer. For instance, many computer games are quite complex and include tutorials to assist the user in learning to play them. These tutorials usually adopt a hands-on approach rather than a question-and-answer approach, but the philosophy of progress one step at a time remains: the program requests that the user performs some specific task, like entering text on a specific edit box, clicking a button, or dragging an icon, and usually prevents "wrong" options. If the user tries to do something that was not intended, the tutorial program will usually warn about the wrong choice, and direct the user towards the proper course of action, sometimes with extra explanations. By working along the tutorial, the user can learn how to distribute troops, manage resources, develop a civilization, sustain a group of people, and negotiate ethically in intergalactic goods or whatever the game play requires.

This level of interaction strikingly echoes Thorndike's remark quoted at the beginning of this section, wishing that "*only to him who had done what was directed on page one would page two become visible*" (Thorndike, 1912, p. 165, in McNeil, n.d.). It is indeed a powerful technique, by ensuring that the learner does in fact perform all intended actions and drills all bits of knowledge seen as crucial. Its major limitation is that it cannot accommodate different learning needs: should some learner like to explore a different area of knowledge halfway through the tutorial, should some specific concept fail to be understood by learner, there is no option to further explore or test it: the learner only solution is to proceed along the preset path. In other words, tutorials restrict the way in which learners are supposed to construct their own meanings: they must accommodate someone else's interpretation of the world and facts.

> *« Students are not encouraged or even able to determine what is important, reflect on and assess what they know, or construct any personal meaning for what they study. What they too often acquire from tutorials is inert knowledge because they are not applying it. »*
>
> *(Jonassen, 2000, p. 6)*

However, modern computer games have further developed the tutorial concept, and the user may indeed have a greater amount of liberty on the direction of learning and action. Currently, it's not unusual for a game tutorial to be actual game-play, with the tutorial system providing directions and suggestions depending on the evolution of play. The following images are screenshots from the tutorial of the game Civilization III (Figure 44), by Infogrames (Firaxis, n. d.). The tutorial for this game is a nice example of how a player can benefit from actual small bits of advice and directions while playing a real game.

Figure 44 – Boxes of Civilization III

From: http://www.firaxis.com/images/interface3.0/gamespage_box_civ3.jpg

Figure 45 – Civilization III entry screen

From: http://www.firaxis.com/images/interface3.0/gamespage_box_civ3.jpg

Figure 45 presents the tutorial option highlighted amidst the options in the game entry screen. After selecting it, the player initiates the game normally, being presented with just a small bit of information, about the starting situation and goals (Figure 46).

Figure 46 – Civilization III's first bit of information: initial situation and goals

Soon, however, the first directions arrive: the player must move the first unit (a "settler") towards a good site for the capital city. The game also provides advice on what is "a good site" and on the game controls for moving the settler (Figure 47).

Figure 47 – Civilization III's first directions: find a good site for the capital city

As the play progresses, the tutorial system will point to elements in the screen that the user needs to use at that moment and provides explanations (Figure 48).

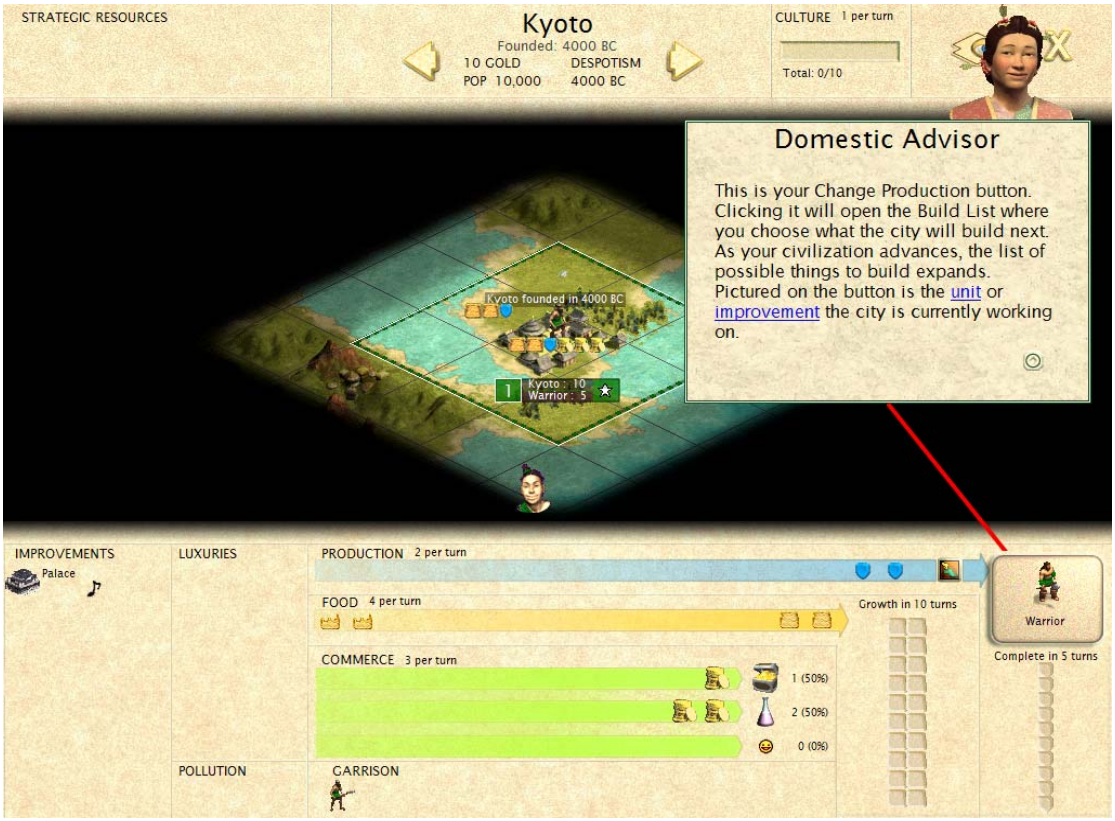


Figure 48 – Civilization III's tutorial explanation of interface elements

In all the referenced pictures, the tutorial text contains blue underlined text, which represents hyperlinks to further, lengthier information. Figure 49 presents an example of an information menu for the "[city] *improvement*" hyperlink in Figure 48. Each element can be clicked for specific information.

Figure 49 – Civilization III's sample screen with hyperlinked information

The user can play as normal, without restrictions. As the play advances, the tutorial system will present information seemingly related to the current situation and directions or recommendation to the actions that should be taken (Figure 50). By "seemingly" I mean that their sequence seldom varies: the player will first be told to build roads around the city, then to create a mine in some square, and so on. It's up to the player to decide whether to do it or not, but the tutorial won't react to that (a more strict, limited form of tutorial, common in many programs, would prevent the user/player from advancing if he/she didn't act as the tutorial says).

Figure 50 – Civilization III's suggestions and directions in mid-tutorial game.

If the user doesn't read, think over or consider the piece of information that the tutorial provides, nothing happens: the tutorial will simply provide the next bit of information later on. The player can repeat the tutorial as often as he/she desires, and given the nature of the Civilization III game, this will present completely new play situations. However, the sequence of suggestions provided by the tutorial will be the same: as stated earlier, the goal of the tutorial is to map specific content into the user. The game relies on its ability to be compelling in order for the player to get to know it better and learn while playing, possibly returning later to the tutorial or reference materials for information that makes sense in a specific situation or game strategy that he/she is attempting to use. But by then, the tutorial isn't the driving force behind learning: its relevance was exhausted in the early stages, providing small bits of knowledge.

There is a clear difference in learning styles, between the phase of tutorial use and the phase of game-play. In both phases, the player is learning how to use the game, but undoubtedly it's when the tutorial is no longer being used that the most profound insights are being achieved. However, it's highly likely that such a level of play proficiency – and therefore, those insights – might not be achieved by many players, where it not for the "training wheels"-style of support provided by the tutorial. And yet, if game knowledge was analyzed in written tests, it's likely that the major increase in tested "knowledge" would be registered during the tutorial or immediately after its completion. This is a clear example of the paradox and on-going debate between the supporters and critics of the "Learning from computers" style of education. (And how, critically, both parties have a point.) The style of learning that takes place during game play is discussed further in the following section, 2.4.4 – "Learning with computers".

The combined ideas from the original CAI and tutorials led to a recent development in this approach to computer-based instruction: **intelligent tutoring systems** (ITS). Basically, these systems adapt the methods of CAI to the student also in terms of content, not just of pace. Typically, these systems employ some level of Atrificial Intelligence techniques to determine which content areas the student may be having trouble with, and provide individualized hints or other textual support; some of these can also create student-specific problems and scenarios on the fly.

> *« (…) ITS goes beyond training simulations by answering user questions and providing individualized guidance. Unlike other computer-based training technologies, ITS systems assess each learner's actions within these interactive environments and develop a model of their knowledge, skills, and expertise. Based on the learner model, ITSs tailor instructional strategies, in terms of both the content and style, and provide explanations, hints, examples, demonstrations, and practice problems as needed. »*
>
> *(Ong & Ramachandran, 2003, p. 2)*
>
> *« An expert human tutor continuously adjusts the content and mode of presentation based on the student's recent responses and the tutor's instructional goals and prior knowledge of the student. Similarly, the courseware presents the student with objectives at selected levels from a mix of curriculum strands in a selected mode, be that a question, a tutorial, brief feedback, or other forms of instruction. »*
>
> *(Thrall & Tingey, 2003, p. 1)*

As a final point, this section presented little comment on the teacher's role during the educational/learning process apart from the initial remarks regarding Thorndike's views (on page 66). This is so because this style of computer use in education is designed to take place without teacher intervention, other than overseer of children/students. The teacher's role is considered to take place before and after the computer activities, not during them (dispensing knowledge or preparing its presentation, beforehand; evaluating the student's progress, afterwards). Teachers are

seen as having to be freed from drilling activities (that computers take care of), to perform other, more meaningful tasks, with the overall goal of conveying knowledge.

> *« Like most of his CCC products, the program wasn't heavy on teacher interaction. "It was designed not to require a tutor," Suppes said. "They're just for trouble-shooting"—that is, responding largely through e-mail. Suppes* (…) [is] *simply trying to find a way around weak teachers, which he seems to regard as education's Achilles heel. "There's a lot of bullshit about teachers. Let's not think they're all beautiful flowers about to bloom. We'd all like to be tutored by Aristotle. But that's not possible." »*
>
> *(Oppenheimer, 2003)*

Thus, the logic of this approach to computer use in education includes the notion that teachers need not be actively involved while children are using the computers. Even in modern ITS systems, this perspective hasn't changed: when developed for use in classrooms, ITS systems also support the classroom teacher, but only by providing him/her with managerial information about the student's progress. The teacher's main role still takes place before or after computer-based training. For instance, a recent research report on the use of one such system indicated its operation in the following fashion: the teacher prepares the computerized tutor, which then takes over teaching; if the student is requiring support, the teacher options are to either take over from the computer tutor and ready the student to return to it, or stay put and let the computerized tutor continue.

> *« To maximize student learning the tutor and classroom teacher need to work together. Therefore the tutor, as emulated by the courseware, should adapt to the teacher's goals for the class and for the individual student, which is accomplished by the teacher's adjustment of strategies for the student within a course or across courses. In addition, the tutor should share with the teacher information about the academic progress of the student, which is accomplished through the courseware reporting system. Based on this information, the teacher can decide when to give the student more freedom or more structure; more direct instruction or more explorations; more focus or more variety. Alternatively, the teacher can let the program adjust to the student and make the decisions about what to do next. »*
>
> *(Thrall & Tingey, 2003, p. 1)*

This is consistent with the behaviorist background that cares above all for the observable causes and reactions, not to what behaviorist theory sees as unobservable internal mental processes, any discussion of which is therefore merely speculative. A consequence of this view is that teachers are supposed to simply prepare and convey knowledge in a setting and fashion that follows the behaviorist theory, and not delve in speculation over the educational developments taking place inside the mind.

> *« Teachers are trained to instruct and not to educate. Professional training in instruction is possible, but not in education, given the present makeup of the teaching profession and the institutions that train them. »*
>
> *(Suppes, 1995, summarizing the educational thoughts of Siegfried Bernfeld with which he sympathizes)*

2.4.4. Learning with computers

Figure 51 – Robert B. Davis, 1926-1997
From: http://www.rbdil.gse.rutgers.edu/bob_davis_01.jpg

While the approach about computer-based learning described in the previous section, "Learning from computers", possesses the longer history, it soon ceased to be the only one in existence. This section, "Learning with computers", presents a style of computer-based learning which, in summary, sees computers as learning aids, to be integrated in the learning-teaching process and provide support for learning (and teaching), rather than become the source of knowledge and control of the learning process. This different viewpoint on the role of computers in education, in contrast with the previously presented behaviorist viewpoint, is possibly associated with the modern tendency to use the term Technology-Enhanced Learning (TEL), instead of referring to "computer-based" education.

The roots of this approach to computers lie in the work of Robert B. Davis, Professor of Mathematics Education at Rutgers University, particularly the project for reform of school mathematics initiated in 1956, called Madison Project. This project "*underlies the elementary school mathematics material implemented on the University of Illinois'* [PLATO[43]] *computer system in the 1970s*" (Solomon, 1996, pp. 7 & 43), which I comment upon further ahead in this section.

One way to render clear the differences between these educational styles is not so much to present the software (which, obviously, is often very different – but sometimes similar), but in discussing how it is meant to be used. Skinner's (and Suppes') goal was to replace teachers, if not completely, at least during computer time, by handing over the drilling to computers and freeing teachers for other tasks: the computer was supposed to be used on its own. Davis, by contrast, sees the teacher as an active element in the educational process, one that must embark on a creative process with the children, "*building on what children already know coupled with challenging children to come up with solutions in new situations through the clever and consistent use of manipulatives*" (Solomon, 1996, p. 57). This, of course, also applies to computer time.

Figure 52 – Screen from an educational PLATO application, in the 1970s
From: http://www.cineca.it/immagini/history/plato.jpg

[43] Programmed Logic for Automated Teaching Operations

In short, it's not so much a matter of describing the software tools, but rather how they can be used as educational tools, rather than how they can make use of the child. But several concepts introduced by Davis in this regard heavily influenced the development of educational software application, two of which the Madison Project made extensive use of: the **paradigm teaching strategy** and **discovery learning**.

Under the paradigm teaching strategy, "*new ideas are introduced through carefully chosen examples*" (Solomon, 1986, p. 32). A teacher would seek to link new knowledge to a child's existing knowledge, effectively building upon it. An example would be to introduce the notion of number fractions by linking it to the process of sharing a candy bar or jumping beans among children, *i.e.*, activities with which children are already acquainted (Solomon, 1986, pp. 32 & 37-39). Software can be introduced in similar fashion, by proposing activities that make similar connections. For instance, Figure 53 presents a screenshot of a modern program, descending from the Davis' ideas, where sums, divisions and fractions are presented in a kitchen setting: the child will have to combine measures in order to adequately prepare the recipe.

Figure 53 – Screen from PLATO Learning's "The Quaddle Family"

From the video at: http://www.plato.com/downloads/movies/QUADDLE_300.mov

Under the discovery learning strategy, "*a teacher might call attention to a problem, but the task of inventing a method for dealing with the problem was left as the responsibility of the student*" (Solomon, 1996, p. 44, quoting Davis). Teachers support such discovery strategies by providing hints and help. This approach aims at enforcing children's use of mental reasoning over canned methods, thus helping the child develop his/her own internal knowledge structure. The teacher interaction provides help in this process, both during build-up and during correction of wrong assumptions, when things don't turn out right. For instance, Figure 54, on page 79, shows a software program intended for this strategy: children explore concepts in a specific context (in this case, an archaeology exploration where word decoding is necessary), and then must solve several problems, with the possibility of trying several strategies.

The publisher's promotional video gives a summary of activity content:

> *« This complete phonics program offers the opportunity to begin with phonemic awareness activities, involving the blending, segmenting, rhyming and isolation of sounds.*
>
> *In comprehensive phonics activities, students participate in decoding exercises, in which they study words to identify initial, final, and medial consonant sounds, as well as long and short vowel sounds.*

In the "word family" activities, students use word family patterns to build their own interactive word walls, and then practice sorting words by word families. »

(Plato Learning Inc., 2004)

Figure 54 – Screens from PLATO Learning's "The Three Decoders"

From the video at: http://www.plato.com/downloads/movies/3_DECODERS_300.mov

Obviously, Davis' early software had the look of Figure 52, not the modern look of the applications presented in Figure 53 and Figure 54, but the main educational ideas presented in the previous paragraphs were already developed from the start.

As an historical side-note, Davis initial software was developed for a computer system known as PLATO (Programmed Logic for Automated Teaching Operations), whose development was initiated in 1961 at the University of Illinois at Urbana-Champaign by Donald L. Bitzer and H. Gene Slottow. PLATO's original aim was to be an "*automatic 'teaching machine' to teach students at various levels how to use a high-speed computer (...) the five-ton ILLIAC I*" (Hutchinson, 2003). Through several evolutionary generations, the development of the PLATO system led to the creation of several revolutionary technologies, such as plasma displays for presentation of graphics, and touch-sensitive screens (Figure 56), which simplified child interaction. PLATO was usually used in network settings, which led to other innovations, in the field of software. These included instant messaging, e-mail, emoticons and several other collaboration features (Dear, 2004). A nice example of its impact on the users and the overall computing environment is this remark to The New York Times by Ray Ozzie, creator of the software called Lotus Notes (that came to represent "teamware"):

Figure 55 – Donald L. Bitzer
From: http://www.ncsu.edu/BulletinOnline/photos/bitzer.jpg

> *« After graduation, I said to myself, "By hook or crook, I am going to build software to recreate the interactive environment I'd used with Plato." That thought led to the creation of Lotus Notes, which sits on nearly 100 million desktops, as well as everything else I've done. »*
>
> *(Ozzie, 2002)*

PLATO became a successful product that was marketed by the company Control Data, which in 1976 acquired the rights to the PLATO name. The PLATO courseware was sold to TRO Learning, Inc. in 1989, which in 2000 changed its name to PLATO Learning, Inc., "*to capitalize on the strength of the PLATO brand name in the educational software market*" (Plato Learning Inc. n.d.). The University of Illinois continued with the development of computer-based education programs, but since the PLATO name was no longer available, the system eventually became known as NovaNET, in 1988 (Dozen, 1998). Ironically, NovaNET is now marketed by Pearson Digital Learning, which means that the flagship examples of this and the previous section found their way underneath the same corporate umbrella.

Figure 56 – PLATO IV station with touch-sensitive plasma display
From: http://www.ece.uiuc.edu/ingenuity/503/images/PLATO_IVb.jpg

Since the early PLATO days, this line of educational computer use saw the development of other strategies and novelties, and is now a very active line of educational computer use.

One striking feature that is noticeable from the application screenshots in this section is that they seem to be fun – *i.e.*, they appear to be games, not educational products. That is all to be expected: because indeed they are games. Currently, most educational computer software for children blends the educational and entertainment components, which is a direct consequence of Robert Davis' paradigm teaching strategy, which employed children's everyday activities in education. And of course, a large part of children's everyday activities is child's play. This breed of

software is called **edutainment**, "*a place that asks children to enjoy what they are learning with a combination of sound, animation, video, text, and images*" (Druin, 1996, p. 64). It is also a growing multimillion euro worldwide industry[44].

This blend of entertainment and education means that this kind of software is used both inside and outside the classroom, with and without teacher assistance or intervention. In fact, its prime business target is usually the household, not the school. A consequence of this need for autonomous use by children led to a steady evolution in user interface design and simplification. In 1996, while describing a product called *My First Encyclopedia*, Allison Druin mentioned:

> *« It throws away the traditional interface of layered windows, pull-down menus, and buttons in favor of a tree.* (…) *there is little text for preschoolers to struggle with: instead, there are video kids that explain ideas and lead activities. »*
>
> *(Druin, 1996, p. 78)*

In light of these changes, the modern version of learning with computers is done under two perspectives: **learning with computers autonomously**, and **learning with computers in schools** (or other teacher-supported learning environments). While this distinction in use is also present in the previous model ("Learning from computers"), it is not so relevant, because the teacher involvement during computer use is null or almost so.

While learning autonomously, it is crucial that computer software possesses **good self-learning features**. The learner must be able to realize on his/her own how to use the software, and must want to use it long enough for it to be effective. In a setting with teacher coaching, these self-learning features of software are not so relevant, since the teacher or a student partner can introduce the learner to the software application, and probably provide support and orientation during its use. However, being easily learnable is also a useful feature for any software that is to be used in an educational setting, lest the learner spends more time and effort learning how to use the software than the benefits that result from it (Jonassen, 2000, pp.18-19).

There is an inherent reason for using the term "good self-learning features" in software for autonomous use, and "being easily learnable" for software intended for teacher-assisted settings. In this latter situation, the software is mostly a tool, which can be easy, hard or almost impossible to understand without extra support (be it a teacher, a manual or a textbook); but for autonomous learning, it is imperious that the software allows itself to be learned, while the user is actively involved. Extra support is relevant only in order for the user to evolve faster or beyond certain complexity barriers, not for initiating the use of the software. Games are a nice example: most children start playing games right away, without reading manuals or instructions. Many games provide manuals and tutorials, but when children (or adolescents, or adults) turn to them, it's usually after already being interested in the game proper.

> *"One day I decided I wanted to help my child play Pajama Sam in No Need to Hide When It's Dark Outside.* (…) *I decided to play through the game by myself so I could "coach" my child as he played. (Now he charges me a dollar any time I attempt to "coach" him when he is playing a video game – he calls it "bossing him around" and "telling him what to do when he can figure it out for himself.") »*
>
> *(Gee, 2003, pp. 4-5)*

[44] "*Total videogame software and hardware sales in the United States reached $8.9 billion, versus $7.3 billion for movie box-office receipts; $6.6 billion of the videogame receipts were from software sales, retail and online*" (Poole, 2000, p. 6).

James Paul Gee (2003, p. 6), called this "*having good principles of learning built into its design*", and games (be they edutainment or not), are excellent examples of this:

> *« So here we have something that is long, hard, and challenging. However, you cannot play a game if you cannot learn it. If no one plays a game, it does not sell, and the company that makes it goes broke. Of course, designers could keep making the games shorter and simpler to facilitate learning. That's often what schools do. But no, in this case, games designers keep making the games longer and more challenging* (...)*, and still manage to get them learned. »*
>
> *(Gee, 2003, p. 6)*

Thus, as a result of both the early experiences of adapting learning to children's activities, and of product-development in the video and computer game market, there is nowadays this class of edutainment and entertainment software that produces an active learning experience by itself, for a huge variety of content, both concrete ("curriculum content", so to speak) and meta-level (learning, exploration and reasoning skills)[45].

The other approach, learning with computers in schools[46], has evolved from Robert Davis' approach to computer-based education, with the teacher in a crucial role. Nowadays, this approach is epitomized by the approach of considering **computers as mindtools** (Jonassen, 2000). Its relevance comes from the fact that computers can't be very good at assessing children's learning, other than using tests; and also because autonomous applications, however successful and learnable, may fail to connect with a specific child or a specific situation. In fact, adequate autonomous-use software for a specific purpose may not even exist. A teacher-based educational setting can be adapted to the needs of a specific child.

Figure 57 – David Jonassen
From: https://courses.worldcampus.psu.edu/welcome/insys446/images/Jonassen.jpg

Using computers as mindtools means using software tools to "*function as intellectual partners with the learner in order to engage and facilitate critical thinking and higher order learning*" (Jonassen, 2000, p. 9). What this means is that under this view computer software is split into two distinct groups: productivity tools and mindtools. Using software as a productivity tool means using it to simplify the production of some artifact, but not to radically change the way it is created. Word processors, for instance, are a common example of a productivity tool: they allow text to be deleted, reshaped, resized; warn on possible misspellings; present a view of the text organization on the printed page; automate tedious tasks like creating a table of contents, index or updating internal references. But for many – if not most – people it doesn't fundamentally change the way word is put to writing.

> *«* (…) *word processors have made all of us more efficient, effective, and productive writers. What is questionable is whether* (…) *word processors have made us* ***better*** *writers.* (…) *They certainly facilitate the process* (…) *but they do not necessarily amplify that process. I am not convinced that William Faulkner's novels* (…) *would have been significantly enhanced had Faulkner used a word processor rather than his manual Royal typewriter. »*
>
> *(Jonassen, 2003, p. 16)*

[45] In his 2003 book, James Paul Gee goes on to deduce, analyze and present 36 "learning principles" that can be extracted from video and computer games.
[46] Some teachers employ videogames that aren't labeled as edutainment in their classes. *E.g.*, for a first-hand account with college students see Amory et. al. (1999).

Of course, not all word-processing tasks are literary ones, and even for those this view is perhaps too strict. For a regular writing task, by a regular writer, I find it likely that a simple feature such as "search" can immensely improve a writer's product. Considering that any line of dialogue by a character can be prefaced by something like "[Butler]", for instance, means that a writer can in a short time check all the character's lines and more easily refine his/her manner of speaking, for instance. Or find the places and circumstances where a specific adjective was used – something that would require a prodigious memory for a traditional writer. This is but one example of how a simple tool like a word processor can seriously impact the user's process. But whether these impacts are "radical" is debatable. Also, this observation doesn't change the point that some tools, such as word processors, are mainly devoted to improving and rendering easy the execution of some task.

By contrast, mindtools provide alternative ways of thinking. This is part a feature of the tools itself, part a matter of the way it is used. So, an example is helpful to clarify what a mindtool is: a spreadsheet, for instance, can be used both as a productivity tool and a mindtool (spreadsheets were already mentioned in section "2.3.2 – End-user programming").

In a spreadsheet, a user can enter numerical data in specific cells in a grid. Typically, spreadsheets are used for accounting, allowing the automation of multiple calculations. The user can enter, for instance, the prices of all products sold in a given circumstance, and get the resulting sum in another cell; if there is a mistake in one of the entered numbers, it suffices to edit its cell and the resulting sum is updated instantly. Quite often, this ability for effortlessly production of calculation results is used also to support decision-making: instead of adding known values, for instance, several different values can be entered in the same cell, to analyze the resulting impact in the result.

When being used to automate calculations, a spreadsheet is a productivity tool: what it does could also be done by hand (and had been done by hand for centuries). However, when used to analyze the impact of different values, this implies a serious change: the user-accountant is no longer just a calculator, but also a "*hypothesis tester (playing 'what if' games)*" (Jonassen, 2000, p. 87).

But spreadsheets even go a step beyond, since most users have to construct their own spreadsheet formulas. When a spreadsheet cell (for instance at column A, row 10) presents the sum of all cells above it (column A, rows 1-9), that's because someone entered a summing formula in that A10 cell (probably "=SUM(A1:A9)"). When such operations are commonplace, customized accounting or decision-making applications are usually developed and sold, so that computer users can benefit from this behavior, without having to know how to build such formulas. So people that use spreadsheets, rather than specific accounting applications (or other numeric processing applications) usually end up resorting to them for those tasks that bear some particularity, some feature specific to the user's situation, for which there is no specific software application. This means that spreadsheet users must enter their own formulas, to produce the desired calculation results. To do so, they "*engage a variety of mental processes* (…) *to use existing rules, generate new rules describing relationships, and organize information* (…). *The emphasis in constructing a spreadsheet is on identifying and describing those relationships in terms of higher order rules*" (Jonassen, pp. 87-88).

So, in short, using software as a mindtool involves using its abilities to attain a deeper understanding of the matter being studied or worked upon. This requires teachers to prepare activities that propel students to use the software in such a fashion, by analyzing content domains, guiding study, and coaching the use of the software, by helping students plan, adapt existing models of using the tool, creating and completing specifications, extrapolating and reflecting (Jonassen, 2000).

« Mindtools are knowledge representation tools that use computer application programs such as databases, semantic networks (computer

> *concept maps), spreadsheets, expert systems, systems modeling tools, microworlds, intentional information search engines, visualization tools, multimedia publishing tools, live conversation environments, and computer conferences to engage learners in critical thinking.* (…) *They can be used across the school curricula to engage learners in thinking deeply about the contents they are studying. »*
>
> *(Jonassen, 2000, p. 19)*

As a final note, it's interesting to realize that there are several connections between what Jonassen refers to as "mindtool" use and the instances of computer programming I presented in section "2.3.2 – End-user programming". In his 2000 book, he gives up on finding mindtool use for word processors, leaving open only the possibility that his concept of a mindtool may be less inclusive than his reader's. But I believe that style-definition activities for text, such as those described in section 2.3.2 fall within Jonassen's concept of a mindtool. And in fact several parts of what he calls "mindtool" use could in fact be related to computer programming skills.

However, Jonassen objects to considering computer programming languages as mindtools (his remark refers just to Logo, but his motives are applicable to other languages), because they fail one of his rules for defining mindtools, which is that they should be easily learnable:

> *« Although Logo is a syntactically simple language, it still requires several months of practice to develop skills sufficient for easily creating microworlds. »*
>
> *(Jonassen, 2000, p. 156)*

As it happens, the creation of microworlds is not the goal of using a programming language in education, although it is one of the methodologies for their use. There are more than a few uses of computer programming that can be learned and applied effectively in well under 1 or 2 hours, so this objection is debatable in face of Jonassen's own definition for the rule of being "easily learnable", which I transcribe here in its entirety, for the sake of debating it:

> *« The mental effort required to learn how to use the software should not exceed the benefits of thinking that result from it. If weeks of effort are required to learn software, the software becomes the object of learning, not the ideas being studied. The syntax and method for using the software should not be so formal and so difficult that it masks the mental goal of the system. Software programs that are overly complicated to use are not good Mindtools. The basic functionality of the software should be learnable in 1 to 2 hours. You may want students to think causally about information in a knowledge domain, but if the system requires weeks of agonizing effort to learn, the benefits of thinking that way will be outweighed by the effort to learn the system. »*
>
> *(Jonassen, 2003, pp. 18-19)*

I propose analyzing reading and writing as mindtools, albeit not computer-based ones. Jonassen indirectly acknowledges them as such when he categorizes language as a "cognitive technology" (2003, p. 13), when discussing the attributes of mindtools. But you can't master reading or writing in 1 or 2 months, (for some children, not even in 1 or 2 years) let alone 1 or 2 hours. And for a few children, the words "agonizing effort" would certainly serve as an apt description of their feelings as they struggle in school to learn how to read and write. So, are reading and writing not mindtools, or at least bad mindtools? And are they being learned by their own sake? Rather, they are being learned (from the point of view of the individual) to prevent children to be limited in their adult lives by lack of literacy, with all assorted consequences (personal, professional, social, etc.).

The classification of reading and writing as mindtools, providing alternative ways of thinking, is supported by recent neurology research. Comparison studies conducted between literate and illiterate adults revealed significant differences in "*performance of specific tasks and, in some cases, in the pattern of functional brain activation and in the volume of certain anatomic regions*" (Castro-Caldas, 2003). Recently, this was even further emphasized when research revealed that adults that learned to read and write as children were faster and more accurate at performing tasks not related to reading or writing, than adults that only learned to read and write as adults.

> *« Results concerning visual processing, cross-modal operations (audiovisual and visuotactile), and interhemispheric crossing of information are reported. Studies with magnetoencephalography, with positron emission tomography, and with functional magnetic resonance provided evidence that the absence of school attendance at the usual age constitutes a handicap for the development of certain biological processes that serve behavioural functioning.* (…) *the occipital lobe processed information more slowly in cases that learned to read as adults compared to those that learned at the usual age. »*
>
> *(Castro-Caldas, 2004, abstract)*

In my view, the definition of the "easy learning" rule was written specifically with the aim of attempting to exclude computer programming, which was already being used in education for well over a decade when Jonassen wrote his book on mindtools. The initial sentence, weighting effort and benefits, would suffice as a perfectly reasonable and apt definition of "easily learnable".

I believe that these concerns include somewhat of a bias on the part of Jonassen against acquiring knowledge inherent to the computer domain, something that becomes apparent when he states that "*many more students are* [nowadays] *able to use computers without instruction*" (Jonassen, 2000, p. 8), in order to dismiss instruction about the operation of computers and their software – something I strongly disagree, due to my experience as a teacher of computing to 18, 19 and 20 year old student teachers, if by "use computers" he means "use computers effectively." While many things about computers are indeed as unnecessary to modern living as learning to use a quill or learning to hunt, it's perhaps too much to demand that computer systems are as unobtrusive as a TV set or toaster, particularly given their metatool characteristics, put forward in section 2.1 – "Body & Mind: hardware and software". In my view, there is no need to fear learning something about computers while using them to learn something else. What's more, while learning a computer programming language can indeed be made troublesome and annoying, it can also be an exciting and enthralling experience, that profoundly involves learners.

> *« Thinking back over the thousands of hours we've watched kids programming in Logo, one things stands out: the fun that some have just playing around, not always exploring or creating, but playing around with things they've already made. All of us have experienced the contrast: the painstaking care, say, to create a square, the exploration to find just the right angles, the false starts. But then, when the square is spun round to make a star, the atmosphere invariably changes, something else comes alive and playfulness emerges the sheer fun of carelessly or rather, nonchalantly playing around with the thing that's just been created.»*
>
> *(Hoyles & Noss, 1999)*

That this latter situation is not always the case in many people's personal experience is perhaps the motive for Jonassen's lack of endorsement. Some historical background supporting this is presented in the next section, 2.4.5.

The line of reasoning I used above for reading and writing could also be applied to most of the curriculum content present in schools, so there is also some conflict regarding Jonassen's

emphasis on mindtools for learning and the content being learned, which to be consistent should be selected as mindtools for living.

The exclusion of computer programming from Jonassen's "mindtool" classification is one of the reasons (but not the single one) it is presented it in this thesis as a separate model of educational computer use: the use of computer programming, described in the next section, "Learning about thinking, with computers", has profound implications on what and how people learn, warranting its own section. (At the end of which I'll come full circle back to this section.)

2.4.5. Learning about thinking, with computers

This fourth model of educational computer use has two main connections with the previous one, "Learning with computers":

- the major aim of computer use is not dispensing information or knowledge to the students, but rather being used to enhance the learning process, by allowing students to explore or better connect to the subject of study[47];
- the learning process may involve a design and development process, *i.e.*, a process of construction of an actual product or artifact (not necessarily a physical one).

The main difference between these models is that in "Learning about thinking, with computers", the exploring of concepts is entirely centered on fostering opportunities for students to realize how they think about the concepts in hand. This model picks on the motto that the best way to learn a subject is to teach it, by proposing to solve a major problem with its implementation: teach whom?

This model's answer is "teach a computer". That is, by teaching a computer how to do something, the aim is that children not only learn about that something, but also about how the computer thinks, and in the process, how they think themselves. This remark may sound puzzling to some readers. After all, computer-style "thinking" is quite different from human-style thinking. Under this educational model, this contrast is seen as extremely valuable. As I explain in section 4.1.2 (p. 259), young children (and even adults) are often egocentric, in the sense that they often fail to realize or account for the existence of different viewpoints and manners of thinking other than their own. "*By providing a very concrete, down-to-earth model of a particular style of thinking, work with the computer* [by programming it] *can make it easier to understand that there is such a thing as a 'style of thinking.'* " (Papert, 1980, p. 27).

Another difference is that in this model the construction of an actual product or artifact goes beyond the "may happen" level of the previous model: here, this construction is something that is almost always involved in the learning process.

Historically, the use of computer programming in education began in the late 1950s, mostly with undergraduate engineering students; the approaches varied between the use of machine-code or assembly languages (*vd.* section 2.2.5), and the use of the first human-friendly programming languages, such as Fortran (*vd.* section 2.2.6). By 1960, this was being seen as a matter of choice between "*computer-oriented or problem-oriented*" programming languages (Katz, 1960, p. 527). Already then, the main goal of teaching programming was "(…) *to use computers better and more wisely by virtue of understanding them first as general tools to be used in reasoning through solutions of problems per se rather than as devices to solve particular problems – because, if the latter, the students* (...) *may not be able to generalize the experience appropriately*" (Alan Perlis, acc. to Katz, 1960, p. 522). But the first widespread language created with educational (rather than mathematical) goals was BASIC, developed in 1963/1964, by John Kemeny and

Figure 58 – Thomas Kurtz & John Kemeny
From: http://www.atariarchives.org/deli/kurtz_and_kemeny.jpg

[47] Kahn (n.d.), includes learning about thinking, the title of this section, as but one of several critical thinking skills that can be learned through computer programming: problem decomposition, component composition, explicit representation, abstraction and thinking about thinking.

Thomas Kurtz, at Dartmouth College, in Hanover, New Hampshire, USA (Kemeny & Kurtz, 1984; Hauben, 1995).

> *« Dartmouth acquired its first computer in 1959, a very small computer called the LGP-30. Kemeny facilitated the use of the LGP-30 by undergraduate students. The ingenuity and creativeness of some of the students who had been given hands-on experience amazed the Dartmouth faculty. Kemeny and Thomas Kurtz, also of the Dartmouth math department, were thus encouraged to "set in motion the then revolutionary concept of making computers as freely available to college students as library books." (Portraits in Silicon, Robert Slater, Cambridge, 1987, p.22) The aim was to make accessible to all students the wonderful research environment that computers could provide. »*
>
> *(Hauben, 1995)*

This aim of Kemeny and Kurtz led to the development of a computer time-sharing system, so that many students could have access to a computer. But that wouldn't be enough: at the time, computer programming methods and languages were highly abstract and dependent on the operation of the hardware. Their desire to render computer programming accessible to large numbers of people meant that simplification and generalization for several hardware platforms was in order: programming needed to be usable by people for whom computers where not the main focus of study or work. This led to the creation of the BASIC language.

> *« The design goals for BASIC included ease of learning for the beginner or occasional programmer, hardware and operating system independence, the ability to accommodate large programs written by expert users, and sensible error messages in English. »*
>
> *(Kemeny & Kurtz, 1984)*

Fundamentally, BASIC became the first language of choice for using programming in education due to fact that it was the first language aimed at non-professional programmers. It had a syntax that aimed to resemble the English language, and also intended to run independently of hardware.

Being the first such language, however, was also a source of problems for the development of BASIC. At the time, Kemeny and Kurtz didn't produce a commercial-quality implementation of the language, and although Dartmouth College had the copyright to BASIC, it was made available free of charge (Hauben, 1995). Initially, the language and its base system were used in "*campuses and government and military situations*" (Hauben, 1995). The spreading of the language occurred when the first microcomputers appeared, but their limited hardware capabilities forced most versions of BASIC to be technically simplified (which usually meant losing some sort of human usability). Particularly successful was a subset of BASIC commands written in 1975 by Bill Gates and Paul Allen for the Altair computer. The explosive spread of the use of microcomputers during the 1980s[48] meant that most people using a computer had access to some form of the BASIC language. By the end of that decade, 10 to 12 million school children had learned BASIC (Hauben, 1995).

[48] From 1982 to 2000, the percentage of USA households with personal computers grew from around 2% (U.S. Bureau of the Census, 1988) to approximately 51%. This growth was faster in recent years than during the 1980s, when this figure grew from 2% to 15%. However, that decade represents the first period of this explosive growth (General Motors, 2003; U.S. Bureau of the Census, 1993). In schools, however, this growth is indeed remarkable in the 1980s: in 1981 only 18.2% of all public schools had microcomputers, but this value had already grown to 97% by 1989 (U.S. Bureau of the Census, 1993).
Statistics for Portugal and the European Union:

- Portuguese households with computer in 1995: 11%; in 2002: 28% (INE 2002).
- European Union households with computer in 2000: 35% (INRA 2000).

All along, the BASIC language developed by Kemeny and Kurtz had evolved from its primitive form, incorporating advanced computer science topics that were appearing in languages. But since each computer manufacturer had its own version of BASIC, which as I referred above was often limited due to hardware constraints or mere carelessness, the BASIC language that most people used was but a pale image of this.

> *« So the BASIC in our family was a sophisticated language with a good collection of structured constructs, sophisticated graphics, all the good modularization tools, etc. What a far cry from what most of the rest of the world had to use.* (…) *To give you an idea of how others had prostituted our beautiful language* (…) *»*
>
> *(Kemeny & Kurtz, 1984)*

In fact, the BASIC that was used in schools throughout those years was some form or other of the limited versions that Kemeny and Kurtz dismissed as "street BASIC". And those forms of the language clearly showed their origins from the mathematical and calculus worlds:

> *« BASIC was designed for a specific audience to replace FORTRAN. The audience was to be calculus students, scientists, or engineers who needed to compute complex series of calculations repeatedly.* (…) *What of the people who do not understand the math and science algorithms? What of the people who want to use the computer for nonnumeric programming?»*
>
> *(Solomon, 1986, pp. 90-91)*

In the quote above, Solomon was talking about the language versions commonly found and used in microcomputers. It mattered little that BASIC was evolving steadily in Kemeny and Kurtz's laboratories, incorporating features such as modularity and graphic commands: the language in hand was not matched to the needs of millions of potential users (children, teachers, and parents) by whom it was being used in homes and schools.

> *« Teachers are finding BASIC difficult to use with their students.* (…) *One of the problems teachers have in teaching children to program is that they emphasize learning the vocabulary and the grammar of a programming language instead of emphasizing the process of exploring ideas. »*
>
> *(Solomon, 1986, p. 92)*

This account remarkably echoes Jonassen's concerns regarding the time and effort involved in learning a programming language, presented at the end of section 2.4.4. But Jonassen (2000, p. 156) was referring to the Logo language, not to BASIC. The irony is that Logo, which I'll present later in this section, is a language that was developed and evolved with the specific aim of being used by children, rather than non-professional (but adult) users. And yet one can find accounts regarding Logo that bear a resemblance to Solomon's remark above, albeit at a higher level of analysis.

> *« Young children and their teachers* [using Logo] *very soon hit a ceiling of what is possible for them within reasonable time and expertise constraints.* (…) *the set of tools and metaphors which are appropriate for navigating around the interface level are not functional below that level – to get below, one has to enter a new world of arcane (usually textual) difficulty that is disconnected from most of the things which made introductory work so engaging. One way round these problems has been the development of microworlds. »*
>
> *(Hoyles & Noss, 1999)*

But this ironic situation is food for thought: as computers evolved, bringing programming to end-users has been a recurrent effort on the part of several researchers, technicians and even

hobbyists. I've frequently met people that found computer programming to be an incredibly enjoyable mental activity, so much so that trying to teach programming (or "show how to program") to friends, colleagues and children was for those people so natural an impulse as to show a favorite book, TV series, landscape, whatever. The following piece, from an e-mail I received, is a clear example of this:

> *«* (…) *I spend the better part of every weekend playing with N, the child who I'm introducing this to. I often have to take breaks though to do other work because I'm a student and have my own programming projects besides. Often when I'm programming he'll sit in my lap and "help" me. My intent is to introduce him to toontalk*[49] (sic) *starting this weekend by informing him that I have a programming project that I have to work on. I will then spend roughly 20 minutes "working" and introducing him to each of the main elements of the system. For example on the first day I'll probably go through the process of decorating my house,* (…) *»*
>
> *(Berthold, 2004)*

One must have in mind that programming languages intended for non-professional users began with BASIC in 1964. That accounts for only 40 years of development in this technology. For automobiles with internal combustion engines, 40 years of development means 1903 (Barach, 1999); for self-powered aircrafts, 1943 (USCFC, n.d.). Naturally, the modern pace of evolution is faster than it was throughout human history, but even comparing this to the evolution of its sibling fast-developing technology, computers, 40 years of development since Zuse's Z1 means only 1981. And automatic mobile phones services made their 40th anniversary in 1988 (Farley, 2005). But it's fair to notice that other technologies gained faster acceptance: electronic television sets turned 40 in 1967 (Bennett-Levy, 2001), already very popular[50]; and Internet's predecessor, the ARPANET, went on-line in 1969 (Hauben, 1997), only 36 years ago.

Programming a computer remains a task that requires effort and focus to be pursued beyond simple initiation activities. I believe that this is but to be expected, since it is a task involving deep reasoning; it is not a simple reproduction or adaptation of a mode of thinking most people commonly use every day.

Computer programming is done with several kinds of programming languages, whose history was presented in sections 2.2.5 and 2.2.6 (a technical framework and presentation is done in chapter 1). Their history parallels computers' history, in that a technology that was developed to facilitate numerical calculations was put to the facilitation of everyday, non-numerical activities. In my opinion, such evolution is nothing short of phenomenal, but cannot easily evade its strict numerical and logical roots.

> *« When I first started to learn about programming* (…) *my idea of how it should work was that it should be like teaching someone how to perform a task.* (…) *Imagine my shock when I found out how most computer programmers actually did their work.* (...) *There were these things called "programming languages" that didn't have much to do with what you were actually working on.* (...) *Wait a second, I thought, this approach to programming couldn't possibly work! I'm still trying to fix it. »*
>
> *(Lieberman, 2001, p. 2)*

[49] A programming language described in section 3.3.5, which I employed for my field activities.
[50] In 1970, in the USA, 95.3% of all households had at least one television set (U.S. Bureau of the Census, 1988, p. 561).

This imperfect match of tool and purpose[51] has important consequences for the use of programming languages in education: what programming languages are nowadays is certainly but a glimpse of what they will become, not merely technically but also conceptually, *i.e.*, in the type of mind-frame that programming will require or allow. One of the first educational researchers on the use of computer programming for educational purposes, Thomas Dwyer, once said:

> *« (…) it seems to me that attempting to innovate with supportive systems that don't begin to match the sophistication of the human learner should be viewed as a betrayal, not a consequence, of a humanistic approach to education. »*
>
> *(Dwyer, 1971, p. 100)*

There is thus nowadays still the need for the educator employing computer programming to strive to adapt, change, and indeed recreate the programming tools being used, in order to maintain the focus of each student's reasoning on the reasoning itself[52].

The computer's educational role in this regard is to interpret the student's reasoning expressed through a program, exposing to the student the relativity of one's statements, but also one's misconceptions or unconsidered paths that are required to achieve the intended goal. Another way to put it is that the computer can present to the student a reflection of his/her ideas, statements, and activities. This reflection can thus be used by the student to gain insight on his/her idea, statement or activity. A particular feature of this computer-based reasoning process is that the computer can iterate one's ideas thousands or millions of times, and help the student achieve a realization or understanding of long-term or widespread impacts of an idea

In the previous paragraph, I presented just one educational role for computers, but used two completely different formulations. And possibly other formulations could be made. These formulations can also be seen as distinct viewpoints on computer use. And viewpoints usually dictate the viewer's style of action. The consequence is that the "imperfect match of tool and purpose" I mentioned above is impacted by different human styles of approaching and using the tool to achieve the purpose (the tool here being the use of programming languages). These two complex relationships, **tool↔purpose** and **tool↔style**, drive the use and misuse of computer programming as an educational model.

[51] Regarding this tool↔purpose problem, a researcher referred to it as a gap, saying that "*for novices, this gap is as wide as the Grand Canyon*" (Norman, 1986, as cited by Smith *et al.*, 2001).

[52] Alan Kay stated, in 1984: "*The computer field has not yet had its Galileo or Newton, Bach or Beethoven, Shakespeare or Molière. What it needs first is a William of Occam*" (p. 43). I believe that this can also be said of educational computing, but that, fortunately, both computers and educational computing have had a nice share of their Pythagoras and Socrates.

Among the pioneers of the educational use of computer programming, one of the first to recognize this complex relationship was Seymour Papert. Before devoting his research interests towards the educational use of computers, he worked in Geneva with Swiss epistemologist Jean Piaget[53] (Logo Foundation, 2000). He also conducted research in Artificial Intelligence, having founded the MIT Artificial Intelligence Laboratory with Marvin Minsky, in 1964.

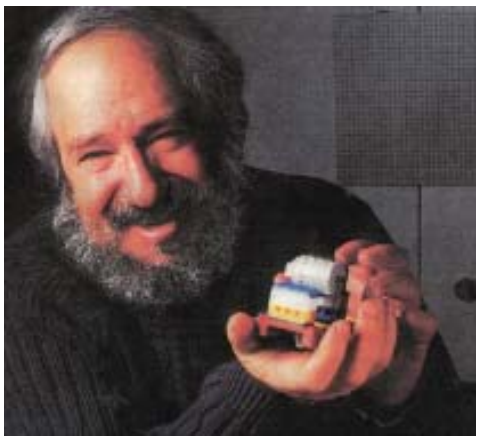


Figure 59 – Seymour Papert
From: http://www.papert.net/images/content/Papert-Leggos.JPG

> *« In 1964* (…) *the focus of my attention was on children, on the nature of thinking, and on how children become thinkers. I moved to MIT* (…) *now my immediate concerns were with the problem of Artificial Intelligence: How to make machines that think? »*
>
> *(Papert, 1980, p. 208)*

While working at the MIT, Papert initially focused on the problem of adapting the tool to the purpose. The purpose was not merely instructing or educating children, but rather a corollary of his research path on thinking: "*seeing ideas from computer science not only as **instruments of explanation** of how learning and thinking in fact do work, but also as **instruments of change** that might alter, and possibly improve, the way people learn and think*" (Papert, 1980, pp. 208-209).

Instead of trying to use the programming languages available at the time, in 1966 and 1967 Papert designed a new programming language in collaboration with Wallace Feurzeig and Daniel Bobrow, from Bolt Beranek and Newman, Inc., called Logo[54] (Feurzeig, 1984; Papert, 1980, p. 210; Logo Foundation, 2000). Logo was then implemented by a team led by Feurzeig and Bobrow, at that company[55] (Feurzeig, 1984; Papert, 1980, p. 210).

> *«* (…) *designing a computer language that would be suitable for children. This did not mean that it should be a "toy" language. On the contrary, I wanted it to have the power of professional programming languages, but I also wanted it to have easy entry routes for nonmathematical beginners. »*
>
> *(Papert, 1980, p. 210)*

A different summarizing remark is that he set to take the best ideas in computer science and child-engineer them (Papert, 1977, in Kahn, 2001a).

> *« The name LOGO was chosen[56] for the new language to suggest the fact that it is primarily symbolic and only secondarily quantitative. »*
>
> *(Papert, 1980, p. 210)*

[53] It is therefore little wonder that he followed on Piaget's constructivist educational philosophy, presented in section 4.1.2. In his book *Mindstorms*, Papert states: "*I left Geneva enormously inspired by Piaget's image of the child, particularly by his idea that children learn so much without being taught*" (Papert, 1980, p. 215). But rather than being a mere follower, he provided new insights and contributions, under the urge noticeable from the continuation of the above quote; "*But I was also enormously frustrated by how little he could tell us about how to create conditions for more knowledge to be acquired by children through this marvelous process of 'Piagetian learning'* " (*id., ibid.*).

[54] From the Ancient Greek word λόγος, meaning "word thought" (Kypros-Net, n.d.).

[55] "*The initial design of Logo came about through extensive discussions in 1966 between Seymour Papert, Dan Bobrow and myself. Papert developed the overall functional specifications for the new language, and Bobrow made extensive contributions to the design and did the first implementation. Subsequently, Richard Grant made substantial additions and modifications to the design and implementation, assisted by Cynthia Solomon, Frank Frazier and Paul Wexelblat. I gave Logo its name*" (Feurzeig, 1984).

[56] By Wallace Feurzeig (see footnote 55, on page 92).

Logo is contemporary of the use of computers in CAI (section 2.4.3) and of BASIC. Given the limited graphical capabilities of computers in the 1960s, it was a language based on written commands, like all other languages at the time. But its specific aim, right from the initial design phase onwards, was to be used by children programmers. This meant that several of its key features were planned, rather than being added as workarounds, in order to allow the programmer to focus more on the idea of the program, and less on its written implementation (in computer-science terms, to spend more time programming and less time coding).

> *« In many schools today, the phrase "computer-aided instruction" means making the computer teach the child. One might say the computer is being used to program the child. In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building. »*
>
> *(Papert, 1980, p. 5)*

Examples of such features are simplified syntax and semantics, well-designed error messages, transparent definition of procedures and direct interpretation of code.

Since 1967 and throughout the 1970s, Papert was involved with children programming in Logo, in different research-related educational environments[57], in and out of school, with a few young children, older children, and adults. Thanks to his previous interests in education and thinking, this involvement of Papert consolidated his ideas on the educational significance of computer programming, which first gained widespread dissemination thanks to his book *Mindstorms* (1980).

In that book, Papert focused on his educational ideas, the philosophy behind the use of a programming language such as Logo. I present his educational ideas in section 4.2.2 – "Seymour Papert and constructionism", but for the sake of clarity I'll mention a few here. They include: **syntonic learning** (Papert, 1980, p. 63) or learning in syntonicity with the learner, *i.e.* when learning is related to the sense and knowledge of children about their own bodies (body-syntonic) or consistent with children's sense of themselves, their likes, dislikes, goals, and intentions (ego-syntonic); **procedural knowledge,** or using programming as materials for thinking about and talking about procedures – all kinds of procedures, not just computer procedures (Papert, 1980, p. 223); **microworlds** as "incubators for knowledge", content environments limited so that their principles can be simulated, experienced, and analyzed (Papert, 1980, ch. 5, pp. 120-134); and **debugging** as a way to instill the notion that something that goes wrong is an opportunity "*to study what happened, to understand what went wrong, and, through understanding, to fix it*" (Papert, 1980, p. 114).

In the late 1970s, the Logo audience started to grow, with the appearance of microcomputers: until then, it had been confined to research sites and a few schools. Pilot projects launched in 1980 involved hundreds of children and extensive teacher training. Over the 1980s, Logo use increased and newer versions by different producers were introduced (Logo Foundation, 2000).

During this period the Logo philosophy and programming language benefited from exposure to large number of children and adult users, and Papert focused more on the problem of the different styles of using the tool – the tool↔style problem.

Already in his 1980 book he had laid several thoughts regarding individual patterns and styles of reasoning (*e.g.*, Papert, 1980: ch. 1, "Computers and Computer Cultures", pp. 19-37; styles for

[57] Research sites were located in MIT (USA), Edinburgh (Scotland), and Tasmania (Australia) (Logo Foundation, 2000).

learning "physical" and "intellectual" skills, pp. 104-105; programming with and without structure, pp. 103-104). But at the time he hadn't still fully developed these ideas[58].

By 1990, however, he and Sherry Turkle published a paper on the conflicting styles of programming preferred by different people (Turkle & Papert, 1990). In it, they contrasted the formal, top-down approach to programming and problem-solving, with the informal, tinkering, bottom-up style.

Turkle and Papert's realization was that the culture of computerists, like the culture of science, mathematics and education in general, favored the formal, top-down approach to problem-solving: a problem would be divided into several smaller parts, and then each of those attacked independently, creating a "procedure" that from then on would be a black-box, *i.e.*, an atomic unit, indivisible, to be used and whose internal functioning isn't ever considered.

> *« Structured programmers usually do not feel comfortable with a construct until it is thoroughly black-boxed, with both its inner workings and all traces of perhaps messy process of its construction hidden from view. »*
>
> *(Turkle & Papert, 1990, p. 140)*

While this is a good and efficient approach to solving large problems, they contend that it is not a universal method everyone can feel comfortable with. From the works of French anthropologist Claude Lévi-Strauss, they picked up the word "*bricolage*" to describe the contrasting approach to the formal one (*id.*, pp. 129, 135-143).

> *« Lévi-Strauss used the term "bricolage" to contrast the analytic methodology of Western science with what he called a "science of the concrete" in primitive societies. The bricoleurs he describes do not move abstractly and hierarchically from axiom to theorem to corollary. Bricoleurs construct theories by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials. »*
>
> *(id., pp. 135-136)*

In contrasting these styles of thinking as applied to programming, they point out that actual practitioners of science commonly employ aspects of bricolage, which can put into question whether it should be branded as inferior to formal thinking; and make analogies between bricolage programmers and painters, cooks, sculptors, and writers that start writing without an outline. They stand for the acceptance of this style of thinking, rather than decrying it as inferior: "*Bricolage is a way to organize work. It is not a stage in progression to a superior form*" (*id.*, p. 141).

These ideas became part of Papert's educational philosophy, under concepts as the **proximality** (or closeness) to the object of study, *i.e.*, when the student places herself/himself in the same place as that object, and **demanding permissiveness**, or have children be expected to work hard, but with wide latitude in choice of project (Papert, 1993, p. 124). But its main consequence was contributing to Papert's global view that the current educational system used in schools cannot be reformed. Rather he stands for the need to a complete redesign of how humans teach their children.

> *«* (…) *we imagine the emergence of a science of thought that would recognize the concrete* [rather than the formal] *as its central object. »*
>
> *(id., p. 155)*

These ideas were already being somewhat expressed by Papert in 1980, under a general unease with traditional schooling, *e.g.*:

[58] Cf. Papert's remarks in 1980 (p. 104) regarding approaches to structured programming and his view on the subject in 1990 (Turkle & Papert, 1990, pp. 138-140).

> *« These* [Logo programming] *experiments express a critique of traditional school mathematics (which applies no less to the so-called new math than to the old). A description of traditional school mathematics in terms of the concepts we have developed in this essay would reveal it to be a caricature of mathematics in its depersonalized, purely logical, "formal" incarnation. Although we can document progress in the rhetoric of math teachers (teachers of the new math are taught to speak in terms of "understanding" and "discovery"), the problem remains because of what they are teaching. »*
>
> *(Papert, 1980, p. 205)*

Papert's educational ideas discussed here came into a global form under a new educational philosophy with the name **constructionism**; this was first presented in 1991 (Papert & Harel, 1991). An attempt to synthesize it (albeit lacking the important ideas referred above) is to say that the construction of personal knowledge is particularly supported when the learner is involved in constructing personally meaningful artifacts (Papert, 1999). Constructionism is presented in section 4.2.2, "Seymour Papert and constructionism".

For Papert, the consequence of this line of reasoning is that the current schooling system is inadequate and can't be reformed, but rather changed, radically:

> *« One could not move from a stagecoach to a jumbo jet by making a series of small improvements. Is it possible to go from the school of yesterday to the school of tomorrow by a series of incremental improvements?* (...) *as long as schools confine the technology to simply improving what they are doing rather than really changing the system, nothing very significant will happen. »*
>
> *(Papert, 1998)*

Constructionism is also sometimes referred to, indirectly, as "the Logo philosophy", but part of this logic that I have been presenting is that the Logo programming language itself is just a technology: a step in the right direction, but not a definitive achievement. The name Logo can thus often be found meaning Papert's educational ideas, representing an educational philosophy, not just a programming language. The following quotes show just this, since the first is from 1982, nine years before publication of the book "Constructionism" (Papert & Harel, 1991), and the second, besides being from Papert himself, is from 1999, eight years from the publication of "Constructionism".

> *« Logo is the name for a philosophy of education and a continually evolving family of programming languages that aid in its realization. »*
>
> *(Abelson, 1982 acc. Logo Foundation, 2000)*

> *« The Logo programming language is far from all there is to it and in principle we could imagine using a different language, but programming itself is a key element of this* [Logo] *culture. »*
>
> *(Papert, 1999, p. xv)*

Several modern alternatives to Logo were developed. They can and have been used under the Logo philosophy, and are presented in sections 3.3.4 and 3.3.5. These include:

- Stagecast Creator (formerly Cocoa, formerly KidSim), aimed at development of rule-based animated simulations and games (a description of activities and an education-oriented discussion is found in Lewis *et al.*, 1997);
- Squeak, an implementation of an object-oriented programming language, Smalltalk-80, intended to be highly-portable, by including platform-independent support for color, sound, and image processing; its community originated Squeak Etoys, "*computer environments that help people learn ideas by building and playing around with them*" (Kay, n.d.);

- ToonTalk, aimed at providing a new metaphor for describing programs, by using animated cartoons: there is no "static" code: the entire program is an animation (its relation to the Logo philosophy and culture is debated in Kahn, 2001a).

A final realization is that since a crucial point is the construction of artifacts, this is something achievable not only in programming, but also in **design of artifacts**: "*Constructionism theory suggests a strong connection between design and learning: it asserts activities involving making, building, or programming – in short, designing – provide a rich context for learning*" (Kafai & Resnick, 1996, "Introduction"). Each designed artifact exists outside the learner's mind and must **pass the scrutiny of the physical, cultural, and further constraints of reality**: it becomes something that "*can be shown, discussed examined, probed, and admired. It is out there*" (Papert, 1993, p. 142). These realizations provide the connection between this section, "Learning about thinking, with computers", and the previous, "Learning with computers". And thus, as promised then, a full circle between them is complete.

> *« The argument* (…) *isn't that* (…) *computer programming in general, is unique in providing an environment for learning these thinking skills.* (…) *But it is a very rich environment where these kinds of thinking skills are "exercised" frequently in a natural context. »*
>
> *(Kahn, n.d.)*

3. Introduction to computer programming

3.1. Aims of this chapter

While learning about other subject matters – or about thinking – using computer programming (see sections 2.4.4 and 2.4.5), one certainly learns a lot about computer programming itself. But what if one is the teacher using computer programming? The level of programming expertise can rise very fast above a teacher's programming competence. While this is an opportunity for learning along with the students, the solutions to the problem may end up proving too taxing for both the teacher and the students without resorting to expert help or extensive study on programming techniques, rather than interesting ideas. As was already mentioned in section 2.4.4:

> *« Young children and their teachers very soon hit a ceiling of what is possible for them within reasonable time and expertise constraints. »*
>
> *(Hoyles & Noss, 1999)*

While this is in great part due to the tool↔purpose problem, mentioned in section 2.4.5, solutions to which are still far from ideal, I believe that part of the problem also derives from the teachers' having little notion of the world of computer science: as I stated in section 2.4.4, there is no need to fear learning something about computers while using them to learn something else. But I believe that in most cases teachers that want to use programming lack supportive information, providing an overview of the programming field.

By this I don't mean that teachers should learn theoretical computer science. Rather, I see a computer-using teacher as being able to benefit from some awareness of computer science topics, much as a cook benefits from cookbooks, in order to include them in educational activities that he/she is devising.

In a closer connection to educational practice, a teacher probably feels comfortable using a large set of techniques and materials for developing learning projects and activities with the students: writing, drawing, typing, gluing, painting, acting, role-playing, debating, scripting, building physical models, making school newspapers, etc. I'd like teachers to feel as comfortable using computer programming.

As an example, one can consider the case of a teacher wanting to introduce the subject of olive oil production: it's likely for the learning content to include the raw materials (olive trees, olive oil), the methods of cultivation and harvesting, the operation of oil mills, and so on.

How can a teacher employ computer programming to achieve this? Without a cookbook picture of the main topics in computer science, it's likely that some previous-used programming strategy will be employed. But what if neither teacher nor students manage to picture a path, however hazy, for using one of those strategies in a learning activity?

In this chapter I present several topics from computer programming, along with suggestions for their implementation in an animated programming language, ToonTalk (described in section 3.3.5). My goal is that these can be used just as many people use a cookbook, *i.e.*, after opening the fridge and the kitchen closets, one may pick up a cookbook and think: "let me see, what different dish can I make now, I've got mushrooms, pepperoni, carrots, sweet corn and chicken… flip page, flip, flip, flip… looks interesting, but no, maybe some other time… flip, flip, flip… perhaps… not sure if I make this it will turn out right… well, I think I'll give it a try."

The point here is that I'm not advocating a teacher making sure he/she has used all these concepts with his/her students. Rather, I recommend that a teacher places a checkmark on concepts already included in activities performed with students, in some event, sometime, some year! When preparing himself/herself for teaching, the notion that a new "recipe" still hasn't been tried out may just provide the insight that the planned educational content is requiring. Of course, some educational styles may dispute the concept of planned content itself, but since this thesis aims at

providing a framework for usage of computer programming in preschool-education contexts, I believe the current context of practice should not be ignored.

The initial sections of this chapter deal with presenting an overview of computer programming in general, along with some background on programming languages for children. The programming language and tool employed for the field work upon which this thesis is based, ToonTalk, is also presented. The final section of the chapter presents the cookbook of computer programming topics, their "translation" in terms of both ToonTalk constructs, and sample educational activities in which they can be used.

3.2. Computer programming styles

3.2.1. Sequential vs. Concurrent programming

> *« Programming languages not only furnish us with a means of expression, but they also constrain our thinking in certain ways, by imposing on us a particular model of computation. »*
>
> *(Burns & Davies, 1993, p. 1)*

The above remark is something quickly realized when trying to teach programming skills to non-programmers. For instance, how do programming learners decompose the intended goal or problem? Do they see it as a sequence of steps, carefully planned one after the other, or as several things happening at once? Before providing background information on the path to an answer, I present here a hands-on situation, to clarify the question itself.

Let's suppose the goal is to learn some physics or mathematics, with the help of a billiard ball (Figure 60). There is the need to put the ball in motion and, upon collision against the side of the billiard table, make the ball change direction, in order to make it resemble the behavior of a real ball.

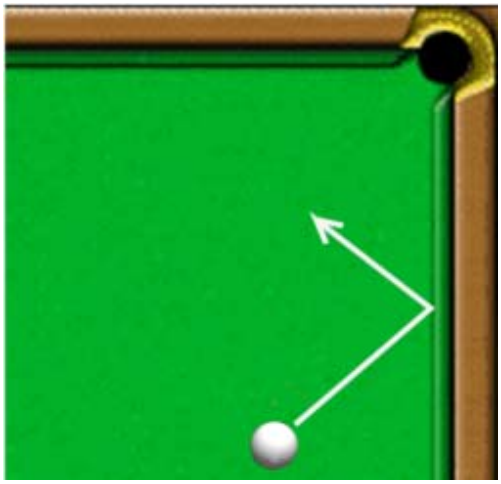


Figure 60 – One billiard ball

In a physics class, this would probably involve a comparison of angles, in math it could be the same thing or the method of updating coordinates, and so on. For the present discussion, the underlying subject is irrelevant. The point is: how do we make a static situation like the one presented in the figure become animated?

One way would be to say: "**move** the white ball in the direction of the first part of arrow, and **when** it hits the side of the table, **change** the direction of movement to that of the second part of the arrow."

In bold, I emphasized the main points of that sentence: an order ("move"), a condition ("when") and another order ("change").

But since computers, deep in their hardware, only execute commands, this entire human decision must be converted into computer orders. That shouldn't be too much of a problem for what is already an order, such as "move" or "change", but what about "when"? How can a computer turn "when" into a command? To help reason on this, I'll resort to Papert's notion of body-syntonicity[59]: I'll imagine myself as the ball receiving that "when" command.

There are two main approaches to perform this "when"[60]:

1st approach: "I'll move a little bit, see if I'm touching the side of the table, move a bit more, and I'll keep doing these two things in turn until I reach the side of the table. Then I'll change my direction."

2nd approach: "I'll close my eyes and keep moving. Eventually my shiny surface will feel the collision against the side of the table, and then I'll change my direction."

[59] See sections 2.4.5 and 4.2.2.

[60] Many other approaches could be imagined, that's the beauty of programming: one can reason around a problem in any desired way and try out hypotheses, to see how they work. But these approaches serve the purpose of rendering clear the intended distinction in programming styles for the present section.

My main point here is that, in the first approach, I (the ball) am doing it all, one bit at a time: move/check/move/check/move/check… and eventually, change direction, and then again resume: move/check/move…

In the second approach, I (the ball's "inner brain") am only concerned about one thing: making my ball-body move. I count on something else to let me know it's time to act differently (in this case, my "ball surface", but it could also be a contact sensor, a bumper-collision detector, etc.).

In terms of programming, these approaches represent the distinction between sequential programming (1st approach) and concurrent programming (2nd approach): in sequential programming, one's reasoning is a neat (or sometimes not so neat) sequence of orders & verifications; in concurrent programming, we're conducting and orchestra of several separate things that are occurring simultaneously[61] (and that orchestra can produce both great music and musical chaos).

I'll now present these approaches to the problem side-by-side:

Sequential approach	Concurrent approach		
1. Move forward a bit. 2. Am I colliding? If not, go back to step 1. If I am, change direction of movement and begin it all again, from step 1.	**Ball "brain"**	**Ball "body"**	**Ball "skin"**
	I'll put the skin looking for touch, and the body moving; then I'll just be waiting for the skin to tell me of a collision. At that time, I'll tell the body to change direction.	I'm just moving forward until someone tells me to stop or change direction.	I'm just checking to see if I'm touching something, and then I'll let the brain know and keep on checking.

Table 4 – Sequential and concurrent approached: moving one ball

Different readers will probably find one of these approaches simpler than the other; that's the whole point, and the reason why I initiated this section with that Burns & Davies' quote. While in all programming there is the freedom to elect how to solve a given problem, the features of the programming language do constrain one's style of reasoning. This section focuses on the presence or absence of the concurrency feature.

Traditional programming languages were created under the sequential model; after all, traditional computers execute one instruction at a time, sequentially, so when the concept of programming languages itself was still fairly new, it would be hard to translate a language that was concurrent into the proper sequence of instructions for the computer.

But there are times when the situation in hand requires some complexity in reasoning, in order to be translated into a programming language that employs the sequential model.

[61] Conceptually. In reality, the computer will be rapidly switching from task to task; but given enough computational power, this can appear to be simultaneous. Only computer systems with more than one execution unit can really do more than one thing at the same time, but that's at a different level, disconnected from this abstraction.

To make clear in what situations that may happen, I'll consider the following case: on that billiard there were two balls to be moved, instead of one (Figure 61).

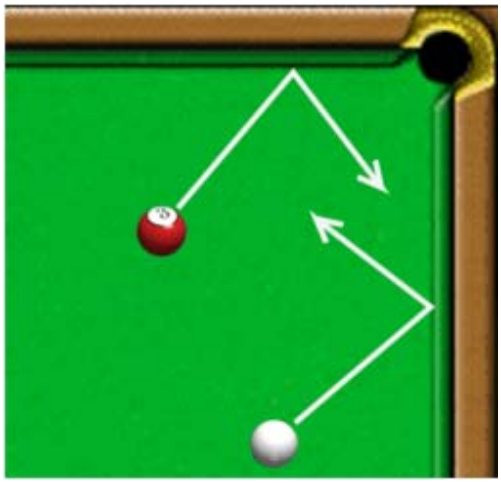


Figure 61 – Two billiard balls

The sequential approach runs into trouble, now, because the two balls are supposed to be moving at the same time. If I command them separately, the global result won't be simultaneous movement:

"**Move** the white ball in the direction of the first part of its arrow, and **when** it hits the side of the table, **change** the direction of movement to that of the second part of its arrow. **Move** the red ball in the direction of the first part of its arrow, and **when** it hits the side of the table, **change** the direction of movement to that of the second part of its arrow."

This sequence of commands will only work out right if BOTH sentences are executed simultaneously. Otherwise, if one is executed after the other, the white ball will complete its course, and only then will the red ball start to move.

But in order to implement this using the sequential model, it's necessary to switch back and forth between both balls, moving them a bit and checking to see if there is a collision or not. Such code is much harder to program, complex, error-prone, and difficult to scale[62].

<table>
<tr><th rowspan="2">Sequential approach</th><th colspan="3">Concurrent approach</th></tr>
<tr><th colspan="3">White ball</th><th>Red ball</th></tr>
<tr><td rowspan="2">1. Move white ball forward a bit.
2. Is the white ball colliding? If not, proceed with step 3. If it is, change its direction of movement and proceed with step 3.
3. Move the red ball forward a bit.
4. Is the red ball colliding? If not, begin it all again from step 1. If it is, change its direction of movement and begin it all again from step 1.</td><td>Ball "brain"</td><td>Ball "body"</td><td>Ball "skin"</td><td rowspan="2">Same thing.</td></tr>
<tr><td>I'll put the skin looking for touch, and the body moving; then I'll just be waiting for the skin to tell me of a collision. At that time, I'll tell the body to change direction.</td><td>I'm just moving forward until someone tells me to stop or change direction.</td><td>I'm just checking to see if I'm touching something, and then I'll let the brain know and keep on checking.</td></tr>
</table>

Table 5 – Sequential and concurrent approached: moving two balls

As the example above shows, when the goal of programming is modeling separate "entities", which behave separately, it's simpler to picture those entities separately in programming also, rather than ensuing with this process of intercrossing their tasks.

[62] One could consider the task of getting all 15 balls on the table, moving at the same time, just to have a better notion of this complexity. In many cases, this complexity is overcome by using techniques such as iteration, in which the problem is considered with 2 or 3 balls and code is written to cycle them performing the same procedures – doing this requires some planning, but then it can be scaled quickly to 1,000,000 balls or more, if necessary.

> *« Tom Kilburn and David Hoarth pioneered the use of interrupts to simulate concurrent execution of several programs on the* ***Atlas computer*** (...)*. The early multiprogramming systems were programmed in assembly language without any conceptual foundation. The slightest programming mistake could make these systems behave in a completely erratic manner that made program testing nearly impossible. By the end of the 1960s multiprogrammed operating systems had become so huge and unreliable that their designers spoke openly of a* ***software crisis****. »*
>
> *(Hansen, 2001, p. 3)*
>
> *« This* ***sequential paradigm*** *does not match the way the real world works: people and animals act in parallel, objects interact in parallel. As a result, many activities can not be modelled in a natural way with sequential programming. The ideal solution is to use a* ***concurrent*** *or* ***parallel*** *programming language* (...) *»*
>
> *(Resnick, 1990)*

Of course, sequential techniques were developed to simplify this process, usually by running cycles of identical actions, building new levels of abstraction; and object-oriented programming also contributed to such simplification. But it is always a complicated matter adding new behaviors: a cue may have to be included, to move the balls; these may have to lose velocity as they go, so that they eventually stop; they may have to collide between themselves while moving at different speeds, so the reaction isn't the same as bumping into the side of the table; etc. All these different situations, even if they're not planned in advance, can be included in the concurrent model as extra pieces, like adding new bits to a toy house or castle, built with a construction set; one doesn't have to tear down the walls to include a chimney or modify the roof – of course, there are still problems, both of sequence (the roof will have to go out in order to change a brick in the wall) and of relationship between parts (one cannot add a meter-long cloud coming out of the chimney without risking toppling the house!). But the point is that concurrent programming provides a nicer method to describe entities that possess separate behaviors. ToonTalk, the language used in the field work supporting this thesis (*vd.* section 3.3.5), is an example of a language designed for concurrency.

Computer scientists and engineers came across this situation quite early in computer history: since computers are real objects, they are composed of several separate components that need to be controlled nearly simultaneously – even if there is only one processor.

> *« The development of concurrent programming was originally motivated by the desire to develop reliable operating systems. From the beginning* (…) *it was recognized that principles of concurrent programming "have a general utility that goes beyond operating systems" – they apply to any form of parallel computing*[63]*. »*
>
> *(Hansen, 2001, p. 4)*

Even in cases when there are separate entities to command, however, it is not assured whether concurrent programming is simpler for everyone to learn and use, or even for most people. Most programming teachers (including myself) frequently encounter the programming students' confusion when they first face the notion of concurrent programming. Is this a contradiction?

[63] According to Hansen (2001), this quote is from himself: Hansen, Per Brinch (1971). *An Outline of a Course on Operating System Principles*, in C. A. R. Hoare and R. H. Perrot, eds., 1972, "Operating Systems Techniques", Proceedings of a Seminar at Queen's University, Belfast, Northern Ireland, August-September 1971, 29-36, article 6, Academic Press, New York, NY, USA.

Not necessarily. Usually, students are introduced to programming with the traditional, sequential, programming. Going from their experience with it to concurrent programming involves a change of thinking style, so it is little wonder that such change is not smooth, to say the least.

> *« It is possible that some students had trouble decomposing problems into concurrent subtasks precisely* ***because*** *of their expertise in traditional programming. »*
>
> *(Resnick, 1990)*

The above remark comes from a study on the relative ease or difficulty of sequential vs. concurrent programming, done by Mitchel Resnick (Figure 62), while developing a version of the Logo programming language for children, called MultiLogo.

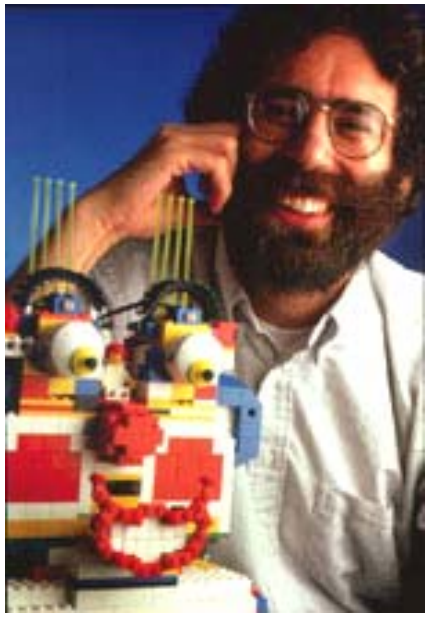


Figure 62 – Mitchel Resnick
From: http://alumweb.mit.edu/opendoor/20030 9/images/resnick.jpg

In that paper, Resnick starts to address this theme by noting previous research reports on novice programmers, who while using traditional sequential languages, often assume parallelism (another way of referring to concurrency) where none exists. The natural question is thus whether such novices wouldn't have found concurrent programming more natural and intuitive to learn.

However, in his research for that paper, Resnick worked with children (4th and 5th graders) that had already one full year of (sequential) Logo programming experience, so some of his conjectures would need further research support, by conducting activities on concurrent programming with children that haven't learned sequential programming beforehand[64].

One particular case, however, was close to the desired research situation of no previous contact with sequential programming, and constitutes a promising indicator:

> *« I asked NL and FB[65] to explain how they would coordinate a "real-life" situation in which two people execute tasks simultaneously, and (at least initially) they both stuck with their time-slicing approaches.* (…) *NL's and FB's initial "solutions" to the real-life problems are an indication of just how strong their sequential-thinking models are.*
>
> (…)
>
> *BW, who worked together with FB during FB's final session,* (…) *thought a good student, had limited Logo programming skills and had never seen MultiLogo. Nevertheless (or perhaps as a result), her instincts on agency decomposition were quite good.* (…) *It is possible that BW's lack of Logo skills made it easier for her to embrace the new paradigm. BW would have had great difficulty writing the program to implement her idea, but she grasped the overall concept of concurrent programming very quickly. »*
>
> *(Resnick, 1990)*

[64] This would be tricky to do in a language such as Logo, which is sequential at heart.
[65] NL, FB and BW are initials of three children Mitchel Resnick worked with.

Another promising indicator are his reports on children's programming errors (bugs) due to what Resnick calls "assumption of excessive parallelism" (*id.*, *ibid.*). He further associates this to previous research:

> *« Soloway et al.*[66] (…) *found that 34% of the students in an introductory Pascal course assumed a type of parallelism in the execution of a* `while` *loop: they thought that the program continuously monitored the while test condition as it executed the body of the loop* (…)*. Similarly, Pea*[67] (…) *found that more than half of the students in one high-school BASIC course expected* `if-then` *conditional statements to act as demons, executing* ***whenever*** *the conditional predicate became true. »*
>
> *(id., ibid.)*

Such problems of assumption of concurrency are commonly reported. For instance, a 2003 summary of research on misconceptions of students about programming constructs includes (focusing on high-school students):

> *« Loop Construction* (…) *Misconception D* (…) *The variables in the* `while` *condition are* ***continuously monitored****. This interpretation is consistent with English "while" as in "while the highway is two lanes, continue north."*
>
> (…)
>
> *Flow of Control Misconception A* (…) *Different lines of a program are interpreted at the same time* ***in parallel****. The students incorrect believe what will happen later in a program influences what happens earlier.*
>
> *Misconception B* (…) *The students get* (…) *assignments for swapping the values of two variables in the wrong order, probably due to a* ***lack of regard for the sequential nature*** [of the statements].
>
> *(Lam, 2003)*

Regarding these common misconceptions, I can say that I was also one learner of programming that initially thought that the `if` statement in BASIC was a permanent statement. I don't have the faintest idea of what I was trying to achieve in one of my first programs, but I clearly remember the astonishment at it not working as intended and the explanation – and realization – that there was a sequence of execution, that I wasn't really speaking to the machine but just feeding it a sequence of commands to be interpreted, one at a time, and forgotten. So much so that it's still present in my memory, 20 years later. The program itself, I've long forgotten.

I will now address the ways by which concurrency can be achieved in a computer and expressed in a programming language.

The first and simplest solution is to have more than a processor in the computer, *i.e.*, more than one component capable of executing orders. But that, while possible, would again drag the programmer from caring about the problem to caring about the machine: what if there aren't enough processors to assign a concurrent task to each one, for instance? Should the number of concurrent tasks be limited to the number of available processors? What if one of the processors fails? Should the program need to be re-written, in order to run with a different number of processors? Under the

[66] Soloway, E., J. Bonar, J. Greenspan, K. Ehrlich (1982). *What Do Novices Know About Programming?*, in "Directions in Human-Computer Interactions", edited by Ben Shneiderman and Albert Badre, ISBN 0893911445, Ablex Publishing Company, Norwoord, NJ, USA.

[67] Pea, Roy D. (1986). *Language-independent conceptual 'bugs' in novice programming*, in Journal of Educational Computer Research, vol. 2 (1), pp. 25-36, ISSN 0735-6331, Baywood Publishing Company, Inc., Farmingdale, NY, USA.

logic of programming languages increasing expressiveness and abstraction from the particularities of the underlying hardware, concurrency in programming should be above all a method of reasoning over a problem, and having to depart from that level to the hardware level would be a step back in time and evolution.

An answer to this concern came from Edsger Dijkstra (Figure 63), who wrote:

> *« In the literature one sometimes finds a sharp distinction between "concurrent programming" – more than one central processor operating on the same job – and "multi-programming" – a single processor dividing its time between different jobs. I have always felt this distinction was rather artificial and therefore confusing. In both cases we have, macroscopically speaking, a number of sequential processes that have to co-operate with each other, and our discussions on this co-operation apply equally well to "concurrent programming" as to "multi-programming" or any mixture of the two. What in concurrent programming is spread out in space (e.q. equipment) is in multi-programming spread out in time: the two present themselves as different implementations of the same logical structure* (…) *»*
>
> *(Dijkstra, 1965, "Concluding remarks")*

Thus, the point is that it doesn't matter how many processors are available in the computer running the program. If there are enough to conduct separate operations, they can be used concurrently. But if there aren't enough processors, the ones available – even if it is a single one – can simply move back and forth between tasks, executing a little bit of each at a time, thus providing an illusion of concurrency.

Figure 63 – Edsger Wybe Dijkstra, 1930 – 2002
From: http://www.cs.utexas.edu/users/EWD/EWDwww.jpg

In that same paper, Dijkstra presented a full basis for what became the most common style of concurrent programming, used by millions of programmers worldwide. Most of the concepts employed in concurrent programming were first presented in that paper.

The most important of those concepts is the notion of each separately-executing task as an entity in itself. Dijkstra called such tasks a **process**, but other names that are commonly used for this notion, under different programming paradigms are **task**[68], **thread**[69], **agent**[70], and **actor**[71]. For convenience, I will use the term "process" throughout this section.

This notion of a program as a collection of separate processes has a direct consequence: if they are separate, but something is to be achieved together, they must cooperate. This, in turn implies that they need to **communicate**, but such communications are also independent (*i.e.*, asynchronous): one of Dijkstra's most important realizations in this regard is that since processes are independent, conceptually each process is working alone within a single, private, virtual computer. And so, since one virtual computer and its process compose an autonomous entity, there is no way to assume relative speeds for execution of processes. This concept is known as **speed independence**.

> *«* (…) *apart form the (rare) moments of explicit intercommunication, the individual processes themselves are to be regarded as completely*

[68] *E.g.*, in the Ada language (Burns & Davies, 1993, p. 160).
[69] *E.g.*, in the Java language (Brookshear, 2003, p. 252). Threads are often considered a special case: different lines of execution inside a process, but such distinctions aren't relevant for this discussion.
[70] Agent-oriented programming (Shoham, 1993).
[71] Actor-oriented programming (Hewitt, 1976; Agha & Hewitt, 1986).

> *independent of each other. In particular, we disallow any assumption about the relative speeds of the different processes. (Such an assumption* (…) *could be regarded as implicit intercommunication.) »*
>
> *(Dijkstra, 1965, "Loosely Connected Processes")*

To understand this concept better, I propose looking at the two billiard balls again, from Figure 61. The concurrency could either be the model used is the proposed one with three processes ("brain", "body" and "skin") or the sequential one for the single-ball case, only running as two separate processes: one for the white ball, another for the red ball.

What the speed-independence concept means is that one cannot assume, for instance, that for each nudge of the ball there will be a verification of collision; or that for each nudge of the white ball there will be an identical nudge of the red ball. So, theoretically, the white ball could in fact be moving without the red ball moving at all. But this isn't necessarily a bad thing: the time-slicing that takes places underneath must try to address all processes adequately[72]. So any discrepancies will be a reflection either of the actual computer operation (other running software, hardware behavior) or of differences between the processes conditions (for instance: if the program for a ball stops working, for some reason, the other may keep moving).

> *« The consistent refusal to make any assumptions about the speed ratios will at first sight appear to the reader as a mean trick to make things more difficult than they already are. I feel, however, fully justified in my refusal. First, we may have to cope with situations in which, indeed, very little is known about the speeds. For instance* [some parts may be dependant on the human operator, others may be idle]. *Secondly – and this is much more important – when we think that we can rely upon certain speed rations we shall discover that we have been "penny wise and pound foolish". It is true that certain mechanisms can be made simpler under the assumption of speed-ratio restrictions. The verification, however, that such an assumption is always justified is, in general, extremely tricky and the task to make, in a reliable manner, a well-behaved structure out of many interlinked components is seriously aggravated when such "analogue interferences" have to be taken into account as well. »*
>
> *(Dijkstra, 1965, ibid.)*

If, for some reason, synchronization is necessary (for instance, to check for a collision after **every** nudge of the ball, to avoid detection from occurring too late in the movement), then communication must be used to ensure such behavior. In the two-ball example, this would mean one of two things:

- the ball body would ask for "permission" to move after each nudge, and such permission would be granted only after the skin asserts that there hasn't been a collision;
- a synchronizing mechanism, such as clock or another timing device, is checked by all processes, to ensure that they all complete their collision-detection tasks before moving another bit; and that they all complete their tiny moves before proceeding with collision-detection[73].

[72] A concept known as fair scheduling; this won't be addressed in this thesis for lack of relevance as a computer-science concept for preschool.

[73] This method of synchronization is technically known as a "barrier" (Tanenbaum, 2001, pp. 123-124).

This way of synchronization renders clearer the metaphor I employed earlier in this section, about concurrent programming being more like conducting an orchestra. Another way to put it is like managing people or refereeing a football[74] match.

One should be aware, however, that in concurrent programming there is no need for a process (or even the programmer) to act as an actual "conductor", "manager" or "referee". Rather, each process can have its own, specific rules, and act in absolute independence. Metaphors for such programs are ants harvesting food, water molecules forming clouds, bacteria reproducing, cars moving in roads. To understand how such a program may generate a meaningful behavior, one must keep in mind that these metaphors are themselves examples of self-organizing behaviors from rules applied to independent particles/actors/processes. An entire scientific field is devoted to the analysis of such kinds of self-organization, known as the analysis of cellular automata[75].

Using communication, a concurrent program for the ball could resemble this:

Billiard ball		
Ball "brain"	**Ball "body"**	**Ball "skin"**
1. "Hey, body, here's a nudge-length move ticket! Do it and let me know." 2. Now I'll wait for the move to complete. 3. "Hey, skin, can it let me know if we're colliding?" 4. If the skin says we're colliding, I'll change the direction before resuming movement, from step 1.	I'm waiting for a ticket, so that I can move forward a bit.	No need to bother checking for collision unless the brain asks me to. (I just hope nothing collides against us!)

Table 6 – Concurrent programming: moving balls with communication

Taking a hint from constructionist theory, I have included two contradicting elements in the above example, to provide for further discussion now:

- in the ball's "brain" process, I've included sequence numbers, to emphasize its internal sequential structure;
- in the ball's "skin" process, I've included another anthropomorphism – a worry – to shed some doubt on the correctness of this particular programming model for a ball.

The sequential aspect of the brain process could easily have been included in the remaining processes as well. It was included here to make the point that concurrent programming doesn't imply, necessarily, the total absence of sequential reasoning: after all, one frequently engages in sequential reasoning. Rather, concurrent programming provides mechanisms to knowingly specify concurrent behavior.

[74] Soccer (not that the distinction matters, in this thesis).

[75] The most well-known case is John Conway's "Game of Life" (Callahan, 2000, presents a description, background information and a playable applet), but this concept first emerged in 1949, in John von Neumann's paper "Theory and Organization of Complicated Automata" (in Burks, Arthur W., 1966, *Theory of Self-Reproducing Automata*, pp. 29-87, University of Illinois Press, Urbana, IL, USA). A large recent (2002) volume in this field is Stephen Wolfram's "A New Kind of Science", ISBN 1-57955-008-8, Wolfram Media, Champaign, IL, USA.

The skin's "doubt", above, is meant as an example of how rich in opportunities for reasoning over a problem the concurrent model is. I'll provide a few here, **which are intentionally self-contradictory**.

- Considering that collisions occur only when it moves itself, this ball wouldn't react when it's standing still and some other ball collides with it, perhaps after all the previous model, which allows the skin to check for contact as often as possible, is better;
- or perhaps this model is almost acceptable, just that it is the body that needs the tickets to go ahead, the skin can be left to operate as previously (as often as possible);
- but if all balls on the table are identical, perhaps this last dictatorial method ("you move, stop and report; you check and report; you move... ") works, since other balls that can hit this ball will recognize the collision and respond accordingly;
- however, if we're working on a physics model, then the ball that is standing still needs to respond to collisions from other balls, too, so perhaps the dictatorial method isn't suitable after all;
- a completely different approach would be for balls to check for collision only when moving, but "warn" other balls against whom they collide, so that they can take the appropriate action, perhaps a global adequate behavior for billiards emerges from all individual ball behaviors[76].

Regarding the implementation of concurrency with actual programming languages, the main distinction to be made is that some languages have been developed from the ground-up as concurrent (*e.g.*, ToonTalk, described in section 3.3.5), while others have had concurrency features added to them (*e.g.*, Logo, described in section 3.3.4, pp. 139-146).

To render clear the point for this distinction, I need to propose a look at languages regarding their design goals, which can be seen related to the kind of system they were designed to tackle: **transformational** vs. **reactive** system (Shapiro, 1989, p. 424). Languages and programs can thus be considered transformational or reactive.

A transformational program is one thought of with a beginning and an end: its aim is to produce a final result, for a given set of initial values or conditions. Conversely, a reactive program is intended to maintain some interaction with its environment, and whether or not that includes producing a "final" result is irrelevant (*id., ibid.*).

Looking at concurrency under this perspective, I will not consider the particular case of sequential programs that are made concurrent just for optimization of performance, be it achieved by sharing multiple processors, by dividing a problem into identical tasks (*e.g.*, partial matrix operations), or by other similar operations[77]. Rather, I am considering those programs that are designed, from a reasoning point of view, as concurrent. These include programs that employ a cooperation of distinct processes with specialized, distinct goals, and also programs that use multiple interacting copies of identical processes – again, the emphasis is on programs that employ reasoning in a concurrent way.

In my view, concurrency differences between transformational and reactive languages are felt at the traditional levels of **inter-process communication**, and **process synchronization**, but also at the level of **program organization**. This is not due to language features imposing a specific

[76] Mitchel Resnick employed the language StarLogo (based on his concurrent MultiLogo), "*designed for situations with hundreds of thousands of objects acting in parallel*" (Resnick, 1997, p. 46) to explore with children the emergence of patterns and behaviors in several models, from huge numbers of concurrently-executing processes with simple rules. Many were related to biology, but not all: one process per termite, one process per car in traffic, per tree in forest, etc. (*id.*).

[77] Shapiro classifies such use of concurrency techniques as "parallel programming" (1989, *ibid.*), but others have used this term differently (*e.g.* Resnick, 1997).

organization; rather they favor one over another. A programmer can use a transformational language in a reactive way, and a reactive language in a transformational way, but it's not always a simple endeavor.

Transformational programming languages lend themselves to programs where central processes command others (Figure 64).

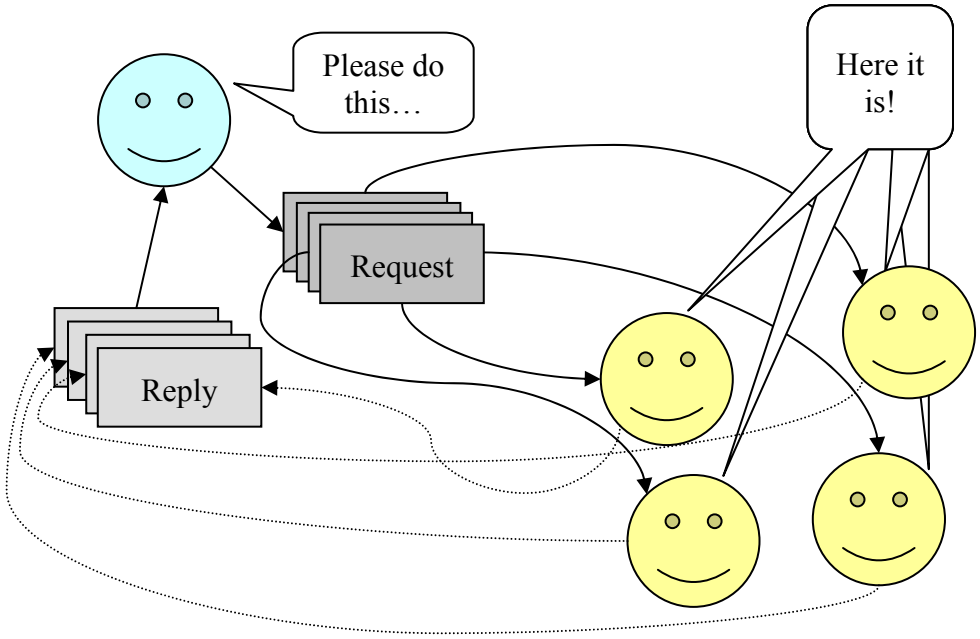


Figure 64 – Typical concurrent transformational program

Reactive programming languages, on the other hand, tend to distribute responsibilities, and wait for events to attend to (Figure 65), and command is often decentralized.

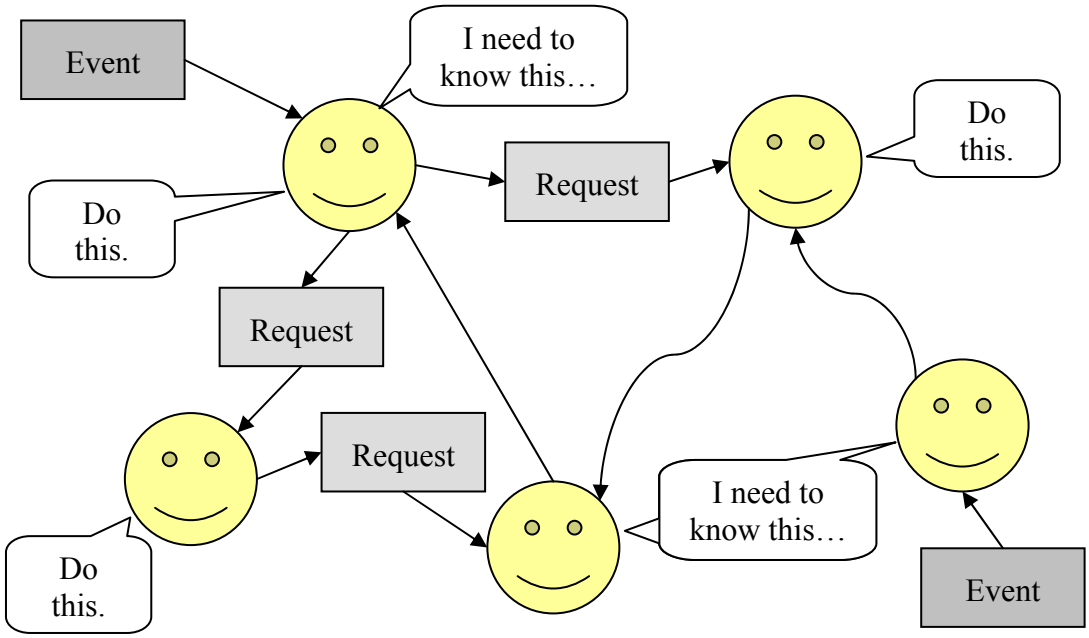


Figure 65 – Typical concurrent reactive program

Regarding the communication and synchronization between processes, both revolve around three different needs:

- data and commands from a process must be able to reach another;
- some processes need coordinate the timing for execution of their actions with others[78];
- there are times when a process must not be interrupted, lest its work become meaningless.

In order to render clear the importance of these topics, I'm going to return to the billiards table, as a way to render these problems more concrete.

This time, the two balls will collide. Supposing the subject matter being explored is Newtonian physics[79], one simple case would be to consider the balls' mass to be identical, with no attrition (*i.e.*, velocity is constant), full elasticity, and no diameter. This would thus be a matter of combining velocities (Figure 66).

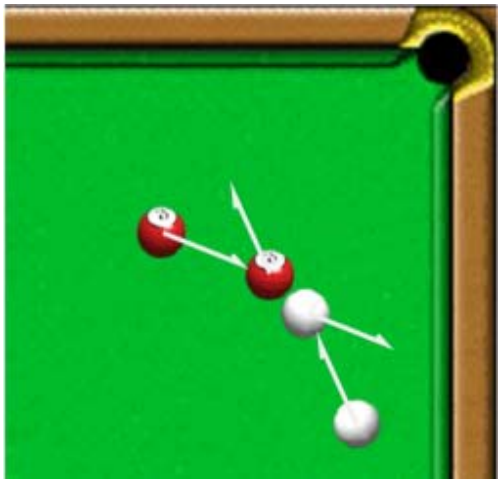


Figure 66 – Colliding billiard balls

Assuming that each ball has one or several processes controlling its behavior, when collision is detected, there is the need for each ball to determine the velocity of the other ball, to decide the new direction.

In such software-based simulations, usually velocities are found as either global properties of the system (*i.e.*, a huge system library with the velocities of all balls) or as local properties of each ball. This latter case is the one which I'll use here.

The white ball will have to do two separate actions: 1) get the red ball's velocity; 2) combine it with its own. The red ball will have to do the same thing. If all goes well, the balls will depart the collision spot following the arrows presented in Figure 66.

But things could go quite wrong. Following the notion of speed independence, mentioned previously, it isn't possible to assume that each ball will do those two steps in nice order. Let one assume this situation:

Time	White ball	Red ball
1	Detected collision	Detects collision
2	Gets velocity of red ball	
3	Updates own velocity	
4		Gets velocity of white ball
5		Updates own velocity
6	Resumes movement	Resumes movement

Table 7 – Concurrency: synchronicity problems

[78] *E.g.* by specifying precedence, priority, and mutual exclusion (Hansen, 2001, p. 22) or coincidence (using the methods referred in footnote 86, at the bottom of page 113).

[79] This example was inspired by an actual event at London's Institute of Education in 2002, when Yishay Mor, Gordon Simpson and I were trying to create a simple high-school physics example using ToonTalk.

After those 6 time slots have elapsed, it's fair to say that the assignment of computer power was fair: the processes in charge of each ball manage to run four times each. However, due to speed independence, this situation causes a strange behavior to occur (Figure 67).

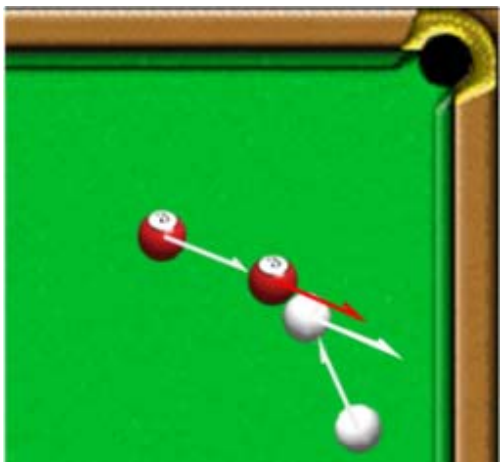


Figure 67 – Wrong result of collision

It looks as if the white ball had crashed into the red! The reason is that the velocity for the red ball (the red arrow in the figure) was wrongly calculated. What happened was this: during time-slots 1-3, the white ball did everything it need to, including the update of its velocity. So, when the red ball finally got a chance to get the white ball's velocity, in step 4, it was no longer the original velocity, heading towards the red ball, but the new one, moving away from it!

So, in time-slot 4, from the perspective of the red ball, it didn't collide head-on with the white ball: it crashed on it from behind, going at a slightly higher speed. The result is thus two balls moving in the same direction at identical speeds[80].

A similar problem can occur if the red ball does everything before the white ball. In fact, the collision will ONLY turn out fine if both balls read each other's velocity before changing their own.

The problem exists because independent processes (the balls) must share states (in this case, their velocities and the event of colliding). Getting information on a shared state and acting based on it is a source of **synchronization problems**. In order to solve this and related problems, concurrent systems offer synchronization methods, usually of the sorts presented in the list that follows this paragraph. As can be seen from that list, the methods for solving the problem of synchronization include methods used for solving the other mentioned problem: **communication**:

- *Lock-out methods*: the access and manipulation of information about the shared state is attributed to a single process, and others have to wait until the shared state becomes free (typical examples from computer science are called semaphores[81], condition variables[82] and monitors[83]).
- *Message-passing methods*: rather than accessing a shared state, processes pass copies of information among them, and are free to do whatever they desire with their copy of the information[84].
- *Remote invocation of methods*: rather than accessing a shared state, processes provide copies of information and request other processes to act on their behalf[85].
- *Parallelize execution*: if necessary, different parts of concurrent processes can be forced to start coincidentally in time, and not to advance beyond a certain point until all involved parts are complete[86].

In section 3.4, I present a more complete description of several of these methods[87], including ToonTalk examples of their implementation and brief suggestions for educational use.

[80] In some cases (such as ToonTalk programming), the behavior can be even stranger, because the balls can overlap a bit by the time the collision is detected. If this overlap isn't undone, a new collision may be detected immediately after the first, resulting in a reversal of movement!
[81] (Dijkstra, 1965)
[82] *Id.*
[83] A method invented by Per Brinch Hansen in 1973, but its complete description (with several developments) was first presented by Tony Hoare (1974).
[84] A method introduced by Hansen (1969) and later expanded by Hoare (1978).
[85] This technique is known as "remote procedure calls", and was first introduced by Hansen (1978)
[86] In computer-science terms, such techniques include parallel statements (Dijkstra, 1975), barriers (Tanenbaum, 2001, pp. 123-124) and synchrons (Turbak, 1996).

3.2.2. What You Want vs. What To Do: constraint programming vs. procedural programming

Regarding this second part of the major topic "Computer programming styles", I found the opening quote of the previous section to be particularly fortunate by having employed the word "constraint." I'll present it here again, to spare the reader the trouble of browsing back to page 101.

> *« Programming languages not only furnish us with a means of expression, but they also constrain our thinking in certain ways, by imposing on us a particular model of computation. »*
>
> *(Burns & Davies, 1993, p. 1)*

In that section (3.2.1), I presented one way in which this could occur, by contrasting sequential programming against concurrent programming. In this section, I present another distinction, by contrasting **programming with constraints**, around intended goals, against **programming with orders**, focusing on specific goals. The term "procedural", used in the title of this section, thus represents its semantic sense; it is not a synonym for "functional programming" (presented in section 2.2.6) as it is sometimes used.

For a programming language, being sequential or concurrent, as presented in the last section, and being procedural or constraint-oriented, as is being presented in this section, aren't necessarily advantages or hindrances: it all depends on the purpose for which the language is intended, on the context of its use – and also on the thinking style of the user (programmer). Programming with constraints provides the ability to express one's reasoning in a way that is quite different from traditional programming, and that, in itself, is a powerful thing.

> *« A significant motivation for programming language research is to find clean abstractions and conceptual frameworks that enable us to understand and reason about complex computational phenomena in a simple way. »*
>
> *(Saraswat, 1993, p. xxxiii)*

The metaphor I elected to use for this section is the common task of moving furniture or other objects through a doorway (Figure 68).

Figure 68 – Moving furniture involves goals and constraints

Expressing such a task in terms of traditional programming models implies that the programming process is focused on **how** the task is performed – what to do. Specifically, one would seek to determine the starting position of the object, its size and format, the distance to the doorway, the size of the doorway, etc. Then one would act upon those data, to achieve the goal of transposing the door.

Here's a typical approach to the resolution of the problem:

- assemble data on object and door dimensions and positions;
- determine need for re-orientation;
- perform necessary re-orientation;
- go through the doorway;
- determine need for re-orientation before setting down the object;
- set it down.

[87] Sections 3.4.10 – "Parallel/Concurrent execution", 3.4.11 – "Parallel/Concurrent statements", 3.4.12 – "Message passing / Communication channels", and 3.4.18 – "Clients and servers".

These operations, of course, imply smaller problems, such as how to turn it around, how to pick it up, drop it, etc. The overall goal of taking the object through the door is kept in mind, but it isn't the focus of the programming task.

Rather, in constraint programming, one focus on **which and what** limitations define the task – that what is wanted. The programming process is focused on specifying the details of the problem, as finely as possible, and by that process to define how it should be solved. Here's a constraint approach to the resolution of the problem by defining it more precisely:

- the object's starting position is[88] on the outside; } The object must
- the object's ending position is on the inside; } move.
- the object's path must be through the doorway; } The object must fit
- the object must not be disassembled; } the doorway.
- the object area facing the doorway must be smaller than the area of the doorway.
- the width of the object are must be less than the width of the doorway; } The object may have to be rotated.
- the height of the object must be less than the height of the doorway;
- the object must be laid down on its base; } The object may
- the base must be leveled with the floor; } have to be rotated.
- the object must not be dropped;
- the place where to set down must be clear. …

I propose that the reader compares his/her own thinking styles while reading the previous two sequences. As one's reasoning progresses along the sequences, the mindframe towards the overall problem (carrying the object inside) is likely to be quite different. Things that are explicit in one model become implicit in another; things that are the major concern in a model become less relevant in another. The reason I am not rendering concrete these differences is because they're most likely to differ from person to person; but I expect some sort of contrast to be deep enough to be clearly felt by most people.

One of the most important distinctions between these two models is the handling of data. In procedural programming (the former sequence) data is collected, stored and processed, flowing through the program, which is composed of operations to be performed on the stored data. By contrast, in constraint programming (the latter sequence) the program flow involves a continuous refinement of constraints, eventually leading to a conclusion: the data to which the constraints apply is almost ignored in the sentences. In them, data seems almost "internal" to each constraint.

One can imagine these sentences being written and read, gaining a better conception of the overall problem, without actually getting hold of actual data values; in other words, as if it were not necessary to have the actual data values for the program execution to proceed.

This last remark derives from the fact that constraint languages are logic programming languages (Shapiro, 1989), which are geared towards processing and combining information in logic relationships, rather than towards processing data values (processing data becomes a simple implementation of the logic structure attained through those languages). The class of constraint languages is thus more fully addressed as **constraint logic programming languages**[89].

[88] On this and the following line, my usage of the verb "is" means to impose a constraint on the mentioned positions; it is not a mere declaration.

[89] There have also been attempts to introduce constraints in traditional programming, under the general name of Constraint Imperative Programming: a summary of this was included in the PhD thesis of Michael Travers (1996, pp. 52-53).

To render clearer these somewhat cryptic remarks, I'll resort to the following concrete case: carrying a table into the house (Figure 69).

Figure 69 – Moving a table

In a procedural programming language, the new position of the table being moved could be calculated by using a formula such as this:

```
NewTablePosition=CurrentTablePosition+StepLength
```

This statement would be an order for the computer to get the currently stored values of the variables (or functions) "`CurrentTablePosition`" and "`StepLength`", add them, and store the result in the variable "`NewTablePosition`"[90].

But in a constraint language, this does not represent an order; rather, it states a relationship between these three variables, so that any of them can be computed from the others.

This means that if at a given moment the program issues the question:

```
Is the NewTablePosition beyond the door?
```

A procedural language will need the door position and the value of the variable NewTablePosition[91]. With these values, a new "Yes" or "No" value is calculated, and that "Yes" or "No" is used for proceeding with the program.

Instead, a constraint logic language does not need to know the values to proceed: rather, the result of that question will be something like:

```
"The result of comparing the door position with
CurrentTablePosition+StepLength."
```

Figure 70 – Vijay A. Saraswat
From: http://www.saraswat.org/penn state/saraswat.jpg

While this may seem an unnecessary complication, the implications for computing are huge: while programs expressed in procedural languages must stop at every step, until data are available and processed, programs expressed in constraint languages can be executed before the actual data are available, by performing this kind of logical processing over the constraints and statements of the program. When data are available, the elaborated constraint logic statements can be rendered concrete, if so desired[92]. This, in turn, opens up the potential of simplifying concurrent execution of programs: as long as the actual data does not have to flow from one part of the program to another for computation to ensue, but only the constraints on the data, those program parts can be executed simultaneously (there are limitations, as I'll mention shortly). This realization, further explained at the end of this section, led to the development of **concurrent constraint programming**, *i.e.* languages for concurrent programming based on constraints[93]. The framework for such programming language was originally proposed by Vijay Saraswat (Figure 70) in 1990 (Saraswat, 1993).

[90] Another option would be to define NewTablePosition as a function, stating that when invocated, it would return the sum of those same two values or functions. This, however, doesn't change the overall issues mentioned in the text, because the data values those functions deal with would still have to be calculated and manipulated.
[91] If, as stated in footnote no. 90, NewTablePosition is a function, the same things happens: it is evaluated and the result must be an actual value, such as "Yes" or "No".
[92] This process is known as **instantiation of the logical variables**.
[93] ToonTalk, the programming language used in the field work supporting this thesis – and described in section 3.3.5 – is one such language, albeit restricted in the kinds of constraints it supports.

The reader unacquainted with logic programming may question how computation can advance, beyond a question such as the one presented, since it's highly likely that the next actions depend on the answer. But often the logic reasoning of the program can be divided into two separate lines of "reasoning", so to speak, to analyze separately the consequences of a "yes" or a "no"; and thus, the calculation can proceed, as long as the actions do not cause external consequences (showing unintended images on the screen, erasing important files, setting off a nuclear bomb, etc.) – and even in those cases, there is often the possibility of following through with the logic reasoning, postponing actual actions that may result from such a reasoning.

Obviously, there are limitations to this line of action: in programs that are not supposed to end, but rather to maintain a relationship with an external source (*e.g.*, a human user), the number of possible branches may be infinite[94]. And even on some programs with an expected termination, the same may happen when performing a logical analysis instead of a numerical analysis, as illustrated by the following citation[95].

> *« A more conventional illustration is provided by the treatment of arithmetic in logic programming languages, such as Prolog.* (…) *no one can deny the necessity for operations to add and to multiply two variables, whose value will be known only at runtime. This would mandate that the constraint-solver also be able to solve systems of basic constraints of the form* `X+Y=Z` *and* `X×Y=Z`*. This problem, however, is enormously complicated – and if the variables* `X`*,* `Y` *and* `Z` *range over the integers, then it is undecidable, as a result of the unsolvability of Hilbert's Tenth Problem!*[96] *»*
>
> *(Saraswat, 1993, p. 26)*

Summing it for the reader, this is the general point: the logic evaluation of a program cannot compute in advance all possible resolutions paths to a problem (something hardly surprising, since that would render it completely deterministic). But its combination with a view of programming as a refinement of constraints allows for a much simplified and powerful specification of concurrent behavior.

To conclude, I'd like to briefly present a little more technical background and consequences of these ideas from constraint logic programming and concurrent logic programming, as combined by Saraswat into the framework of concurrent constraint programming, already mentioned above.

When dealing with decision problems such as the one presented before, constraints help the process of logical deduction, by allowing the program to focus on constraints themselves, not on the actual data or its eventual values.

[94] Such programs, as presented in the previous section, are dubbed "reactive" programs, while programs that are conceived to end, eventually, after completing some processing, are dubbed "transformational."

[95] In logic programming without constraints, this situation can also occur in situations where the number of alternatives is finite: the number of logic variants may be so huge as to render unviable the scrutiny of all possible answers to each decision – a problem known as "combinatorial explosion". This problem is tackled by control methods present in those languages – which also pose limits on concurrency.

[96] For a presentation of Hilbert's problems aimed at laypeople, including his tenth problem, I recommend the article by Portuguese physicist and mathematician Jorge Buescu (2003, pp. 73-86). English-language sources suggested by Jorge Buescu in that article include the book *The Hilbert Challenge*, by J. Gray & D. Rowe (Oxford University Press, Oxford, UK, 2000), and the Web page by David Joyce from Clark University, at http://aleph0.clarku.edu/~djoyce/hilbert/toc.html.

For instance, supposing one wants to drop a table that is being carried, there could be a specification that such could not occur from higher than 5 cm. Instead of analyzing, monitoring and controlling several values for the height of the table, a constraint language could rather specify that:

```
A table-drop is valid...
... when the floor distance is less than 5 cm.
... when the floor space is clear.
... when the doorway has been transposed.
... when the table is being carried.
...
```

The program can thus work around the concept of drop height not as a specific height, but just with this set of constraining information, which is built as the program progresses. This is a concept dear to constraint programming, known as **computation through approximation** (Saraswat, 1993).

A direct consequence of this notion is that in concurrent constrain programming what is stored are not the actual data values, but rather the combination of all constraints on the data, such as the constraints above. This means that this model views constraints as being combined and stored, instead of the data. The storage element[97] of the computational system thus becomes itself one large and complex constraint on the data being processed by the program. This is known as the notion of **store as constraint**. The direct consequence is that the data is disregarded in favor of its information content – a distinction similar to focusing on the value of money instead of its figures.

Since the storage element of the system is seen as a constraint on data, not as storing data, in constraint languages it makes no sense to speak in terms of storing or retrieving data (or, in more usual computer-science terms, write and read data). Rather, this programming style expects a problem such as "`Is the table to be dropped?`" to be converted into questions **asking** the constraints whether the consequence of dropping the table would be valid. (This "ask" operation is the analog operation to traditional data-read or data-retrieval operations.)

In the same sense, rather than conduction operations on the data, to change it, as happens in traditional procedural languages, the constraint programming style expects rather that programs impose further constraints on the data, by adding such constraints to the storage, in effect **telling** it about new constraints. (This "tell" operation is the analog to traditional data-write, data-store, and data-edit operations.)

It is this focus on constraints that also immensely facilitates concurrency: since computation is not changing a value, but rather refining it by application of constraints, "*there is no reason to assume that at most one agent is available to carry the computation forward. A host of agents may simultaneously work on the problem. Even if all concern themselves with the same variable, it is still possible for them to do useful work by contributing non-redundant pieces of information constraining the variable*" (Saraswat, 1993, pp. xxiv-xxv).

Finally, it should be noted that constraint programming isn't simply a limited programming model, restricted to specific kinds of programs; as I already mentioned on p. 115, it is a form of logic programming, retaining the entire generality of logic programming. Just as in logic programming, the programmer is focused on the processing and combining of information in logic relationships, rather than on the processing of data values (which is the focus of functional programming). But the logic programs process data values all the same: it's just that processing data becomes a simple implementation of the logic structure attained through the use of this kind of language.

[97] *I.e.*, the memory.

3.3. Programming languages

3.3.1. Perspectives in view of selection for preschool

Programming languages aren't exactly a rare item: I am not aware of any collection claiming to be complete, but a nice approximation of that is Bill Kinnersley's "The Language List" (Kinnersley, 1991), a Web site which, at my last count, included 3118 languages, from *LISP[98] to Zz[99].

With such an ample choice, selecting languages for research on programming in the context of preschool education requires some sorting. This section 3.3 aims to provide clarification on three sorting parameters later mentioned in section 5.1: written vs. visual programming; concrete vs. abstract programming; and animated programming (as opposed to animating programs).

3.3.2. Textual vs. Visual programming

The relevance of this first division of programming languages is, I believe, clear enough in the scope of this thesis: the vast majority of children aged 3, 4 or 5 don't know how to read or write; and many – I dare guess "most" – of those who do must work hard to identify words and letters. This situation clearly requires that, in order to research programming concepts in general preschool settings, one considers programming languages that do not depend on reading or writing of textual information. Typically, programming languages that are not based on text are dubbed "visual languages" – hence the name used in this section's title. I will defer mentioning programming languages and concepts that were developed with the specific aim of empowering children users, until the coming sections 3.3.4 and 3.3.5.

Such deferring is necessary because visual languages are not seen as useful only for children or other text-impaired individuals; rather, their origin and development is linked to the human skills for image-processing and recognition of visual patterns. Those skills can more readily be put to use when a language takes advantage of visual features. Hence, this section deals with a presentation of general concepts of such languages, regardless of the assumed technical expertise or age of intended users.

That statement about putting to use human visual skills is a general one, which includes traditional languages and methods. For instance, even if a language is entirely textual, visual cues such as placement of words, line-splitting and indenting are commonly used to increase the readability of its code by human programmers[100] (*vd.* example in Table 8).

Code in the Pascal language without indenting or splitting of lines	Code in the Pascal language with indenting and splitting of lines

`...`
`while (count < limit) do begin test=getnewitem(); if (test=TRUE) then inc(count); end;`
`...`
	`...`
	`while (count < limit) do`
	`  begin`
	`  test:=GetNewItem();`
	`  if (test=TRUE) then`
	`  inc(count);`
	`  end;`
	`...`

Table 8 – Impact of indenting in code readability

[98] "*A data-parallel extension of Common LISP for the Connection Machine*" (Kinnersley, 1991, citing "The Essential *LISP Manual", TM Corp 1986).
[99] An extensible language designed to develop compilers for the Ape100 parallel computers (Cabasino *et al.*, 2001).
[100] The textual language Python even acknowledges this in its syntax, by using indentation to specify grouping of instructions, rather than using traditional pairs such as "begin end" (used in Pascal) or "{}" (used in C, C++, C# and Java).

Other visual techniques are common in traditional programming, particularly in planning and design. One common example, employed in the specification of programs, is the use of flowcharts. For instance, the code used in Table 8 could be represented in the flowchart format[101] of Figure 71.

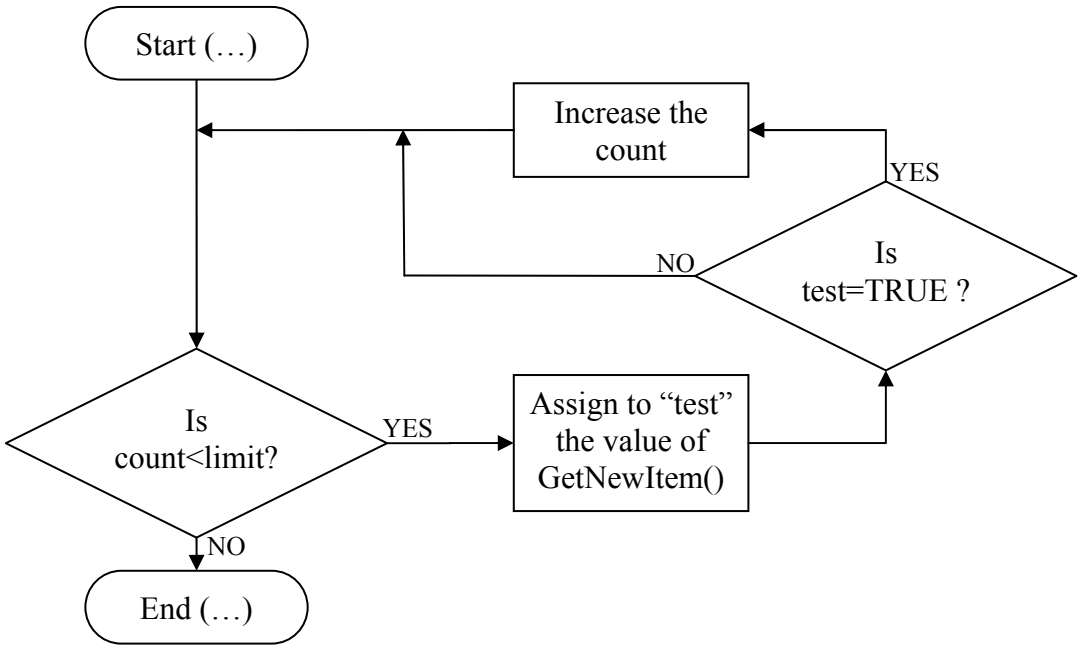


Figure 71 – Sample flowchart specifying the program in Table 8

Traditionally, flowcharts were used in the planning and design phase of software programming projects, but the final coding took place with a textual language, into which the flowchart was manually converted.

> *« In those days, the accepted method for writing a program was first to manually draw a flow chart, in other words a diagram of the flow of control, and then to translate that diagram (again manually) into lines of FORTRAN (or other language) code. »*
>
> *(Scrivener, 1994)*

This manual conversion is no longer necessary in modern visual programming languages: the full program can be expressed visually, without further manual conversions.

While the historical evolution of programming with textual instructions was presented in sections 2.2.5 and 2.2.6, it was so as a necessary part of the general history of computer programming. In this section I simply intend to make a presentation of visual computer programming systems, not present a full historical view[102].

Visual languages are often seen as providing an easier-to-use and easier-to-learn environment for programming. However, research on the positive, negative or neutral impact of visual languages in professional programming activities has been limited and contradictory (Whitley, 1996; Whitley & Blackwell, 1997, pp. 3-4), although the most recent research tends to be positive (Johnston *et al.*, 2004, pp. 17-18).

In my view, an important insight into this matter was provided by a thorough survey on the expectations of professionals from academy and industry, conducted by Kirsten Whitley and Alan Blackwell (1997), regarding the impact of visual programming languages on programming in general. In that study, academics often had "*optimistic theories regarding the influence that new*

[101] According to Nickerson, Herman Goldstein claims to have created the fist flowchart for computers in 1947, while working with John von Neumann (Nickerson, 1994, ch. 2, "The Origins of Visual Programming").
[102] For the reader with an interest in historical data on visual programming languages, I recommend the section "The Origins of Visual Programming", in the PhD dissertation of Jeffrey V. Nickerson (1994, ch. 2).

[visual] *programming languages can exert on the mental processes of the programmer*", while professional programmers tended to be focused on "*potential improvements in productivity that arise from straightforward usability issues, rather than from theories of cognition*." And within the professional programmers surveyed, there was a strong distinction in views between a mixed group of programmers visiting a trade show and another group, of programmers with experience in the visual programming language LabVIEW (Figure 74, p. 125). The programmers with experience in this visual language had significantly more favorable views regarding the usability and power of visual languages in general than the sample of general programmers.

This paints a global pattern similar to the issue on concurrent vs. sequential programming styles and their ease of use (*vd.* section 3.2.1). The cognitive impacts under the academic focus are dependent on pre-existent cognitive strategies that people initiating visual programming may have, in the programming domain; and the impact on productivity and usability, which is the focus of current programmers, is certainly dependent on how visual languages adapt to the programmers' current usage strategies (textual), rather than on any novel visual-specific global strategy. Indeed, those researchers acknowledge this, by saying[103]:

> *« The professional programmers exhibit a preference for the tools that they have had most experience of using. We recognize that this might produce significant biases when programmers are questioned about the value of their tools. These biases can even extend to significant skepticism about the advantages of new techniques, whether or not the programmer fully understands the technique being described. »*
>
> *(Whitley & Blackwell, 1997, p. 22)*

Under this line of reasoning, it is a promising indicator that programmers experienced in the visual language LabVIEW had favorable views regarding its comparison with textual languages[104].

But before presenting technical features of visual-programming, I propose a look at the notion itself: **visual computer-programming languages**. Such languages, in order to deserve their name, must possess features relevant to all naming words of their category:

- they must, first of all, be **programming languages**, *i.e.* allow the expression of meaning and content of programs (therefore, mere graphic visualizations of the execution of programs, not of the programs themselves, are not considered[105]);
- they must have been designed for **computer-programming** tasks (therefore, hand-sign languages, flag languages and so forth are not included);
- they must be **visual**, in the sense that they are not textual (this only excludes languages that are exclusively based on written words for expressing the program logic – it does not exclude those that employ text along with other visual content[106]).

> *« ''Visual Programming'' (VP) refers to any system that allows the user to specify a program in a two (or more) dimensional fashion. Although this*

[103] In a more recent paper, they follow up on this line of reasoning, by stating "*We propose that VPL research be partly directed by methodical study of skilled users who are experts in the use of existing VPLs*", VPL meaning "visual programming languages" (Whitley & Blackwell, 2001, p. 436).

[104] The group of LabVIEW programmers in this study by Whitley & Blackwell had different levels of expertise regarding textual programming languages, but the vast majority had a large experience: over 160 of those programmers reported experience in programming categories "*Medium projects*", "*Big projects*", "*Several years*", and "*10 years+*", for "*General programming*" (not "*LabVIEW programming*"), while only about 55 programmers indicated the categories "*Played around*", "*Training course*", and "*Small programs*" (Whitley & Blackwell, 2001, p. 453).

[105] "(…) *it is more accurate to use the term Visual Programming for systems that allow the program to be created using graphics, and Program Visualization for systems that use graphics only for illustrating programs after they have been created*" (Myers, 1989, p. 5).

[106] "(…) *products such as Visual Basic are not graphical languages because such products require that program logic be typed in as text*" (Whitley & Blackwell, 1997).

> *is a very broad definition, conventional textual languages are not considered two dimensional since the compilers or interpreters process them as long, one-dimensional streams. Visual Programming does not include systems that use conventional (linear) programming languages to define pictures, such as, Sketchpad, CORE, PHIGS, Postscript, the Macintosh Toolbox, or X-11 Window Manager Toolkit. It also does not include drawing packages like Apple Macintosh MacDraw, since these do not create ''programs''* (…). »
>
> *(Myers, 1989, p. 4)*

By using these definitions (my own and Myers') to analyze the examples in Table 8 and Figure 71, it is clear that the mere usage of indentation does not turn a textual language into a visual one[107]. The flowchart, however, represents a visual language, since it contains the entire logic of the program, and while using text, its full meaning is not conveyed just by it: the symbols' shapes provide insight on the meaning of the textual information, and the connecting arrows allow the reader to understand the sequence of instructions and their transitions (a.k.a. flow of control).

But the converse is also true: the non-textual information isn't sufficient to convey the entire logic of the program. In fact, each element in the flowchart matches an instruction in the textual program, and the flowchart just provides a simpler way for the programmer to visualize the flow of control.

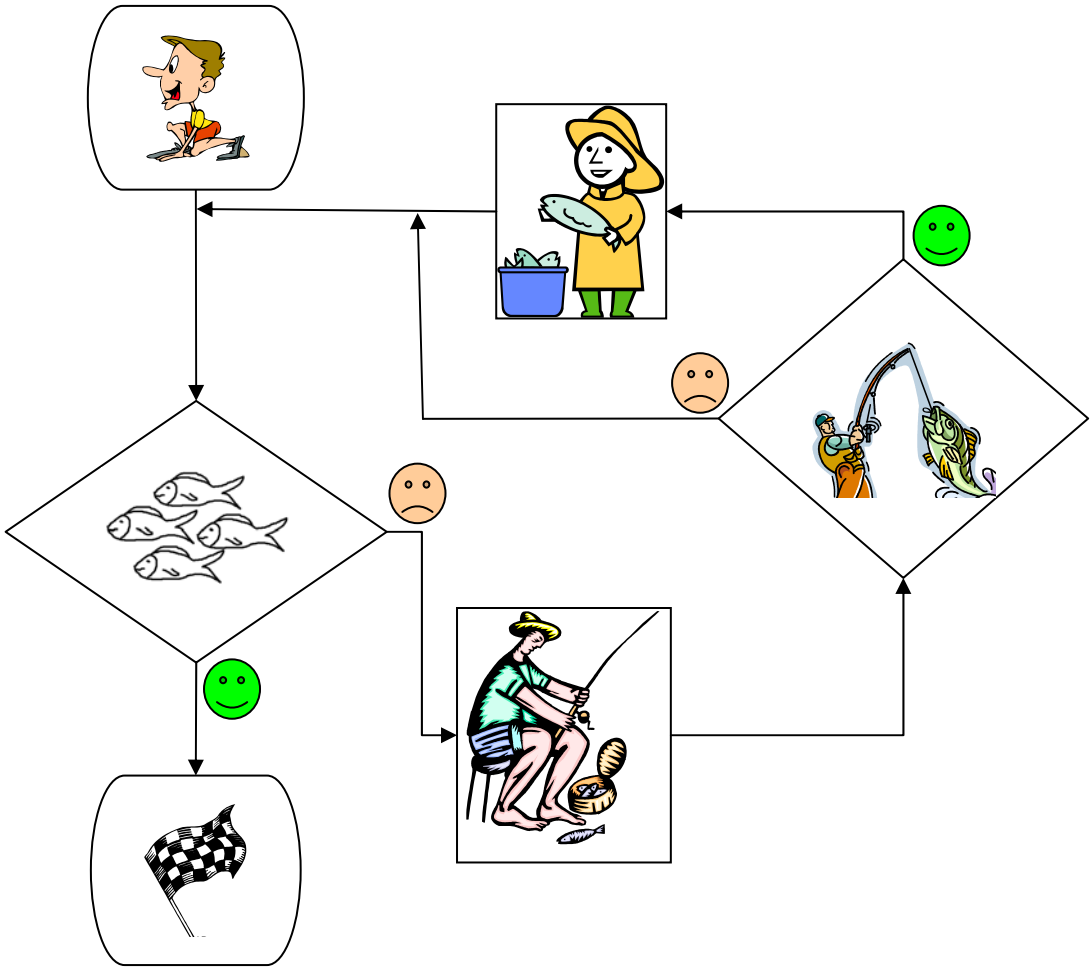


Figure 72 – Sample pictorial flowchart specifying the program in Table 8

[107] Indentation as used in Python (footnote 100) can be seen as a sequence of space or tabulation symbols, inserted in the sequence of the text. Therefore, it doesn't alter the one-dimensional nature of the programs.

This might at first sight invalidate usage of such kind of diagrams as flowcharts in pre-literate settings. However, the mere presence of the text is not an absolute limitation, since flowcharts can be used resorting to pictorial information rather than textual one. Figure 72, on p. 122, presents a pictorial version of the previous flowchart, constructed by simply employing clipart pictures. One can easily imagine such a flowchart being constructed in physical fashion (paper, crayons ...) in a preschool activity room, with the children themselves taking part in the drawing of the pictures and their gluing to the flowchart symbols.

A flowchart like this can be used in preschool settings as an activity in the mathematic logical domain. The starting point is represented by an athlete preparing to race, the finishing point is represented by a checkered flag, the positive and negative results are represented by color-coded smileys, and the remaining activities and decisions are represented by actual pictures depicting actions and events.

An activity could take place with the physical version of this flowchart, along with an actual bucket for fish, toy fish, toy fishing rods, and toy pond.

- "Start!"
- "Are there 4 fishes in the bucket?"
- "No? Set the bait and lay the rod!"
- "Did any fish bite?"
- "One did? Put it in the bucket!"
- "Check to see if there are 4 fish in the bucket."
- … (Eventually, there will be 4 fish in the bucket, and the flow ends at the checkered flag.)

This example is a way of turning a visual language based on text into a fully visual language. I'm not presenting it as a proof of concept, but simply as a way to expose two problems and limitations that occur in such a process: the pictures used are either **metaphorical/hieroglyphic** (*i.e.* standing for a meaning or word, rather than for what they actually depict) or artistic (or photographic) **depictions of an actual activity** or condition.

Metaphoric or hieroglyphic pictures, which in computer programs are usually called **icons**[108], should take in consideration the main concepts around the notion of metaphor, which will be addressed in the next section (3.3.3, "Visual programming for children: concrete vs. abstract", but in a short note for the convenience of the reader I'd like to point out here that the interpretation of metaphors and hieroglyphs is a convention, based on cultural norms and personal experience – and can easily become as complex to read and use as a textual language (or even more complex).

However, the depictions of an activity or condition, while aiming to be as truthful as possible to their intended meanings, can suffer from the same problem. For instance: the picture meaning to represent a fisherman placing its catch in the bucket can be interpreted as taking a fish out of the bucket instead[109]; or as scaling the fish just-captured; or even as eating it! But the major technological consequence of basing a language on such depictions is that they cannot be standardized, due to their open-ended nature: the number of depictable activities is virtually endless. This renders unviable their interpretation by a computer.

[108] The visual language Cantata calls them "*glyphs*" (Mosconi & Porta, 2000, p. 78).
[109] The logic behind this can be, for instance, "I took one fish out, so I'll return one fish to the pool."

Some languages have partly avoided these problems, by using techniques of programming-by-demonstration (*vd.* section 2.3.2), recording not the descriptions of the actions to be performed, but the actions themselves. Two simple examples of such techniques are:

- recording of initial state and final state (considering that transformation to be the action – this is known as graphical re-write rules), and
- the recording of a specific action script, as in a theatre play, movie of comic book.

Examples of programming tools using these methods are Stagecast Creator and ToonTalk, respectively, which are presented in the coming sections (3.3.4 and 3.3.5).

Flowcharts are far from being a perfect example of visual languages, thought: I have employed them so far merely for the sake of their familiarity. They allowed me to present the general ideas above, without mixing them with the technical aspects of visual programming.

One down-to-earth limitation of flowcharts is that they "*tend to get large and messy, as decisions can have many branches. In order to work around this problem, extensions to flow charts allow for charts to be terminated and then resumed on different pages* [or screens, or using scrolling]. *Yet, since flow charts require loops to show return arrows, a program with many loops requires return arrows that may span many pages* [or screens, or require long scrolling or zooming in order to be followed]" (Nickerson, 1994, ch. 2).

The advantage of flowcharts, however, comes from them being basically visual representations of imperative languages, and therefore easily transformed into computer instructions[110]. For this reason, several visual-programming systems were based around the flowchart model[111]. But the existence of different programming models (as mentioned in section 2.2.6), such as functional, data-flow, and concurrent constrain logic programming, led to the development of visual languages inspired on them[112] (ToonTalk, presented in section 3.3.5, is a visual language under the concurrent constraint paradigm, which was used in the field research for this thesis).

One example of a language under a different paradigm is presented in Figure 73. It is a visual programming language based on functional programming, authored by Georg Raeder in 1984, called PIP (Programming in Pictures). In this language, a "*function is defined by drawing a picture of the functions input data, the functions output data and the actions on the data*" (Nickerson, 1994).

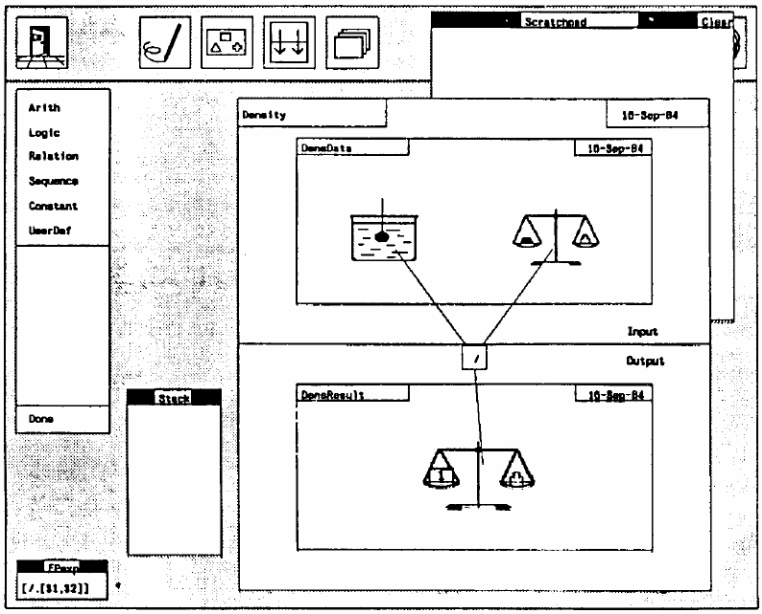


Figure 73 – Programming in the PIP visual language
From: From: Nickerson, 1994, fig. 2.61

But it was the dataflow paradigm in particular that led to the creation of many visual programming languages, including the previously-mentioned LabVIEW (Figure 74), one of the most successful ones in

[110] It is interesting to note that these advantages are at the technical level of implementation, while the mentioned disadvantages are at the human level of usability.
[111] For example, Ephraim Glinert's languages PICT, presented in 1985, and Blox, presented in 1986 (Nickerson, 1994, section 2.6.1). The earliest example, involving graphical specification of procedures, is from 1966 (Sutherland, 1966).
[112] In a survey of Visual Programming Languages, Brad Myers analyses no less than 37 different visual languages (Myers, 1989).

professional settings. This is partly due to dataflow programming being intrinsically related to a graphic notation, since dataflow programs, being based on dataflow graphs[113] can easily be expressed graphically (*vd.* section 2.2.6), partly due to the advantage of having a graphical-inclined notation that includes representations both of operations on data, and of the data itself.

In dataflow visual languages such as LabVIEW (Figure 74), data enters graphical items[114], representing program modules or procedures, where it is transformed; it then exits those items, in this way flowing across the program.

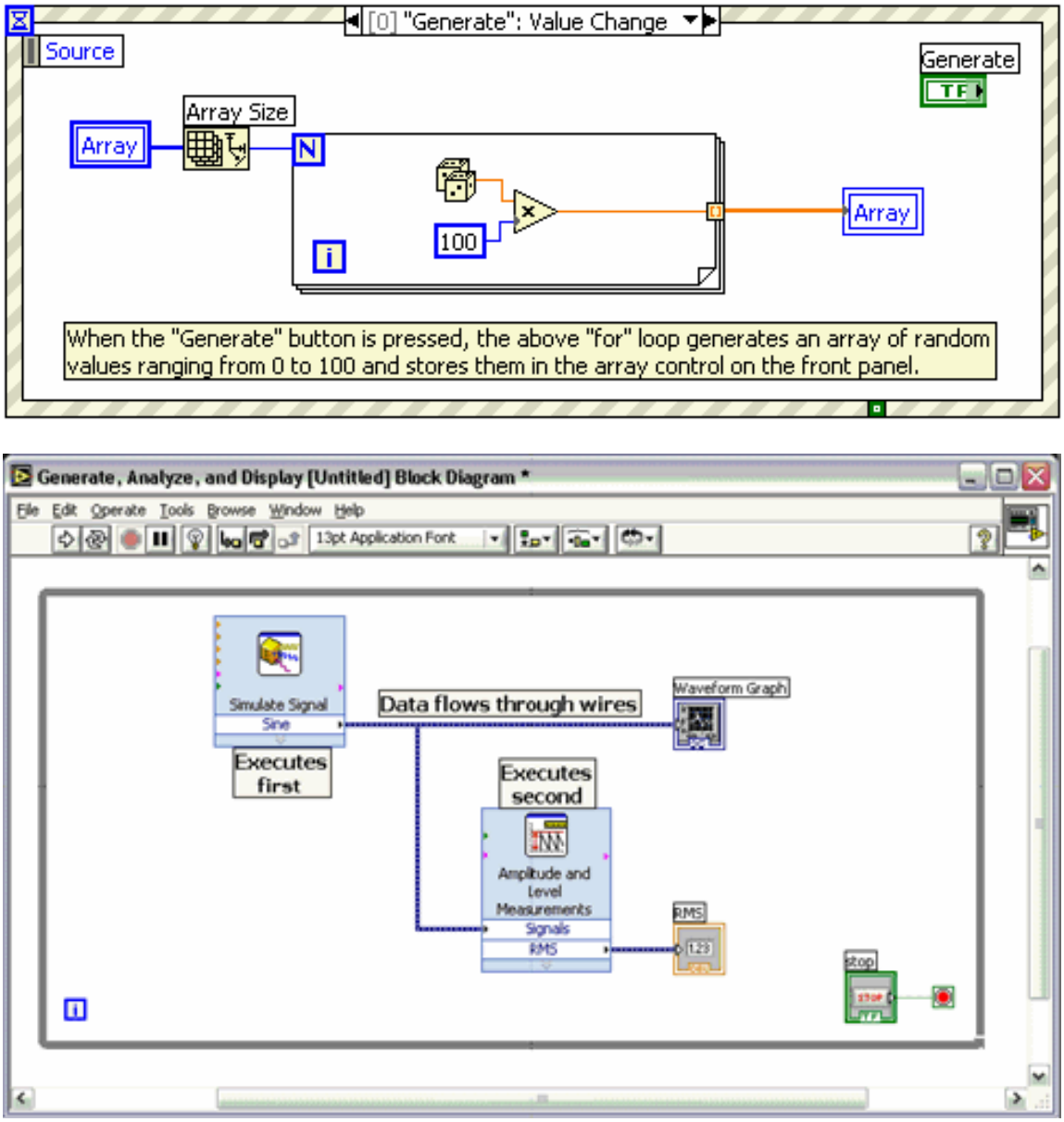


Figure 74 – LabVIEW programming examples: generating an array of random integers and using dataflow to determine execution order

From (respectively):
http://zone.ni.com/devzone/conceptd.nsf/2d17d611efb58b22862567a9006ffe76/60e39a753406ee7986256e5c000030c6/Overview/0.80A2?OpenElement&FieldElemFormat=gif
and
http://zone.ni.com/devzone/conceptd.nsf/2d17d611efb58b22862567a9006ffe76/f34045d2cc5357f486256d3400648c0f/Content2/5.1126?OpenElement&FieldElemFormat=gif

[113] "*The 'machine' language of programs designed to be run on dataflow hardware architectures is the dataflow graph. Most textual dataflow languages were translated into these graphs in order to be scheduled on the dataflow machine*" (Johnston *et al.*, 2004, p. 17).
[114] In LabVIEW, which was designed for creation of software interfaces for hardware devices (mainly in the field of real-time data acquisition) these items are called "virtual instruments".

Another feature of dataflow languages that also benefits from a visual representation is their potential for achieving large amounts of parallelism at the instruction level.[115] The visual editing of programs in a visual dataflow language can contribute to the programmer's identification of each procedure as an independent item, which can be running at the same time as other items.

An important notion is that while the aforementioned visual languages are visual in design and coding, when trying to analyze their execution (to detect and repair errors), most visual languages are **static** languages. *I.e.*, one can follow the flow of data or of control from element to element, but there is nothing to take advantage of the human abilities for visual pattern recognition of motion. "*Programs describe dynamic patterns.* [Most] *Visual programming languages attempt to encode descriptions of these dynamic processes with static pictures*" (Kahn, 1996b, p.3). In this aspect, static visual languages are identical to traditional textual languages. Also, they lack of adequateness to express dynamic changes in the program itself: the creation of new objects, for instance, is troublesome in dataflow languages[116].

But some languages did include the notion of **code animation**, by which the code can run in a visually animated fashion, therefore contributing to the programmer's visual understanding of its execution.

> *«* (…) *visual programming has failed to become wide-spread because it isn't radical enough.* (…) *Dynamic pictures, or animations, are a much better fit* [for describing dynamic processes]. *»*
>
> *(Kahn, 1996b, p. 3)*

A pioneering language in this regard was Pictorial Janus (Kahn & Saraswat, 1990). This language not only allows the fully graphical specification of a program, but also the animated, dynamic visualization of its execution. As an example, Figure 75 presents the frames of the animation of a program to append two lists[117].

> *« We envision a user describing – preferably in a visual manner – transformations, enhancements, and viewpoints on the infinite resolution animation to produce a "documentary" that delivers the required information. »*
>
> *(Kahn & Saraswat, 1990, p. 12)*
>
> *« A* (…) *program* (…) *will animate as* (…) *elements that grow, shrink, move, and dissolve. The animation of an agent reduction shows a rule expanding until it visually matches the agent contour. It then dissolves away leaving behind the body of the rule. Links shrink as newly created agents grow. »*
>
> *(Kahn, 1996b, p. 6)*

[115] Since data is entirely transmitted from a procedure to the next, any procedure can start to be executed "*as soon as its operands become available*" (Johnston et al., 2004, p. 3), without the need to wait for its turn in a specific execution sequence. This situation is similar to what happens in concurrent constraint programming languages: agents wait until there is enough information to proceed, and then apply their actions concurrently to the constraints (*vd.* section 3.2.2).

[116] "(…) *traditional dataflow systems have program graphs with a topology that is fixed at compile-time. Even in so-called dynamic graph systems, "dynamic" refers to the dynamic creation of a subgraph instance (for example, a new loop iteration or procedure call). The subgraph structure is still fixed*" (Grimshaw, 1993, p. 7).

[117] Unfortunately, animation frames like these are a poor substitute for the actual motion. This and other videos of Pictorial Janus programs can be found at the Web site (retrieved on October 15th, 2004) http://www.cs.concordia.ca/~haarslev/pjmovies/.

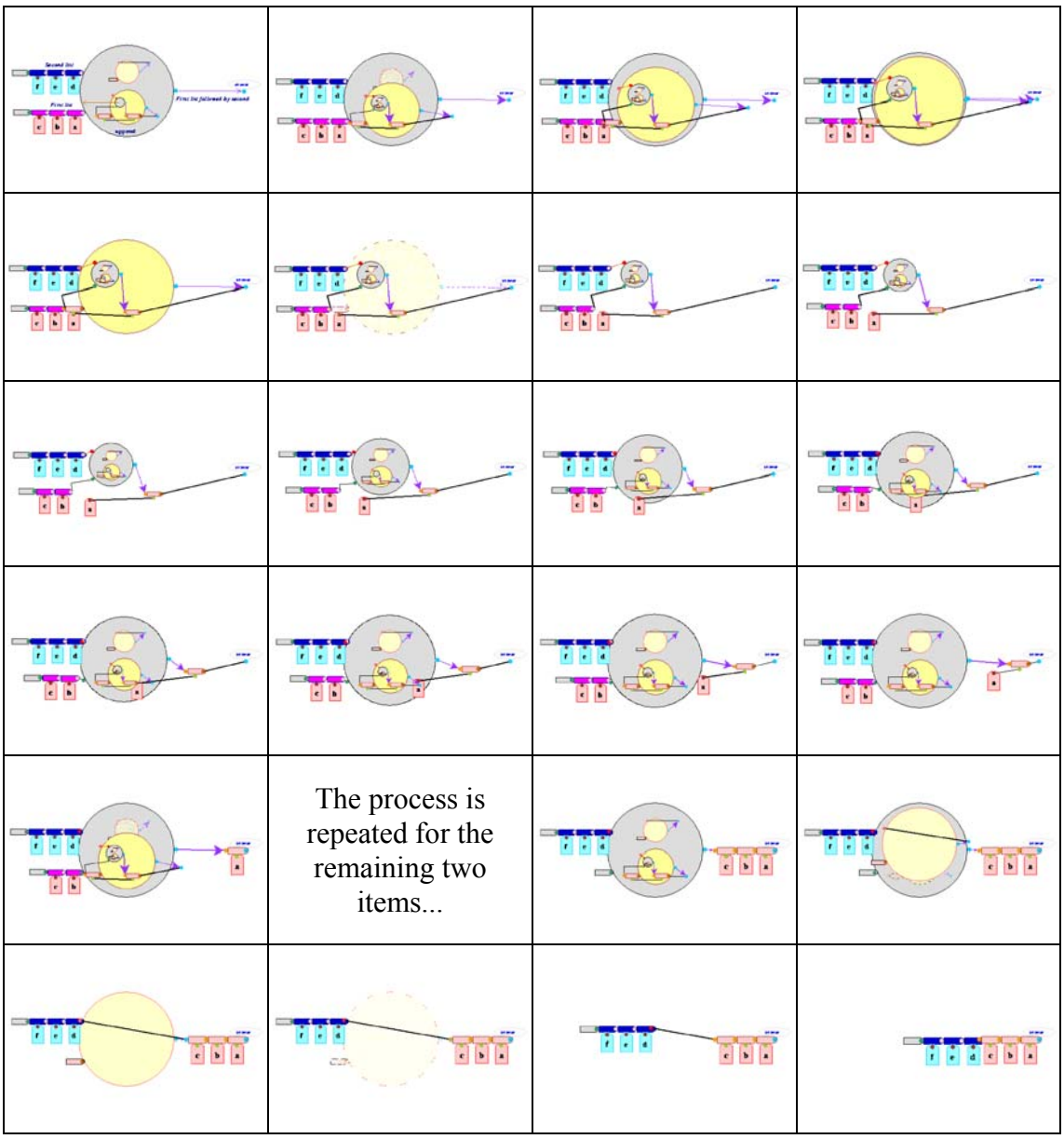


Figure 75 – Pictorial Janus, execution example: appending the lists "a b c" and "d e f" to produce "a b c d e f"
From: http://kogs-www.informatik.uni-hamburg.de/~haarslev/pjmovies/append.qt (animated movie)

In spite of all advances and improvements in visual languages, from flowcharts to visual programs with animation of their execution, one problem is common to all the languages referenced here (which are a fair representation of the genre). And that problem is that the usage and interpretation of these languages, irrespective of the reader's stance on their advantages and disadvantages, remain highly complex for the average (non-technical) computer user.

> « [In most visual programming systems] *(...) formal diagrams are used to encode programs and many people find formal diagrams difficult to understand and construct. Venn diagrams, for example, are much simpler than visual programs and while most readers of a* [technical computer] *journal* (…) *find them very easy, one forgets how hard it is for most children to learn them.* »
>
> *(Kahn, 1996b, p. 7)*

Due to this conceptual complexity, it's necessary to analyze programming languages in a different manner, not just in visual vs. textual, as was performed in this section. In order to be easily learned and used by children, languages must overcome this formal complexity at the level of their usage. This is the subject of the next section, 3.3.3.

3.3.3. Visual programming for children: concrete vs. abstract

As put forward in the previous section, the mere fact of being visual doesn't by itself render a programming language accessible to children. The visual syntax and visual formalisms can be as complex and hard to grasp and use as the formal textual ones. A typical way to pose this problem is by saying that those languages are abstract; by stating the problem in this fashion, a solution would require the annulment of that abstractness – *i.e.*, rending programming concrete, rather than abstract.

But this view is just a simplification of the problem statement, apparently "simplified" by saying that rendering programming accessible is achieved by rendering it concrete. In fact, the problem remains, but now in a different guise: when and how does programming cease to be abstract and start to be concrete?

This view of opposition between abstract and concrete is quite prevalent in education (although contradicted by recent theory and research, as I'll explain further ahead in this section): it is easily fitted within general "common sense", and supported by the work of several influential researchers on child development and children's cognition. Among these, the most prevalent educational theory that directly addressed this duality is the constructivist theory associated with the philosophy of Jean Piaget, Lev Vygotsky and Jerome Bruner, although the concept is also part of the earlier philosophy of Arnold Gesell, known as maturational theory (these theories are presented in section 4.1, "Early childhood education philosophy(ies)").

Under these theories, human cognitive development during childhood is a progression towards the ability for higher-order thinking, seen as achieved when the child demonstrates the ability to employ formal or abstract ideas. This progression is represented as a sequence of stages or levels that children traverse, until they reach mature thought, seen as formal and abstract. Following this logic, abstract thinking is also seen as superior to concrete thinking, which is regarded as a mere phase in children's cognitive development[118].

These notions of separate abstract and concrete thinking styles, seen as superior and inferior styles (or, in other words, seeing the concrete thinking style as a mere transitional style towards abstract thinking), have important implications for the design of computer programming languages for children.

The most crucial implication, in my view, is that if such notions were to be hold as true, then programming based on concrete notions, rather than abstract notions, would be necessarily inferior, limited in scope and reach. Programming languages based on the concrete would also not be as expressive or powerful as those based on abstractions. And therefore, children programming and programming languages aimed at children, by aiming to be concrete, would be hopelessly confined to "toy language" status, nothing more than an entry point into programming, unsuitable to explore the most powerful ideas, techniques and expressiveness allowed by an abstract programming language.

This limitation, in turn, would imply that the usage of concrete programming languages in education could only carry very limited benefits, unless it was seen as a transitional step towards abstract programming.

However, by pairing the ideas of abstract thinking separated from concrete thinking, and of children progressing from concrete to abstract thinking, one can reach a realization conflicting with real-word experience: children should then only be able to use "toy" tools (concrete tools) until they

[118] This has been questioned in recent years, and an example was already mentioned in this thesis, in section 2.4.5 (p. 80), when I presented the ideas of Sherry Turkle and Seymour Papert. They urged for a reevaluation of the concrete mode of thinking (which they called "bricolage"), and pointed out examples of its use by adults in non-trivial situations (Turkle & Papert, 1990).

become capable of using "real" tools (abstract tools)[119]. While the usage of toys and toying with non-toys are common activities for children, they are far from being all that children do. A widely-known situation that serves as a counter-example of that realization is that children are able to learn how to play a "real" musical instrument while they are very young (2 or 3 years old), seemingly still in the realm of concrete thinking, *e.g.* by using the Suzuki violin teaching method (Komlos, 1998)[120]. One may argue that a musical instrument is a real, physical object, and therefore concrete. But by playing it, a child produces music – which is not physical: the sounds have a physical impact, but its appreciation is at the cognitive and cultural level. This, too, could be dismissed by saying that such a child is merely reproducing – aping – sounds that he/she heard, and therefore acting on the concrete. This view of a child playing an instrument without a cognitive appreciation of the music being played is for me hard to accept: it seems just one step short of branding the child as unable to have emotions regarding the music being heard or played. But I am not presenting this example as a proof or demonstration of my viewpoints. Rather, it is used to clarify to the reader my point of apparent existence of inconsistencies on the view of strict separation between the concrete and the abstract, which I expressed at the beginning of this paragraph as a "realization conflicting with real-world experience".

Indeed, both research and more recent cognitive theories provide support for a different view on concrete thinking and its relationship with abstract thinking. It comes from three main areas: the constructivist views on the processes of learning, the ecological systems theory dating from 1979, by Urie Bronfenbrenner[121] (both are presented in section 4.1.2), and particularly the work on concrete mathematics done by Uriel Wilensky (1991 & 1993).

Under the ecological systems theory, the child is influenced by the entire environment where he/she is embedded. "Environment", here, means not just the physical and the social environment in the child's immediacy, but also events occurring in settings in which the person is not even present, such as the ties between school and home, or the conditions of parental employment.

And the environment's main influence upon a child is his/her interpretation of that environment, rather than any "real" features of it. As a consequence, any attempt to understand a child's actions is superficial, if it is solely based on objective features of the environment: to understand the child's actions, there is the need to question the meaning of those features for the child. This includes the impact of "real" objects and events on the child's motivation, and also the impact of "unreal" elements, such as those created by the child using his/her imagination, fantasies, and conceptions (Spodek & Saracho, 1993, p. 78).

This theory implies that the notion of "concrete" goes beyond concepts such as being physical or touchable: the child's feelings and sensations towards the world, as well as interpretations of events, imagined situations, etc., are very much "real" (concrete) for the child.

However, this ecological view is ignoring the cognitive processes at work in the child. The discussion on the abstract and the concrete thus needs to connect it to the constructivist view of cognition, particularly to the constructivist notions of accommodation and equilibration explained in section 4.1.

Simply put, these notions explained by Piaget (1975) mean that when people experience novel situations, they sometimes contradict one's previous notions and understandings, rendering them insufficient. This in turn unbalances one's conceptions, and the way to balance them, reaching cognitive equilibrium, is to accommodate the novel experience, by changing one's own self, constructing new notions.

[119] Or use real tools in "toy" fashion.
[120] A more radical point of view is that children learn about world and society in the real world and in the real society, not in toy versions.
[121] Now renamed as "bioecological system theory".

Pairing this conception with the implications brought by the theory of ecological systems on the notion of "concrete", as explained above, one reaches the idea that children's cognitive development by equilibration can occur both with physical and non-physical concepts, and these two kinds of concepts are present both in abstract and in concrete situations.

But this reflection about the implications of Piaget's and Bronfenbrenner's theories on the notions of abstraction and concreteness was still, in itself, abstract to me, as I hope the reader is by now finding this explanation so far. It was only when I became aware of Wilensky's view on the connection between concreteness, abstraction and personal viewpoints that I found this matter to have become concrete to me (Wilensky, 1991; *id.*, 1993, pp. 52-71). And in the same fashion, by now summarizing Wilensky's thought, I hope to give the reader a chance to have these explanations become concrete at last.

Figure 76 – Uriel Wilensky
From: http://www.agent.maine.edu/LLPI/people/UriWilensky2.jpg

Wilensky puts in question the intuitive physical notion of concrete and abstract, by presenting examples of how even physical objects such as snow, a pen or a chair can be, in fact, abstract, to extend the duality of concrete vs. abstract into a **continuum of degrees of abstraction and concreteness**. Concepts are thus not simply "concrete" or "abstract", but rather contain a certain degree of each.

> *« This particular pen that I am currently using, which is made by Papermate, is black, has a cap roughly one sixth as long as the stem, which has some chew marks on it, is much more concrete than just plain "pen" or even "Papermate pen". These descriptions of my pen ascend in levels of abstraction and can be further abstracted by making the move to "writing implement" or "communication tool". »*
>
> *(Wilensky, 1991)*

Wilensky puts forward the notion of relativity of the concrete, based on a simple notion: concreteness, under the example above for "pen", "Papermate pen", "writing implement" and "communication tool", can be seen as relative to the number of physical objects that a label refers to, but this association between labels and objects is personal, dictated by one's worldview and culture.

> *«* (…) *as was first noted by Quine*[122] (…)*, this is not the case: There are a multitude of ways to slice up our world. Depending on what kind and how many distinctions you make, your ontology can be entirely different. Objects like snow which are particulars in one ontology can be generalizations in another. Indeed for any concrete particular that we choose, there is a world view from which this particular looks like a generalization.* (…)
>
> *Thus it is entirely possible to imagine a visiting alien "seeing" what you call this concrete chair as a random collection of variegated particles designated by some strange abstract label. The alien might not even perceive the area of space you call chair to be filled at all, or to be filled partially by one object and partially by another. To make this alien more concrete, just imagine for instance a virus's "eye"-view of say a wicker chair. »*
>
> *(id.)*

[122] Wilensky is referring to Willard Van Orman Quine's 1960 work, "Word and Object" (MIT Press, Cambridge, MA, USA).

One doesn't have to picture an entity as different as an alien of a virus to realize how much culture and world-views can impact the experience of concreteness. The following quote from a literary work by William Camus[123] provides a clear example of this.

> *« It wasn't hard to recognize the chief's tepee, entirely white, plainly adorned with grey feathers. I rasped the door three times and entered.*
>
> *We, the Sioux, considered that the palefaces carried with them a smell of dead people; on their part, the white men stated that the Sioux smelled like a blend of badger and rancid fat. I thought that this was due to the fact of white men and redskins not having similar senses of smell. However, having lived for so long amongst the whites, the first thing to impress me upon reaching the Wichita was the odor that filled the air, entering the nostrils and impregnating my entire body. Above all, you mustn't think that it smelled bad; the white man, not being able to distinguish the various odors, smells everything at once and claims it to be uncomfortable. But a Sioux olfaction can separate each effluvium. That characteristic smell was composed of bison fat, horse manure, charred wood, animal sweat, old leather and remains of animals, bones, furs, innards, rotting under the sun. Each of these aromas wasn't unpleasant by itself, but together, they were surprising for the nose of a paleface. »*
>
> *(Camus, 1968, ch. 3, p. 92; originally written in French, translated into English from the Portuguese translation)*

Previously in this section, from the ecological systems theory, I stated that non-physical ideas can indeed be concrete (or, under Wilensky's continuum view, more concrete than abstract), since their impact on the reasoning of the child depends on the child's own interpretation of the physical and non-physical environment. Wilensky proposes a similar view, by following from the notion of relative personal views on the concrete. He defines concreteness not as "*a property of an object but rather a property of a person's relationship to an object*" (Wilensky, 1991). And from this, arrives to the notion of concreteness as being a property of both physical and non-physical concepts.

> *« It is because children share a common set of sensing apparatus* (…) *and a common set of experiences such as touching, grasping, banging, ingesting* (…)*, that children come as close as they do to "concretizing" the same objects in the world.*
>
> (…)
>
> *The more connections we make between an object and other objects, the more concrete it becomes for us. The richer the set of representations of the object, the more ways we have of interacting with it, the more concrete it is for us. Concreteness, then, is that property which measures the degree of our relatedness to the object, (the richness of our representations, interactions, connections with the object), how close we are to it, or* (…) *the quality of our relationship with the object. »*
>
> *(id.)*

But a most important realization by Wilensky is that this means that any concept can be concrete or abstract to someone.

[123] "*William Camus was born in the Yukon territory, his mother French and his father Iroquois. He was raised 'the Indian way' in Canada, where his father was a fur trader. He was a stock-car pilot in the USA* (…)*, and later a journalist and a defender of the Indian cause.*" Translated from the French version (CIELJ, n.d.).

> *« (…) any object/concept can be [/] become concrete for someone. The pivotal point on which the determination of concreteness turns is not some intensive examination of the object, but rather an examination of the modes of interaction and the models which the person uses to understand the object. This view will lead us to allow objects not mediated by the senses, objects which are usually considered abstract – such as mathematical objects – to be concrete; provided that we have multiple modes of engagement with them and a sufficiently rich collection of models to represent them. »*
>
> *(id.)*

A final consequence of this is that, from the point of view of a person, most concepts start by being abstract. As that person acquires more methods of interaction with a concept, that concept becomes increasingly more concrete. Thus, the more advanced concepts for each person are abstract, but not because of inherent "abstraction" characteristics: rather, they are abstract for a person because that person still hasn't rendered them concrete.

> *« It would thus appear again that the standard Piagetian view of stage*[124] *is turned on its head. In the school setting, rather than moving from the concrete to the formal, we often begin our understanding of new concepts (just as we often do with new people) by having a formal introduction. Gradually, as the relationship develops it becomes more intimate and concrete. Outside of school, in the world, our nascent understanding of a new concept, while not usually formal is often abstract because we haven't yet constructed the connections that will concretize it. The reason we mistakenly believed we were moving from the concrete to the abstract is that the more advanced objects of knowledge (e.g., permutations, probabilities) which children gain in the formal operations stage are not concretized by most adults (…) they remain abstract and thus it seems as if the most advanced knowledge we have is abstract. It follows that the actual process of knowledge development moves from the abstract to the concrete. Only those pieces of knowledge that we have not yet concretized remain abstract. »*
>
> *(id.)*

Returning to computer programming, these ideas point to the notion that computer programming can become concrete to the programmer if the programming system (and environment, following the ecological system theory) provides many ways for the user to "connect" with it.

These "ways to connect" can be seen as elements of the context of programming – or indeed, of the context of human activities in general. Such elements are numerous: *e.g.*, whether one is working with general concepts or specific concepts; manipulating an example or a generic placeholder; working mostly at the mental level or with some physical interaction; working with sets or isolated elements; with novel things or familiar things; etc.

An important technique in this regard is the use of metaphors and analogies. Employing metaphors and analogies, people can relate to novel areas and concepts by comparing them and linking them with areas or concepts that are well-known and familiar. In this regard, these links are not the property of any metaphor: they must be created by the person using the metaphor, either

[124] Piaget's stages are presented in section 4.1.2, p. 258.

from scratch or adapting someone else's metaphor[125]. Another way of putting it is that the links between concepts are, themselves, the metaphor.

This may at first be seen as confusing if one only considers metaphors in their classical sense. Traditionally, a metaphor was considered to be a mere stylistic technique in language, the usage of a word outside its conventional meaning. For instance, the often-heard "the processor is the brain of the computer" is a classical metaphor: since one considers brains to exist only within living organisms, the usage of the word in that sentence intends to convey the idea that the processor controls the computer like the brain controls the body.

But the modern view, known as contemporary theory of metaphor (Lakoff, 1992), sees metaphor as a much more profound connection between elements than a mere elegant comparison. Metaphor becomes a world-view, a mapping between different domains, to the point of sometimes the person using it not even realizing that a metaphor is being used. In this sense, most human thought is metaphorical.

> *« Metaphor is viewed more as a basic tool of cognition rather than a special turn of language, and most concepts are generated by metaphors. The exceptions are those concepts that are thought to be perceptual or cognitive primitives, such as **up** or **cat**. Aside from these references to concrete physical objects and experiences, metaphorical understanding is the rule. »*
>
> *(Travers, 1996, p. 31)*

This is also the meaning I am using for the concept "metaphor": a mapping between two concepts, allowing one to be understood in terms of similarities with another. In this sense of metaphor, simple sentences such as the above "the processor is the brain of the computer" are mere results of the larger metaphor. This larger metaphor is known as a **metaphoric model** (Lakoff & Johnson, 1980), or simply "metaphor", and the isolated ideas that it creates, such as that sentence, are considered to be mere **metaphorical expressions** (Lakoff, 1992).

Several examples of this perspective of metaphor as a world-view, or even a world-shaper, are given by linguist George Lakoff and philosopher Mark Johnson, *e.g.* their example of the metaphoric model "time is money":

> *« TIME IS MONEY*
>
> *You're wasting my time. This gadget will save you hours. I don't have the time to give you. How do you spend your time these days? That flat tire cost me an hour.* (…) *Put aside some time for ping pong. Is that worth your while?* (…) *He's living on borrowed time.* (…) *Thank you for your time. »*
>
> *(Lakoff & Johnson, 1980)*

But there is another side to this view: metaphors acting in this manner can also be thought of explicitly, looking for new ways to understand the connected concepts. By doing this, a person is in fact acquiring new perspectives on both subjects[126], and even though all such comparisons are limited in correctness, they provide further grasping points for the thinker. And as I've explained above, this can help the thinker render those concepts less abstract and more concrete.

[125] A notion that helps to explain the absence of meaningful results in experiments comparing programming systems usability in the presence and absence of metaphors (Blackwell & Green, 1999).

[126] Tony Veale calls this ability the "creativity of metaphor" (Veale, 1995, pp. 9-11). An example of such use is provided on the next page, for the field of computer science. Other, similar efforts have been done in many areas, including education and educational practices (*e.g.*, Vasconcelos, 1990).

*« If **Operating-System** is like **Religion***

***Then** Customer-Satisfaction is like Sense-Of-Belonging*

and Brand-Loyalty is like Religious-Faith

and Consumer is like Believer

and User-Base is like Congregation

and Windows-O/S is like Anglicanism

and Graphical-Window is like Stained-Glass-Window

and Graphical-Icon is like Religious-Icon

and Wimp[127]-Environment is like Cathedral

and Mac-O/S is like Catholicism

and Command-Line-Interface is like Protestant-Work-Ethic

and Ms-Dos is like Protestantism

and Graphical is like Baroque

and High-Level-Software is like Religious-Text

and Software-Command is like Moral-Directive

and Machine-Code-Instruction is like Cabalic-Message

and Hexadecimal is like Hermeneutic-Code »

(Veale, 1995, p. 11)

In computing, metaphors are extremely common, perhaps due to the fact that most elements of computers are not visible. For instance, while the processor simply combines and transforms electrical signals, humans refer to such actions as adding, subtracting or shifting values. This interpretation carries the implicit metaphor of the processor as a person doing calculations on numbers. And everyday cases are also common, in expressions such as "booting up" the computer, or "shutting down the system".

But the overall notion presented here is that metaphors, seen as models, are not simply exceptional situations for clarifying a point: they are an essential element of each person's interpretation of the world. And although some thoughts are obviously metaphoric, in other cases the underlying metaphors are not so clear-cut.

> *« Most people are not too surprised to discover that emotional concepts like love and anger are understood metaphorically. What is more interesting, and I think more exciting, is the realization that many of the most basic concepts in our conceptual system are also normally comprehended via metaphor—concepts like time, quantity, state, change, action, cause, purpose, means, modality, and even the concept of a category. These are concepts that enter normally into the grammars of languages, and if they are indeed metaphorical in nature, then metaphor becomes central to grammar »*
>
> *(Lakoff, 1992, p. 212, as cited in Travers, 1996, p. 33)*

[127] Is in fact an acronym (WIMP) for "Windows, Icons, Menus, Pointing devices" or "Windows, Icons, Mouse, Pull-down menus". The pun is around the lines that "real users use a command-line, all other are wimps".

In the same vein, metaphors are central to computing. As common examples, one can see the computer activity as a spatial movement (from Travers, 1996: "*the process is blocked*"; "*the machine went into a run state*"), and memory is commonly seen as space (*ibid.*: "*garbage collection*[128]", "*partition*[129]", "*allocation*[130]", "*compacting*[131]"). These are examples of metaphors that become so prevalent that one no longer thinks of them as metaphors. They are often referred as "dead metaphors", but while this classification is accepted in the classical sense, it is disputed when one considers Lakoff's concept of metaphor as a mapping of conceptual domains.

> *« The MEMORY IS SPACE metaphor might be considered dead since it is extremely conventionalized, but it is still alive in Lakoff's sense — the mapping between domains is still present and can be generative of new constructs, such as the slangy term "bit bucket" (the mythical space where lost bits go)* (…) *»*
>
> *(Travers, 1996, p. 33)*
>
> *« When the wine reviewer of The Times contrives the metaphor "a muscular vintage", she relies upon the reader to see it as an extension of that established metaphor whereby wines possess body, and in doing so he breathes new life into the metaphor (we actually conjure an image of the wine as a well-biceped body). »*
>
> *(Veale, 1995, p. 16)*

For this reason, some authors proposed the renaming of this concept as "transparent metaphors" (Travers, 1996) or "dormant metaphors" (Veale, 1995)[132].

I don't mean to ignore criticism of the concept of metaphoric thinking, particularly from the field of formal science (including computer science). The famous computer scientist Edsger Dijkstra made clear his position in 1989:

> *« By means of metaphors and analogies, we try to link the new to the old, the novel to the familiar. Under sufficiently slow and gradual change, it works reasonably well; in the case of a sharp discontinuity, however, the method breaks down.* (…) *the metaphors become more misleading than illuminating. This is the situation that is characteristic of the "radical" novelty. Coping with radical novelty requires an orthogonal method. One must consider one's own past, the experiences collected, and the habits formed in it as an unfortunate accident of history, and one has to approach the radical novelty with a blank mind, consciously refusing to try to link history with what is already familiar, because the familiar is hopelessly inadequate. »*
>
> *(Dijkstra, 1989, p. 1398)*

In my view, this negative view on metaphor is completely consistent with the contents of this section. It refers to the intentional use of metaphors as tools for understanding of novel concepts. But as I explained above, human thought and language is laden with dead/transparent metaphors, which have acquired so deeply the meaning to which they became associated in a new field, to the point of not even being seen as metaphors. Another contribution to realize the quixotic nature of attempt to entirely avoid metaphor can also be extracted from my previous presentation (at the

[128] The process of freeing allocated memory locations that are no longer referenced by any variable.
[129] A logical division of memory.
[130] Assignment of a memory block to a program.
[131] Moving allocated memory blocks in order to make them contiguous, maximizing the amount of contiguous free memory.
[132] Of course, the usage of "dead", "transparent" or "dormant" implies a metaphoric view of the concepts of metaphor as a living organism that can die, sleep and awake (dead, dormant) or as a lens affecting our vision (transparent).

beginning of this section) of the current cognitive theories and their view on abstraction. The notion of each individual constructing its own ontology or world-view implies that communication between two individuals cannot be achieved without a conversion between their distinct ontologies. Such a conversion only makes sense by incorporating the novel knowledge/information into the current personal ontology – and thus acquiring a metaphoric property in this process.

> *« (…) formalism does not really offer an escape from metaphor, for two separate reasons. First, even formal mathematics is riddled with metaphorical terms and concepts, such as the notion of a function having a slope (a physical metaphor) or being well-behaved (an animate metaphor). Secondly, very few mathematicians would claim that the use of formal methods exempts them from the need to use their imagination! »*
>
> *(Travers, 1996, p. 37)*

Overall, the importance of metaphor in the scope of this thesis derives from the contemporary theory of metaphor under two points (*id.*, p. 33):

- "(…) *some of our most fundamental concepts are structured metaphorically*",
- "(…) *it is possible* (…) *to gain a new viewpoint on these concepts by proposing alternate metaphors.*"

Since this discussion on abstraction, concreteness, and metaphor is quite long, I now present a brief consolidation of these ideas and their relation to programming languages:

- no programming language is completely formal/abstract or concrete, but rather somewhere in a continuum between these extremes, the exact placement depending on the relationship between the programmer and the language;
- consequently, the power of a programming language is not connected to hypothetical abstraction or concreteness features, since these depend on the richness of connections between the language and the programmer, not on the connections between the language and the computing machines;
- the creation of richness in connections between a language and the programmer is a metaphorical process of mapping between the concepts embedded in the language and the programmer's concepts;
- this metaphorical process is stimulated by properties of the language itself, designed for specific groups of users, but also by the overall social and technical context and background within which the language is introduced to the programmer and used by him/her.

Under this view, **programming languages for children** are NOT those that are "simpler", "smaller" or "less powerful", but rather **those that were designed to facilitate the metaphorical mapping between programming concepts and typical world views of children**. Being "simpler", "smaller" or "less powerful" can be important notions ruling some of these designs, but they aren't determinant or mandatory.

As an example, two pieces of code to achieve the same goal are presented on the next page. The top one was written in C, the bottom is its ToonTalk[133] equivalent. In both, a computational process (the "parent") is spawning another (the "child"): in the C example, the "fork" command is used, in ToonTalk a loaded truck is sent off. The parent process then sends data to the child process using a communication channel: in the C example, using a "pipe" construct; in ToonTalk, using a bird/nest pair. The child-language isn't in any way diminished in terms of expressive power or potential for elaboration, but its concepts are closest to children's typical worldviews.

[133] ToonTalk is an animated programming language; its syntax is presented in section 3.3.5, but one should be aware that this is only a "comic strip" of an interactive animation controlled by the programmer in a videogame-like manner (this approach to presenting ToonTalk code is also presented in section 3.3.5, on p. 198).

```
int pipedescriptor[2]; char text[10]; char msg[10];
pipe(pipedescriptor);
if ( fork() == 0 ) {
   /* child process executes this */
   close(pipedescriptor[1]);
   read(pipedescriptor[0], text, 10); /* get from parent */
   /*...do something...*/
}
else { /* parent process executes this */
   memcpy(msg, "Hi there!", 10);
   close(pipedescriptor[0]);
   write(pipedescriptor[1], msg, 10); /* send to child */
}
```

Start of procedure with "Hi there!" text. | Pick up a nest. | Drop it (egg hatches). | A bird is born.

Pick up a truck. | Using the magic wand… | …make a clone. | Put clone in truck.

Pick up an empty box. | Put the nest in the box. | Put nest in truck. | The truck goes away.

Pick up the text. | Give it to the bird. | Bird takes it to its nest. | The loyal bird returns!

Figure 77 – Sample program in traditional textual syntax and in child-oriented syntax[134]

[134] ToonTalk syntax is presented in section 3.3.5. As with any translation, there are subtle differences. In this ToonTalk code the child process will wait for the message and then will reproduce the parent's behavior, spawning its own child process and sending it the "Hi there!" message. Other ToonTalk alternatives for the spawning could be the use of a library (in ToonTalk, a notebook) or passing the procedure as another entry parameter (simply by including it in the original box alongside the "Hi there!" message). All these would unnecessarily complicate the equivalent C code.

3.3.4. Survey of programming languages for children

Early programming languages (for adults) employed a metaphor that was almost transparent: that of a person following a list of orders on paper. When programming in assembler languages (an example was presented in section 2.2.6), this much would be rendered clear by instruction names such as JMP (standing for "jump"), as if the computer had fingers that could jump from a line to another[135].

This metaphor evolved into one of communication or speech: **expressing a program** (or plainly, programming) would be seen as an act of telling the computer what to do, and the program would be a conversation between a person and a computer. For this reason, many attempts at rendering programming easier for non-technical users (or novice technical users) were taken under the light of simplifying this metaphorical conversation.

But while this conversational metaphor helps the conceptual mapping of part of the process of programming, it ignores two other crucial areas: the **execution of the program**, in the sense of relating a program's behavior to its form[136], and the **programming environment**, in the sense of the tools and support through which the programmer expresses the program. Another area which I address, later in this thesis, is the social environment of programming (*vd.* section 7.1). In this survey, I will not use it as a distinctive feature, for usually it is not a feature simply of programming languages themselves: the matter of how they are introduced and used is paramount in this regard. Also, the traditional design of programming languages for children has focused on the relationship between the child and the computer. Therefore, this survey section is conducted under the three initial analysis categories: expressing a program, understanding the execution of the program, and child-orientation of the programming environment.

Hypothesizing the impact of social aspects in programming language design is beyond the scope of this thesis, and is a field lacking adequate research and development efforts. However, as an example of possible outcomes of such research and development, I'd just like to point out here, briefly, the case of programming environments aimed at facilitating simultaneous development of a program by several users (a.k.a. "collaborative programming"). Examples of such environments and languages[137] are MOOSE-crossing (Bruckman, 1997), Cleogo (Cockburn & Bryant, 1998), and collaborative computations on the floor (Fernaeus & Tholander, 2003). A recent contribution is Mulspren, presented in the PhD dissertation of Timothy Wright, which also includes an analysis of collaborative programming environments (Wright, 2004).

There are four programming languages for children whose impact warrants a detailed description in this dissertation: ToonTalk (which was used in the field work supporting this dissertation and is the subject of section 3.3.5), and Logo, Squeak Etoys, and Stagecast Creator, the three of which I will now present in detail. The final part of this section presents a commented list of other programming languages for children.

[135] A non-metaphorical version of such "jump" instructions would be named something like CPC (for "change program counter"). This, of course, if we ignore that the name "program counter" is itself a metaphor for the digital circuit of a register holding the address of the currently-executing instruction (another common name for such registers is "instruction pointer" – even more clearly metaphorical).

[136] *I.e.* the instructions composing the program.

[137] Programming with physical items, as described latter in this section, also provides an environment where collaboration can easily be employed, by having more than one person manipulating the objects used for programming. But in general, this possibility didn't impact the design of these systems.

Logo

(1966/67)

The Logo language was the first to be designed specifically for children use. Previously, other languages such as BASIC and COBOL aimed to simplify the programming process for non-technical users, or "novice" programmers; but such users were thought of as young adults, rather than children – even if many children ended up programming in BASIC, for instance.

Logo history and general design goals were presented in section 2.4.5, but not the child-aimed features of the language. A major problem in doing it for a language almost 40 years old, of which many versions were created, is deciding which features to consider, besides those available since the first version. I decided to focus on the most traditional aspects of the Logo programming environment, namely the original language features, available in the first version of Logo, and the "turtle graphics" subset which was included a few years later. There are three reasons for this choice to include turtle graphics here, while ignoring newer Logo-related developments: firstly, it was a command subset of tremendous popularity, to the point of sometimes people mistakenly assuming Logo is only about turtle graphics; secondly, turtle graphics have greatly impacted Seymour Papert's educational ideas[138]; and thirdly, the turtle graphics command set and programming model have inspired and were employed in many other programming languages for children, which are mentioned later in this section. By including Logo graphics here, under the Logo heading, their merits are assigned to the appropriate language.

As I mentioned in section 2.4.5, from the very start the key features of Logo were planned to allow the programmer to focus more on the idea of the program, and less on its written implementation. In this sense, its first contribution was the simplification of the written expression of programs. To achieve this, many typical syntax elements were abolished, and the language keywords and commands chosen carefully to resemble the current English language, sounding less technical (an approach that had also been pursued previously, although perhaps not so radically, in other designs of programming languages for novices).

A way to understand this simplification more clearly is to compare Logo with Lisp, the language upon which the design of Logo was based, rather than comparing it to a completely different language. Table 9 presents the same command in Lisp and in Logo. It results in the presentation of the text "Hello World" on the screen[139].

Lisp	Logo
`(princ "Hello World!")`	`print [Hello World!]`

Table 9 – "Hello world!" code in Lisp and in Logo

The differences in this example are: the parentheses used in the Lisp code disappeared in the Logo example (although they could have been used, if so the user wished); the quotes became square brackets; and "`princ`" became "`print`". What's the logic behind these changes? The resulting command does seem more readable as an English-language expression, apart from the change between quotes and square brackets, but why wasn't Lisp just as readable in the first place?

[138] *E.g.* in the idea of body-syntonicity (as mentioned in section 2.4.5 and explained in section 4.2.2).

[139] Presenting "Hello World!" on the screen is the most typical example provided by programming language textbooks and manuals to introduce a language's syntax to a new user. Several Web sites, such as http://www.cuillin.demon.co.uk/nazz/trivia/hw/hello_world.html, provide collections of "Hello World!" code in several languages.

The reason lies in the goals of each language. Lisp was developed as a language where several mathematical ideas were explored[140], and its general design aim was the encoding of fairly complex logic relationships, in such a way that would simplify not the coding process but rather the demonstration of its "correctness".

> *« (…) this combination of ideas made an elegant mathematical system as well as a practical programming language. Then mathematical neatness became a goal and led to pruning some features from the core of the language. This was partly motivated by esthetic reasons and partly by the belief that it would be easier to devise techniques for proving programs correct if the semantics were compact and without exceptions. »*
>
> *(McCarthy, 1979, p. 2)*

This is, for instance, the reason for the use of parentheses in the expression above: their absence in that specific command, where they are redundant, would pose an exception. Logo doesn't require parentheses when the expression can be unequivocally evaluated without them. For instance: to print the sum of 3 and 4, Logo accepts `print sum 3 4`, whereas Lisp would require[141] `(princ (+ 3 4))`. This use of parentheses, however, is also valid in Logo. This way, potentially equivocal expressions can be clarified, if necessary.

The design goals of Lisp are also the reason for the odd-looking "`princ`" command for outputting text to the screen. Why not simply "`print`"? Because that name is used for another output function, which uses some extra formatting characters[142]. The output of text intended to be read by humans is a special case, and thus remitted to the strange-looking "`princ`", standing for "print characters" (Adams, n.d.). In Logo, the presentation of text to humans is considered to be as important in itself as the presentation of text to other language procedures. And the method employed (and encouraged) to address procedures and language primitives that are similar but provide different functionality is to use English-language words, rather than potentially odd mnemonics (which can always be defined afterwards as abbreviations). So, if one wants to output structured data, rather than present text to the user, an instruction named "`show`" is used. The same functionality is achieved without resorting to odd-sounding words such as "princ":

```
print [Hello World!]
```

Hello World!

```
show [Hello Word!]
```

[Hello World!]

[140] "*As a programming language, LISP is characterized by the following ideas: computing with symbolic expressions rather than numbers, representation of symbolic expressions and other information by list structure in the memory of a computer, representation of information in external media mostly by multi-level lists and sometimes by S-expressions, a small set of selector and constructor operations expressed as functions, composition of functions as a tool for forming more complex functions, the use of conditional expressions for getting branching into function definitions, the recursive use of conditional expressions as a sufficient tool for building computable functions, the use of λ-expressions for naming functions, the representation of LISP programs as LISP data, the conditional expression interpretation of Boolean connectives, the LISP function eval that serves both as a formal definition of the language and as an interpreter, and garbage collection as a means of handling the erasure problem*" (McCarthy, 1979, pp. 1-2).

[141] This usage of "+" before its operands is called prefix notation, and the notation used in traditional algebra, "3+4", is called infix notation. Being aimed at children, Logo accepts not just Lisp's prefix notation (using "sum" instead of "+", to ensure its readability in current English) and the traditional infix, "3+4". This is another child-oriented simplification of Lisp, but not a simplification regarding other languages, since the infix notation is the most common in programming languages in general.

[142] The aim being to provide formatted expressions on output, to feed a function called "read", which is used for input of expressions.

Another important example of simplification at this level is the method of defining a procedure, by using the word "`to`", which results in an English-language construction:

```
to sayHello
   print [Hello!]
end
```

This allows the child to issue the just-defined procedure "sayHello" (immediately after, if so desired), just by typing:

```
sayHello
```

Hello!

Completing the analysis of the example in Table 9, the usage of square brackets in it, instead of quotes, seems to be counter-intuitive, since quotes are common sentence-delimiters in the written English language. Logo uses square brackets because it views sentences as lists of words, and thus requires sentences to be clearly expressed as lists – hence the square brackets, used as list delimiters[143]. I am unaware of the actual motivation behind this use of brackets in the language design, rather than using quotes, but I can point two advantages:

- Since quotes are common in written English, there's a high likelihood that a child will need to issue a command to print a sentence such as «Marty said 'Hello'». This poses a problem, similar to the one I had while writing this paragraph: encapsulating quotes. I resorted to using two different symbols, but in Logo this can be expressed directly, as `print [Marty said "Hello"]`. Typically, programming languages use special characters to overcame this limitation, which results in things like `print "Marty said \"Hello\"`. Obviously, Logo still faces this problem when printing brackets; the advantage lies in brackets being less used in text than quotes.
- Understanding that sentences can be viewed as lists of words is a powerful concept, allowing a child to write programs that handle them as such. This means that imposing from the start the view of a sentence as a list of words can be used to leverage more advanced concepts.

This analysis of quotes allows me to bridge onto the topic of "simplification of syntax" not being a clear requirement definition. I have no intention of questioning whether the option of using brackets instead of quotes is advantageous or not, but rather of looking at it under my proposed view of design of a language for children as providing a metaphoric model. In that sense, the analysis of this option can be completely different depending on whether children are used to reading or writing text employing quotes or not. In the likelihood that most children have encountered quotes before, the option for brackets is a detachment from a previous context of using text, and thus, questionable as a link to previous knowledge. But also commendable, for those reasons I listed above, as a link to further knowledge. Other options could also be debated, *e.g.*:

> *« Logo is consistent and not overly error-prone, although some of the syntax can cause confusion, for example the difference between "`name` and `:name` (`:name` is used to access the value of `name`, while "`name` is used to access the address). »*
>
> *(McIver, 2001, p. 5-21)*

Since the object of this thesis is not the specific analysis of Logo or textual programming, I do not wish to probe these issues of textual-language syntax in further detail. A recent deep analysis at

[143] Braces would have been another option, more closely linked with school math; speculating, there are two possible explanation for using squares brackets instead: one is that braces {} are used in mathematics for unordered collections, and lists are ordered; another is that many 1960s computer keyboards didn't have braces.

this level, and the specification of a textual programming language, GRAIL, which incorporates the results of that analysis[144], can be found in Linda McIver's PhD dissertation (McIver, 2001).

Under another category of analysis, program execution, three of Logo's important features were having well-designed error messages, direct interpretation of code and turtle graphics.

Regarding error messages, if a child, for instance, wrote a non-existent command or mistyped an existing one, Logo provided a more helpful message[145] than a typical opaque remark such as "Syntax error":

```
? pritn Hello World!
I don't know how to pritn
```

Another example is when a child wants to update an array item and forgets to indicate the index number. The message prompts the child to look at the command and detect which element is missing, rather than just finding out "what's wrong". Another important feature of these messages is that they point out the "*particular procedure that complained*" (Harvey, 1997, p. 16).

```
? setitem :mylist [6]
Not enough inputs to setitem[146]
```

These examples also demonstrate the feature I called "direct interpretation of code". Being an interpreted language was not a feature introduced by Logo, but a design choice, between the two existing possibilities (interpreted languages and compiled languages). It was chosen over the alternative, because it allows children to explore the language just by typing a command and hitting the "Enter" key, thus providing an immediate association between a command and its result.

Before presenting turtle graphics, I want to present the final category of analysis: the programming environment. In it, my attention goes to the transparent definition of procedures in Logo.

In most programming languages, one typically writes a procedure (such as the `sayHello` example I presented earlier in this section) in some text file, or in some part of the text which also contains the global program. While coding, the programmer must navigate the entire text, containing the code of the main program and of all subprocedures. This is often helpful, especially in computers with modern text editors, since an experienced programmer can navigate its program at will, searching and analyzing each bit.

In the Logo environment, such a programming style is also possible, allowing the development of large and complex programs[147]. However, for a novice or children initiating programming, this also introduces the necessity to deal with the entire code, which can be a complex task, because the text of code extends quite rapidly as one starts to make more and more procedures. So, the Logo environment included this ability: a child can simply define a procedure such as `sayHello` and it will be stored by the system; from that moment on, it can be included in the program as any built-in command[148].

---

[144] *E.g.*, in Logo, creating an array is achieved as:
`make "mylist array 10`
whereas in GRAIL, it is done as:
`item mylist is array of 10 number`

[145] The error messages presented here were extracted from Berkeley Logo (distribution archives for several operating systems are available for download at http://www.cs.berkeley.edu/~bh/logo.html).

[146] The correctly-formed instruction could be, for instance, `setitem 1 :mylist [6]`.

[147] For instance, by using the instruction `edit procedures`.

[148] This method of invoking procedures is also an advantage Logo got from Lisp. In some languages, a special command had to be used to request the execution of a procedure; and in others, such a command did not exist and the control flow had to jump around using `goto` commands, a technique that led to confusing and error-prone code, as Edsger Dijkstra put forward in a classic paper (Dijkstra, 1968).

This means that there is no need to navigate through the entire code for the program: one can simply focus on the specific problem at hand. In order to analyze procedure definitions, during debugging, Logo provides commands such as `po` (meaning "Print Out"), which shows the code of a specific procedure, `pops` ("Print Out ProcedureS"), which shows the code for all procedures, and `edit`, which allows the programmer to edit the specific procedure he/she is worrying about, rather than navigating the entire code (this command also allows the programmer to edit several procedures at the same time, if so desired). In this fashion, if some procedure is not producing the intended result, it can be edited, tried out, and tucked away nicely.

The final element of Logo I will address is also the most famous one: turtle graphics. This is a subset of Logo commands for producing graphics, based on a physical metaphor: the graphics are drawn by commanding a virtual pointer, instructing it to move, and sometimes drawing lines. This pointer is known as "turtle" (hence the name "turtle graphics"), because it originated as a way to command an actual physical, turtle-shaped, moving toy[149].

Figure 78 – Children working at BBN with one of the first wireless turtle-robots (named "Irving") in the early 1970s

Picture and caption text from: http://web.mit.edu/6.933/www/LogoFinalPaper.pdf, p. 17

> *« The first[150] turtles were actual robots that rolled along the floor. They got the name "turtle" because of the hard shells surrounding their delicate electronic innards. A robot turtle has a pen in its belly, which it can push down to the floor, or pull up inside itself. When the pen is down, the turtle draws a trace of its motion along the floor. »*
>
> *(Harvey, 1997, p. 180)*

[149] Created by Paul Wexelblat (according to a personal interview by Chakraborty *et al.*, 1999, p. 17).
[150] Brian Harvey is referring only to Logo turtles. The first "turtle" robots were actually built much earlier, between 1948 and 1949, by Grey Walter, but there is no connection between them and Logo (Holland, n.d.; Resnick & Silverman, 1997).

The movement options of the physical turtles formatted the command subset used in its control. They can move forward and back, turn around an axis, and draw using a "pen" as described in the previous citation from Harvey. This command set is comprised fundamentally by the commands in Table 10.

Command	Shortened	Description
`forward 12`	`fd 12`	Makes the turtle advance 12 turtle-steps[151] along the direction it is facing.
`back 12`	`bk 12`	Moves the turtle backwards 12 turtle-steps, along the direction it is facing.
`left 12`	`lt 12`	Turns the turtle 12 degrees towards its own left (*i.e.* counter-clockwise).
`right 12`	`rt 12`	Turns the turtle 12 degrees towards its own right (*i.e.* clockwise).
`penup`	`pu`	Lifts the "pen", meaning that from this moment on the movement of the turtle will not produce a line on the screen.
`pendown`	`pd`	Lowers the "pen", meaning that from this moment on the movement on the turtle will produce a line on the screen.

Table 10 – Turtle graphics subset of commands

> *« The crucial thing about the turtle, which distinguishes it from other metaphors for computer graphics, is that the turtle is pointing in a particular direction and can only move in that direction. (It can move forward or back, like a car with reverse gear, but not sideways.) In order to draw in any other direction, the turtle must turn so that it is facing in the new direction. (In this respect it is unlike a car, which must turn and move at the same time.) »*
>
> *(Harvey, 1997, p. 180)*

The original turtle, with all its advantages of being tangible and visible, was a physical artifact, which had to be manufactured and serviced, and could execute only a limited amount of drawing instructions in a reasonable amount of time. A virtual representation would have no such limitations.

The first version of the virtual turtle was developed by Hal Abelson (Chakraborty et al., 1999, p. 19), and the virtual version has become the most well-known one. Figure 79 presents a sample figure drawn with the following sequence of commands.

```
right 30
forward 100
left 120
forward 100
left 120
forward 200
left 120
forward 200
```

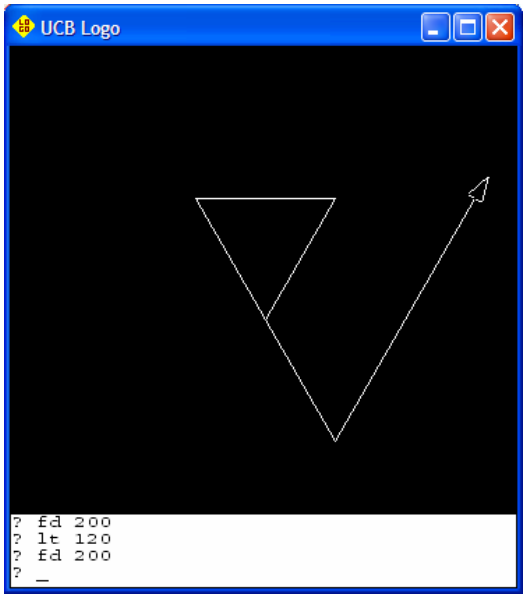


Figure 79 – Turtle graphics

[151] "*Generally, one turtle step is the smallest line your computer can draw. This is slightly oversimplified, though, because that smallest distance may be different in different directions. But the size of a turtle step does* ***not*** *depend on the direction; it's always the same distance for any given computer*" (Harvey, 1997, p. 180)

The importance of turtle graphics comes from providing graphical results for programs. Rather than programming just with words and numbers, children can see graphical representations of their programs, and reason in a graphical way. They can look at the last line in Figure 79, for instance, and try to estimate the distance between the turtle and the smaller triangle; they can associate that distance with its numerical representation; they can try to make the turtle move to the smaller triangle, to be sure whether it is exactly at the same height (vertical position) or not, and from there reason on the nature of equilateral triangles.

While creating graphical shapes, the children can get involved in deep reasoning, using regular programming techniques such as loops, variables, recursion and conditions. Instead of being limited to textual results, turtle graphics allowed Logo activities to be visual. This provides a strong metaphorical link to one's experiences, providing a mapping to visual notions of closeness, farness, rotation, etc.

A problem with turtle graphics, however, is that their simplicity allows educators to look at them just as an appealing way to introduce children to programming, as an end in itself. As I mention in sections 2.4.5 and 4.2.2, there are two main purposes for introducing programming in education: a plain one is to help children control computers, not necessarily as programmers but as users that understand notions of sequence, concurrency, control and decision; a more ambitious one, as people that explore their own thinking while reasoning over a problem in the process of teaching the computer how to perform intended actions. The simplicity and power of turtle graphics has thus been a boon, by allowing a stronger link between children and problems, but also a hindrance, by diverting people's attention from the larger goals (Chakraborty *et al.*, 1999).

> *« (…) intellectual play is the best reason for learning about computer programming in the first place. This is true whether you are a kid programming for the fun of it or an adult looking for a career change. The most successful computer programmers aren't the ones who approach programming as a task they have to carry out in order to get their paychecks. They're the ones for whom programming is a joyful game. Just as a baseball diamond is a good medium in which you can exercise your body, the computer is a good medium in which you can exercise your mind. That's the real virtue of the computer in education, not anything about job training or about arithmetic drill. »*
>
> *(Harvey, 1997, p. 4)*

Squeak Etoys

The Squeak language is an implementation of the object-oriented language Smalltalk, developed between 1971 and 1983 by Alan Kay[152] (Kay, 1993b). The initial development of Squeak took place between 1995 and 1996 (Ingalls et al., 1997), and its aim is to provide children an environment for exploration of mathematics and science, through the construction of computer models of ideas – Etoys[153]. Squeak Etoys is a programming environment for the Squeak language, where children can program Etoys (Kay, n.d.), initially by using graphical commands and then by expanding those with mathematical expressions[154],[155]. Etoys are also sometimes called SqueakToys, due to trademark conflicts (Steinmetz, 2001)

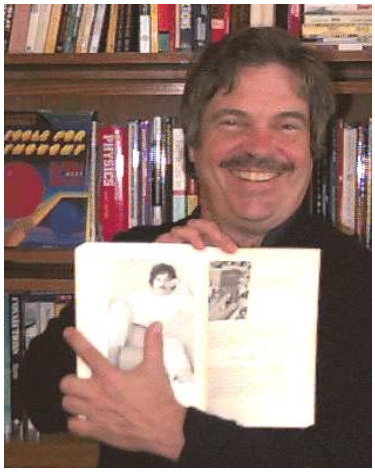


Figure 80 – Alan Kay
From: http://www.cc.gatech.edu/classes/cs6751_97_fall/projects/intelliphone/images/Alan_Kay.gif

> *« Etoys are "science and math" in the large. By this, we mean that science is much more than "scientific knowledge"—it is not only a collection of practices for being able to find out things, but a way to gain some sense of how accurate the findings are likely to be. Mathematics is one of science's most important practices. Here I take the larger meaning of mathematics as the art and skill of being able to reason—that is, to be able to take representations in a language and to show how they are related to other representations in a language. »*
>
> *(Kay, n.d.)*

The object-orientation of Smalltalk (and thus of Squeak) means that every element in a program is an object, an entity that encapsulates data and methods to manipulate those data. There are two main motivations for such an object-oriented design: the first is to allow the programmer to focus on the interaction between objects, simplifying the management of data manipulation across different parts of the program; the second is to simplify the expansion of a program, by duplicating identical objects.

> *« Programming in procedural languages often requires elaborate combinations of procedures, and the complexity of getting such constructs to work increases exponentially with the length of the program.* (…) *Kay opted for a building block approach of layered design, built around the notions of encapsulation and inheritance. Smalltalk was the programming language which finally emerged. »*
>
> *(Kreutzer, 1998)*

[152] Squeak is based on the "release version" of Smalltalk, called Smalltalk-80.
[153] Longer versions of Etoys that "*string several ideas together to help the learner produce a deeper and more concerted project*" are called SimStories (Kay, n.d.). A linked concept is that of narratives that incorporate active (programmed) media constructions, in order to both explain and exhibit ideas. These narratives are called "active essays" (anon., n.d.-1).
[154] If one so desires, the environment also allows direct editing and creation of textual Smalltalk code, thus allowing an advanced programmer to use the full power of Smalltalk.
[155] "*The Disney Squeak team developed the infrastructure and ideas for Squeak-based learning environments. Alan Kay originated many of the general concepts and project ideas, and these have been realized and implemented brilliantly by Scott Wallace, John Maloney, Ted Kaehler, and Dan Ingalls. Kim Rose has organized experiments in classrooms and sought the system improvements needed to make them feasible. BJ Conn and Kathleen Brewer have designed projects specifically for children, and their work has helped the programmers to improve the system. Meanwhile, Mark Guzdial at Georgia Tech has been using Squeak with adult learners.*" (Steinmetz, 2001).

The Squeak Etoys method of expressing a program is to allow children to work on objects drawn by them: resizing, rotation, etc. While doing so, they can see numerical and textual "properties" for the object. For instance, Figure 81 presents a hand-drawn car surrounded by its set of controls, common to all objects. One of those controls is "heading", and since the car in the picture has been slightly rotated to the left, its value is now -2.

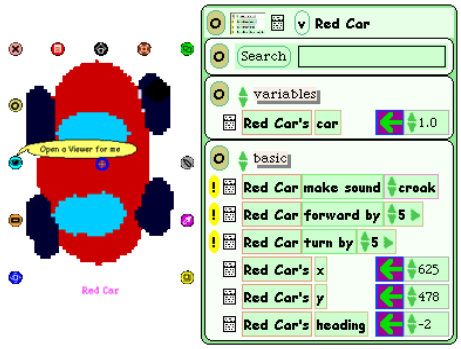


Figure 81 – Etoys object properties
From: http://www.squeakland.org/school/drive_a_car/images/StandardViewer.gif

While the car is rotated, this value is updated in real-time; and if the numerical value is changed directly, the car also rotates in real-time, accordingly. For young children, "*the textual form of these properties doesn't look as exciting as the iconic car that can be directly manipulated* (…). *But, the symbols can make the car do things!*" (Squeakland, n.d.).

The educational rationale behind this is to build on the educational ideas developed in the Logo culture and turn the computer programming environment into a place following the educational ideas of Maria Montessori (presented in this thesis in section 4.1.1).

> *« Maria Montessori* (…) *saw how to let children exercise their wants and still get them to learn what we think they need. Her great idea starts with the observation that children are driven to learn their immediate environment and culture through play* (…). *She realized that what children want is not always the same as what they need — especially in 20th century Western culture.* (…) *Her genius was to devise a special environment and artifacts that look like toys to children — catering to their wants — but with beautiful side effects that help them learn what we adults think they need. »*
>
> *(Kay, n.d.)*

The children can move the car directly, but the numerical properties allow them to connect their actions to notions such as negative numbers (for turning counter-clockwise; positive number turn clockwise), and other mathematical ideas.

In order to move beyond mere editing of properties, the child can use some properties that define behaviors (those with an exclamation point in Figure 81 and Figure 82). The commonly-used Squeak example of programming the driving of a car renders this clear: by clicking the exclamation point for the property "`forward by 5`", the car moves forward 5 points. Such behaviors can be dragged together to the desktop, to create a script. In Figure 82, the script reads:

```
Red Car script1
Red Car forward by 5
Red Car turn by 5
```

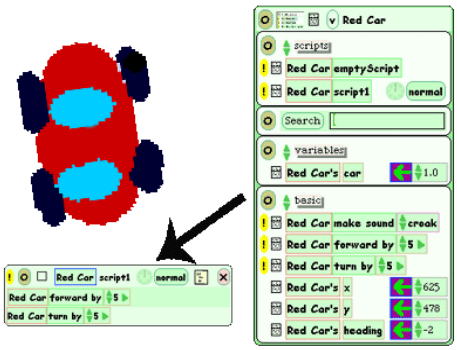


Figure 82 – Etoys script creation
From: http://www.squeakland.org/school/drive_a_car/images/drag_out_tiles.gif

By clicking the clock icon (pointed by the arrow in the figure), the script starts running, *i.e.* the car advances 5 points and then rotates 5 degrees to the right, over and over again, as the "clock" ticks. The resulting motion pattern is a circle. This can be better visualized by changing the property "`penDown`" to "`true`" (the primitives used in this example are identical to Logo's commands for drawing with Turtle graphics). Figure 83 presents the resulting circle (the car keeps moving in circles as long as the script clock is ticking).

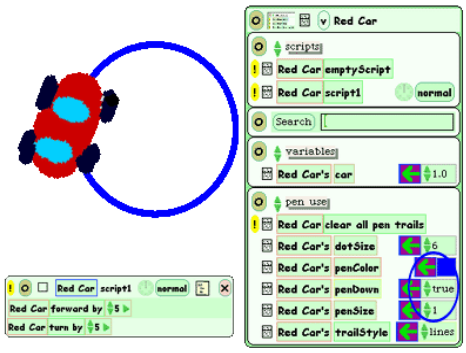


Figure 83 – Circular motion in Etoys
From: http://www.squeakland.org/school/drive_a_car/images/pen_true.gif

This example also serves as a presentation of important features of the Squeak Etoys environment in terms of the understanding of the execution of the program: the program can be edited as it is running. For instance, a child can change the amount of turn from 5 to some other number, and immediately see the resulting behavior in the car's path. The pen can be turned on or off, or erased, without having to stop the program, edit the script, and run the program again.

Any object can have several scripts, running concurrently, and these can be turned on, turned off or be subjected to some condition (*e.g.*, the mouse button being depressed), independent of one another, while the program is running. This allows a child to more simply understand the impact of each script change – and of each script – in the behavior of the overall program.

In terms of the child-orientation of the programming environment, Squeak Etoys has several advantages and a few drawbacks. An advantage is the already-presented feature of allowing the objects' properties to be combined graphically, simplifying tasks such as the assignment of values from different sources. This can even be used between different objects: for instance, in Figure 84 the car is no longer turning by a fixed value such as "5", but rather by the value of the "`heading`" property of another object – the steering wheel. A child can do this assignment of variables among objects by dragging the Steer's "`heading`" property to the place where the "`5`" laid, in the car's script.

Figure 84 – Etoys: controlling a car with a steering wheel
From: http://www.squeakland.org/school/drive_a_car/images/turn_by_steer.gif

Another important feature of the programming environment is the built-in ability to allow different programmers to cooperate in real-time over the same program. The Squeak system includes a registry where children "*who have already done projects can sign up to be mentors and colleagues*" (Squeakland, n.d.). By communicating with other children that have worked on similar projects, a programmer can ask for help and advice, and those can include actual remote manipulation of the local Squeak objects and their properties and scripts.

Finally, Squeak includes a simple graphic editor, integrated in the programming environment itself. In this way, a child can quickly generate a picture object for programming with it, keeping his/her focus on the programming environment. There is no need to specify which areas of a picture

drawn elsewhere are to be transparent, or going through any procedure for import, or for locating files in the disk drives[156].

Regarding drawbacks, while Squeak itself is a powerful general programming language, the Squeak Etoys environment focused on allowing the programming of specific math and science activities, rather than on providing a complete metaphor for mapping the entire Squeak programming model. As a consequence, the Squeak Etoys environment does not allow a programmer to progress smoothly to more advanced programming techniques.

> *« A current "drawback" in the Squeak etoys is that they were an experiment aimed at a particular age group -- 9 to 12 year olds -- for particular purposes -- about 50 etoys in math and science. The good news is that, in this range, they really work extremely well, and are learned by virtually all children and adults who try them. That was the experiment. The downside is that there is not a lot of extension in the current system, and it gets awkward for older children and experts. »*
>
> *(Kay, 2003a)*

Another drawback is that the number of properties available for each object is possibly daunting, and its navigation can be troublesome, lest one invests considerable time in its exploration – or has constant access to a mentor, expert or reference guide. The problem with large sets of properties is that people only have two ways of solving a novel problem on their own, when well-known object properties provide no apparent solution: either have an insight from the overall knowledge of properties or browse them all in search of one that triggers such an insight. While an appreciation of this as overly complicated or acceptably simple is highly subjective, the main point is that acquiring such insights is likely to be harder when the programmer has to navigate dozens of different properties. The possibility of losing focus on a specific problem is, in my view, quite high.

Figure 85 presents an example of this drawback: a partial list of the properties for a Squeak Etoys object, which spreads from top to bottom of a maximized Squeak Etoys window in my system[157]. The properties grouped under 4 categories are visible, and as can be seen from the superimposed list of categories, many others are available. A

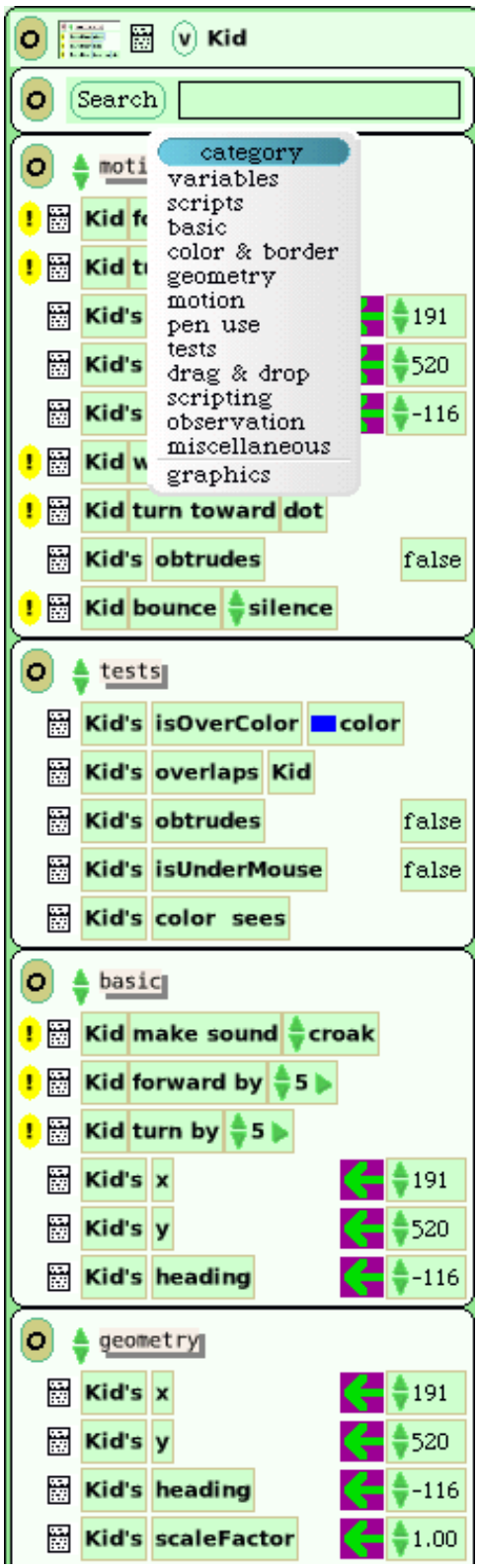


Figure 85 – Etoys: partial property list and categories of properties

[156] Information organization in a disk drive is an important computer skill; adults and children should be able to navigate efficiently in their computer's file systems. However, being able to work without little knowledge and skills in this area means that children can start programming and rendering computers useful without having to acquire file navigation as a prerequisite.

[157] Windows XP at 1152 by 864 pixels, of which 64 rows are used by the taskbar.

similar observation could be made (and has been made[158]) regarding the "halo" of options available for each object, visible in Figure 81, which is composed of no less than 12 icons. But since these icons are all in plain view at all times, and balloon help is provided, this is more a matter of balance between the effort expected of novice users and the convenience for experienced users.

> *« (…) you can set a preference to have the halo of handles come up on mouseover: many teachers use this, some don't. There is balloon help on most things in the interface that is delayed one second so it doesn't get in the way of those who have become more expert. IOW[159], there are things you can do to deal with these problems. E.g. most of the many hundreds of children we've had experience with don't have any problems here, so I think it's more a style of approach that is affecting things. »*
>
> *(Kay, 2003b)*

In noting these drawbacks, I am merely centering on issues related to the design and conception of the programming environment, not on its implementation, which can always be improved. It is interesting to note that the issue I mentioned above regarding navigation among objects' properties could be similarly raised regarding the need to know or browse lists of textual commands in a typical language such as Logo. Similarly, the scripting process can be found simple by some users and complex by others. The overall dilemma is mentioned by Squeak's creator Alan Kay in a 2003 e-mail message:

> *« Neither the current Squeak syntax nor the Logo syntaxes are ideal for children and other end users. We really should be thinking about what improvements in UI should be made to help them. Andreas Raab has pointed out that the syntax of a programming language is actually part of its user interface -- and I think this is a really important observation. If we look at the difficulties of having children understand (say) parameter passing in Logo, we should be thinking about how it should look. »*
>
> *(Kay, 2003a)*

In reading this, one should bear in mind that a major contribution of Logo was its simplification of syntax; the important acknowledgment is that typical textual and visual syntaxes for general programming languages are still far from being ideal for end users. Radical departures from these typical syntaxes are thus important possibilities to consider, moving on in the path to powerful end-user and children programming. ToonTalk, described in section 3.3.5 and used in the field work supporting this thesis, is an example of such a radical departure, by immersing the programmer in the virtual world of a computer game.

> *« I'd like to urge that people consider radical alternatives to textual syntaxes. Syntaxes based upon diagrams or pictures have had limited success either because they weren't very general or, despite being visual, they were too abstract and difficult for children and other end users. But there are alternatives to text (even with tiles) and to pictures. My ToonTalk (www.toontalk.com) is an example. The equivalent of a Squeak method in ToonTalk are the actions you train a robot to take in a game-like animated world. Syntax isn't a good way to think about such things. What is the syntax of showing someone how to tie a knot for example? »*
>
> *(Kahn, 2003)*

[158] *E.g.*, « *My first impulse would not be to alt-left click an object to pull up a halo and then click the blue eyeball to pull up one object's viewer at a time. Every time I explain this process, people just look at me or ask "And who came up with THAT?"* » (Basu, 2003).

[159] Commonly-used Internet lingo acronym for "in other words".

Stagecast Creator

This programming system was developed by David Canfield Smith and Allen Cypher, mainly between 1990 and 1997. Originally, at Apple Computer Inc., it was called KidSim (Smith *et. al.*, 1994), and then renamed Cocoa (Smith *et al.*, 1996). It is now a commercial product of Stagecast Software, Inc., under the name Stagecast Creator (Smith *et al.*, 2001).

Figure 86 – David Canfield Smith & Allen Cypher
From: http://www.cs.umd.edu/hcil/muiseum/canfield/davepicture.jpg and http://www.acypher.com/images/ACypher.gif

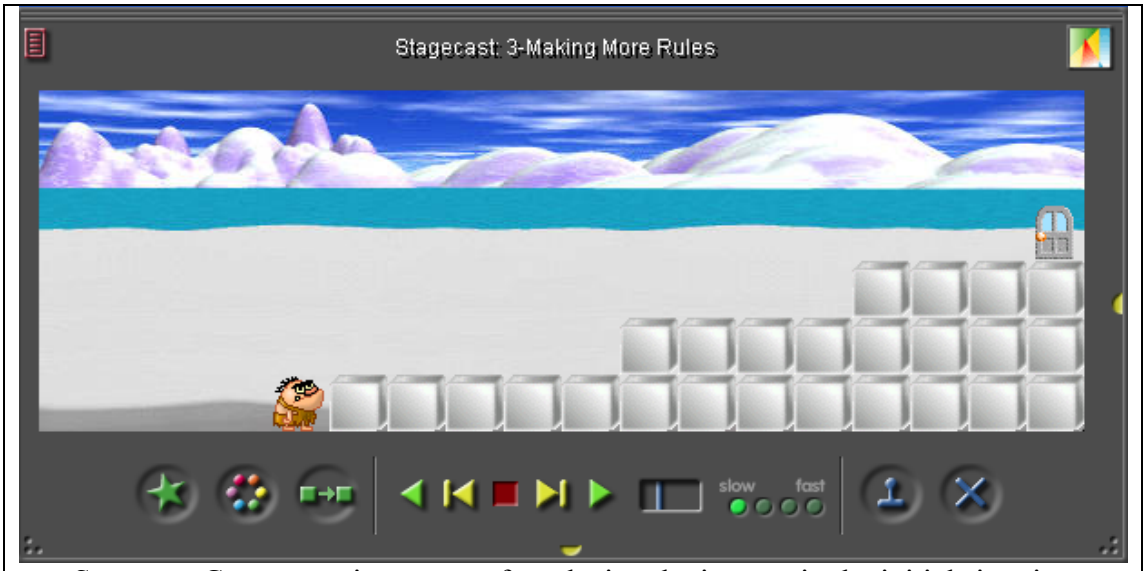
Smith & Cypher believe textual programming to be inappropriate for use by people without an interest in the actual technical aspects of computer programming itself, and thus Stagecast Creator was developed as a visual programming language, at least in its core programming method[160].

> *« Our first insight was that language itself is the problem and that any textual computer language represents an inherent barrier to user understanding. Learning a new language is difficult for most people. Consider the years of study required to learn a foreign language* (…)*. A programming language is an artificial language that deals with the arcane world of algorithms and data structures. We concluded that no conventional programming language would ever be widely accepted by end users. »*
>
> *(Smith et. al., 2001, p. 9)*

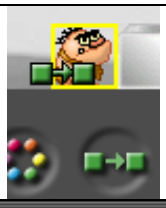
Following this reasoning, the emphasis of Stagecast Creator is on the simplification of expressing the program. Smith & Cypher attempt to "abolish" syntax, at least in terms of its conscientious perception by the users, by basing programming on visual rewrite rules, *i.e.*, specifying graphical beginning and ending situations, rather than the actions that are to be performed by the language itself. In this sense, there is a connection between Stagecast Creator and the idea of constraint programming, with its pre- and post-conditions. Figure 87, on page 152, presents an example of rule-definition in Stagecast Creator.

> *« It's no good allowing users to create a program easily and then require them to learn a difficult syntactic language to view and modify it* (…)*.* (…) *Creator does in fact have a syntax – the lists of tests and actions in a rule – but people can create programs for a long time without even being aware of this syntax.* »
>
> *(Smith et al., 2001, p. 10)*

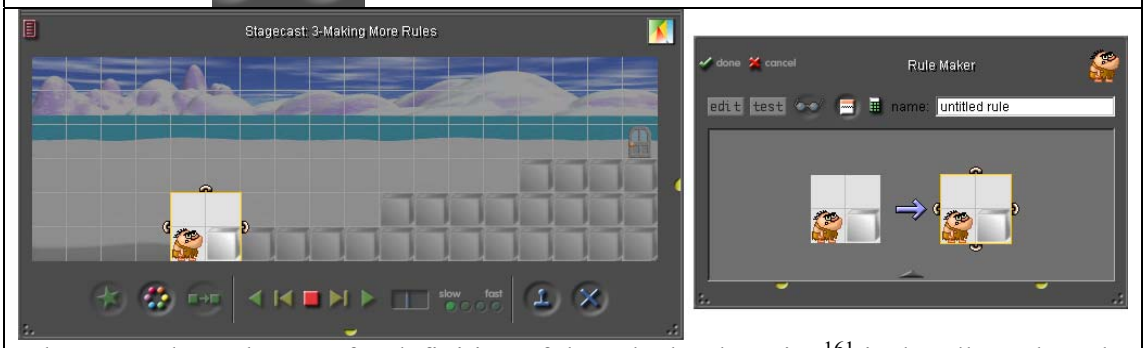
[160] Parts of Stagecast Creator, such as conditions and usage of variables, require some textual syntax, as is mentioned further ahead in this section.



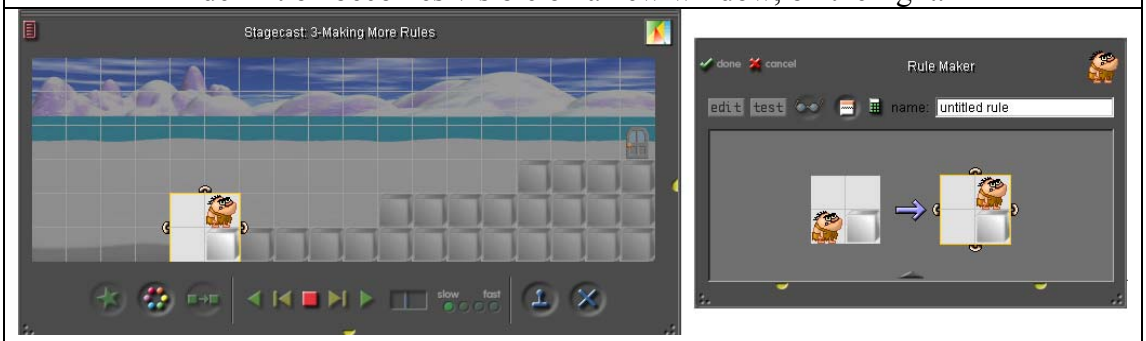
Stagecast Creator environment, after placing the iceman in the initial situation



Using the "rule" tool on the desired element (the iceman).



The user selects the area for definition of the rule, by dragging[161] its handles. The rule definition becomes visible on a new window, on the right.



By dragging the iceman to the top of the ice block, the rule is defined. This can be done in either the main window or the rule window. The definition becomes complete after clicking on the button labeled "close" (with the green checkmark)

Figure 87 – Defining a rule in Stagecast Creator (climbing on top of an ice block)

Screenshots from the tutorial available at www.Stagecast.com

[161] As the image shows, Stagecast Creator uses a grid structure for defining rules. This has the advantage of simplifying the specification of starting conditions, since all elements must be located at well-defined positions. However, this also means that when using this visual technique "*all motion is stepwise and movement at any angle other than to one of the squares surrounding the current position is impossible. Similarly changing the visible representation of a character happens abruptly*" (Sheehan, 2004).

The user must understand this graphical syntax in two situations:

- in order to change the ordering of the rules (Stagecast applies the rules in sequence for each element, not concurrently, so the order of their execution may need to be re-arranged, in order to attain the desired effect);
- in order to include conditions not represented in the graphical environment (for instance, a mouse-click, a key-click, or a variable value).

Figure 88 – Conditional rule in Stagecast Creator

As an example, Figure 88 presents a conditional rule: the bat will move up one empty square, but only if the user clicks on it with the mouse cursor. The symbol to the right of "is clicked on" means "bottom square of the two squares of the rule".

Textual syntax is also required when variables have to be used. For instance, Figure 89 is part of a Stagecast tutorial and explains how to add an action to a character to change the value of a variable. In this case, the magical power "zoom" is being assigned to the variable "Magical Power", when the magician moves over a magical potion.

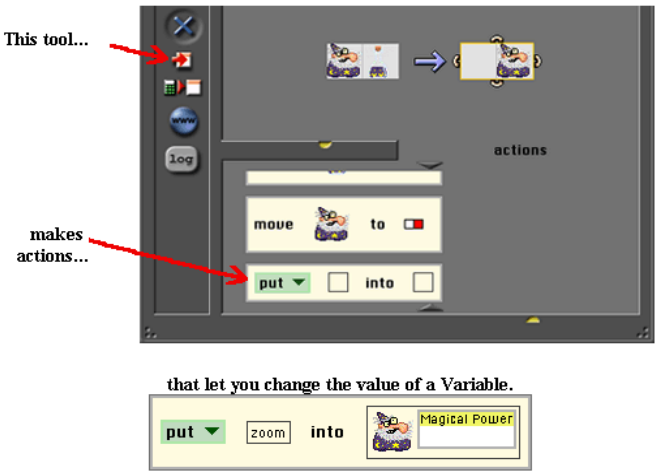


Figure 89 – Updating variables in Stagecast Creator

While all these features are relative to the expression of the program, Stagecast Creator also includes some features useful for understanding the program's execution. When a program is not behaving as the programmer intended, it can be paused, and the rules for each element inspected.

This inspection is performed at two levels: a global view of the element's rules (Figure 90), and a test analysis of an individual rule (Figure 91). For instance, in the global list of rules for the iceman (presented in Figure 90), the red "light" near each rule report that it cannot be fired. (And a green light would report a rule that can indeed be fired.)

By double-clicking on the specific rule that the programmer intended to have executed, the Stagecast Creator can test the rule and show the program which parts are valid or invalid.

In Figure 91, the rule defined in Figure 87 is being tested in a different situation: the iceman is facing a pile of two ice blocks, instead of just one.

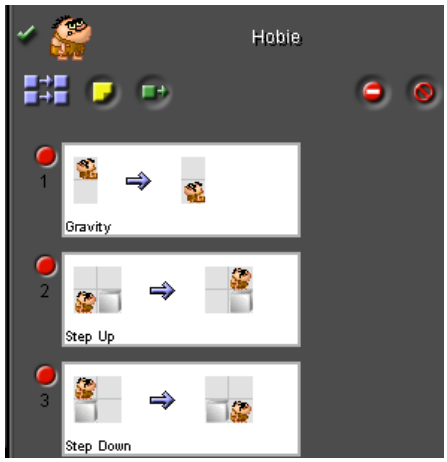


Figure 90 – Analyzing which rules apply or don't apply in Stagecast Creator

By testing the rule, Stagecast Creator shows the programmer that the problem lies in the top-rightmost square, which should be empty but isn't (it's occupied by an ice block).

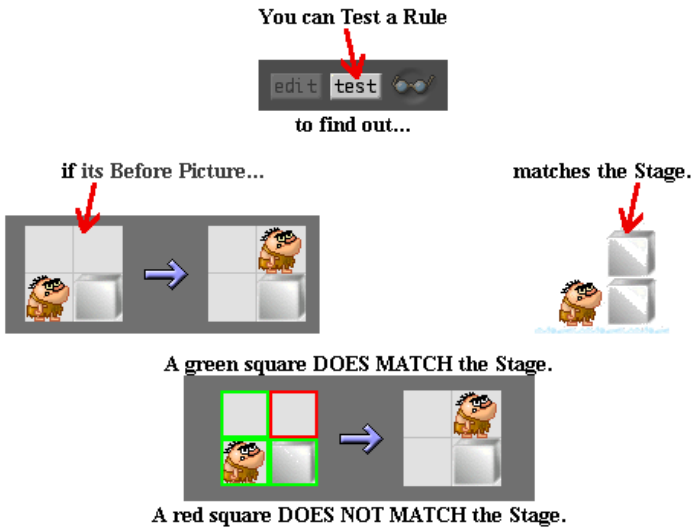


Figure 91 – Testing a rule in Stagecast Creator

Regarding the child-orientation of the programming environment, one important feature of Stagecast Creator is its strong connection between the expressed program and the running program. All programming takes place alongside the area (called "stage") where the program will be running. And the basic actions of programming can be performed in the same fashion either on the stage or on the rule-definition window. For instance, the dragging of the iceman presented in Figure 87 can be performed on the stage or in the rule-definition window, and the resulting program will be the same. There is thus a smooth transition between the execution area and the rule-definition area. A similar situation occurs in Squeak Etoys, where some graphical properties can be changed either in the picture object or in the property box (as explained in p. 147); however, these are different representations[162]. In Stagecast, the representations in the rule-definition window and in the stage are identical.

This smooth transition is a major advantage when solving programming errors – bugs. Stagecast allows the stage to be paused, and the rules to be inspected regarding the current state of execution of the program. This is a classical debugging environment and technique: interruption of the program execution in order for variables and other elements of the program's running state to be analyzed. Since the rule-definition area is the same as the programming area, there is not a noticeably separate "debugging environment" to be learned, with associated concepts of tracing and stepping. Rather, the place were a rules was defined is also the place where it is analyzed, using the methods just described above.

A smaller feature of Stagecast Creator is that it also allows children to draw, in the programming environment itself, the "characters" they want to program, like the Squeak Etoys environment. As mentioned in footnote 156, (p. 149), this allows greater focus in the programming task, diminishing the switching back and forth between the programming environment and a graphical editor, and simplifying file-management problems.

Stagecast Creator is quite adequate and efficient for developing various kinds of simulations, games, and other programs. However, its visual rules are not a general-purpose programming approach. For several situations, such as number or text processing, one needs to resort to more traditional programming styles, such as the one presented in Figure 89. This limitation also applies to some fairly common visual behaviors, such as programming an alien spaceship to fly in circles, making a ball rebound in a realistic manner, or describe how a monster should chase a player not in its immediate vicinity[163].

[162] As pointed out in that section, the developers of Squeak Etoys see this difference of representations as an advantage, a way to make a smooth transition not between programming and debugging activities but between graphical and numerical representations.

[163] During the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01), in Stresa, Italy, I inquired Allen Cypher about this latter problem (I was wondering if one might program ultra-fast, invisible "scout sprites" to locate objects), and he mentioned Stagecast was considering implementing a "radar" tool.

Other child-oriented programming systems and languages

In May of 2003, Caitlin Kelleher and Randy Pausch presented a large survey of programming environments and languages for novice programmers (Kelleher & Pausch, 2003, henceforth "K&P"). It included both children-oriented programming and programming for inexperienced adults, and was the basis for the list of systems I present in this section. However, from that survey I have selected only programming environments and languages that are either aimed specifically for children, or that possess important child-related features.

K&P organize the systems under taxonomy of purposes and goals, rather than taxonomy of actual technical aspects or features. Such an approach to taxonomy is necessarily subjective. Since this section aims to provide an overview of the field of programming for children, I elected to employ subjectivity in a different fashion: the systems are simply grouped to convey an overall look to the reader:

- Simplified-syntax textual programming languages
- Programming with sequences of icons
- Programming inspired by Logo's Turtle Talk
- Programming of social interactions
- Programming with physical objects
- Graphical programming by demonstration
- Games that include programming activities
- Game editors and game-creation systems

Several languages and environments were developed since that survey was conducted (Concurrent Comics, Eco-Pods, MediaStage, Mulspren, PervoLogo/MicroWorlds JR, Physical Programming, Point & Click Development Kit, Scratch, System Blocks, Tangible Programmable Blocks, Topobo), and some were omitted there, but I believe them to be useful in this overview of the genre (AlgoCard, FASTR, Function Machines, Game Maker, Games Designer, Icicle, HURG, Lemmings, Logiblocs, Multimedia Fusion, RCX Code, RoboLab, RobotWar, TEACH, The Game Factory). In this section, I include these extra systems, as well as information and figures about systems mentioned by K&P that are most representative of each of the above groups.

SIMPLIFIED-SYNTAX TEXTUAL PROGRAMMING LANGUAGES

These programming languages aim to simplify the process of expressing or understanding textual programs, usually by simplifying the syntax or by providing support for associating the textual program with its execution.

Smalltalk (1971)

(Kay, 1993b)

Smalltalk was an early attempt at redesigning the way computers were programmed. At the time, other languages intended for novices, such as Logo and BASIC, focused on the adaptation of traditional programming, which was generally devised for mathematical calculations. Rather, Smalltalk was designed with the concept that normal, non-technical users could use personal handheld computers[164] themselves: "*millions of potential users meant that the user interface would have to become a learning environment* (…) *needs for large scope, reduction in complexity, and end-user literacy would require that data and control structures be done away with in favor of a more biological scheme of protected universal cells interacting only through messages that could mimic any desired behavior*" (Kay, 1993b). This eventually led to a language where the concept of object-orientation was first implemented, something that is nowadays common in many professional programming languages such as C++, Java, and Smalltalk-80.

Smalltalk originated Squeak and Squeak Etoys, presented in section 3.3.4, from p. 146 onward.

Boxer (1986)

(diSessa & Abelson, 1986)

Boxer is an effort by Andrea diSessa under the general aim of rendering available to normal users "*the possibilities of the computer screen itself as an expressive medium*" (diSessa & Abelson, 1986, p. 859).Under diSessa's view, that aim means that users should be able to program the computational medium in an easy way: for example, if a user wanted to produce an optics book in this medium, "*all elements of the book, beyond simply entering the text, would be created through some sort of programming*" (*id.*, p. 860).

In Boxer, everything on the screen is inside a rectangular region called a "box", and boxes can be placed inside other boxes, for organization of the code as its complexity increases[165]. Boxes can contain code, graphics, or text, and make reference or use the contents of other boxes.

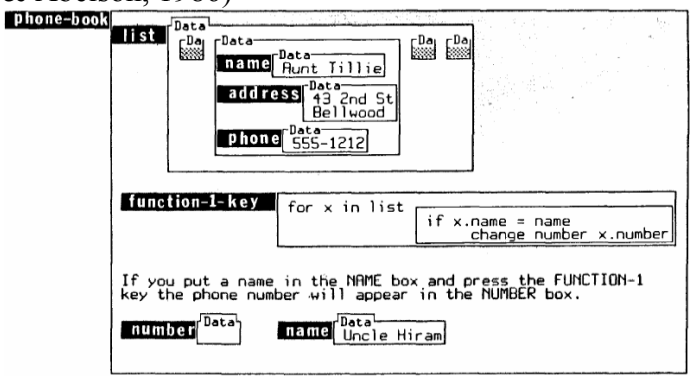


Figure 92 – Boxer program for finding phone numbers

From: diSessa & Abelson, 1986, p. 863

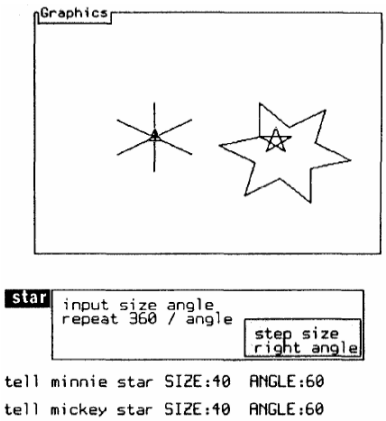


Figure 93 – Boxer program: controlling graphical elements

From: diSessa & Abelson, 1986, p. 865

[164] At the time, names for this idealization of Alan Kay included "KiddiKomp" and "Dynabook".

[165] This technique inspired, for instance, the programming environment LiveWorld, also mentioned in this section (Travers, 1994b). There is also a similarity with the ToonTalk method of placing code behind pictures and then organize it in different levels by placing those pictures behind others (*vd.* section 3.3.5, from p. 195 onward).

Playground (1989) & Liveworld (1994)

(Fenton & Beck, 1989; Travers, 1994a; Travers, 1994b)

Playground is a system in which children create graphical objects and assign behaviors to them. This assignment is done by using a textual scripting language, designed to be similar to the English language. According to K&P, it is very easy to create and interact with graphical elements in Playground, but it's much more difficult to interact with the attributes of each object.

Liveworld is an evolution of the concepts developed in Playground: it employs a graphical interface for the rules and attributes of each object, aimed at diminishing the problem of accessing attributes and managing rules.

Liveworld has another important distinction, in that it employs Lisp[166] as the programming language for the scripts (extended with some primitives for message-passing and accessing attributes).

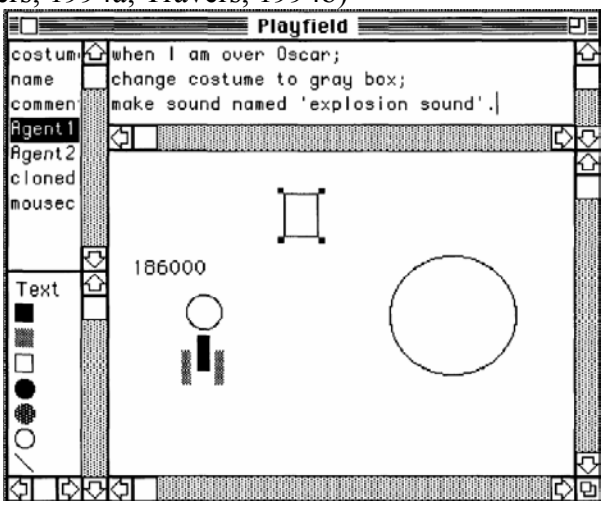


Figure 94 – Playground programming environment

From: Fenton & Beck, 1989, p. 124)

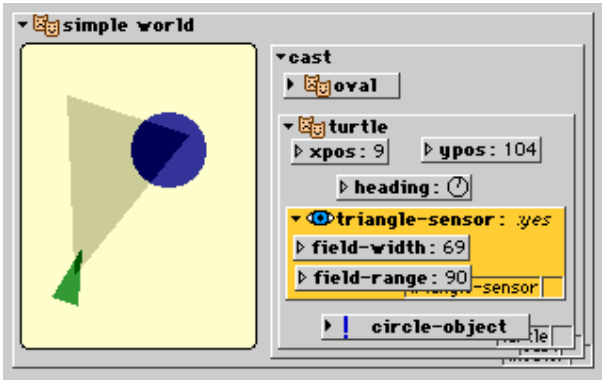


Figure 95 – Liveworld environment

From: http://xenia.media.mit.edu/~mt/liveworld-simple.gif

GRAIL (2001)

(McIver, 2001)

This is a textual programming system based around the hypothesis that the complexity of programming is greatly due to the complexity of the textual syntax and awkward grammatical constructs. GRAIL attempts to have an "obvious" syntax and maintain a close relationship with the meaning of each command as a word in the English language. GRAIL programs are at first glance very similar to languages such as Logo or Pascal, but contain many small differences. An example was already mentioned in footnote 144, in page 142; another, lengthier example is presented at the right (from McIver, 2001, appendix B).

```
type student has
     field name is text
     field id is number
end student

item myStudent is student

myStudent's name ← "Linda"
myStudent's id ← 12345678

write "Name: ",myStudent's
name, "ID: " ,myStudent's id
```

Table 11 – GRAIL code for defining a record type and a record, assigning values, and displaying them on the screen

[166] Already mentioned in the section describing the Logo language, on page 139.

HANDS (2002)

(Pane, 2002)

This is a system developed as part of John Pane's PhD dissertation, aimed at programming by 10-year old children. Its development was based on research knowledge from the fields of human-computer interaction and difficulties met by beginner programmers.

"*HANDS provides queries and aggregate operations to match the way non-programmers express problem solutions, and includes domain-specific features to facilitate the creation of interactive animations and simulations*" (Pane, 2002, p. 3).

The actual metaphor used is that of a deck of cards being handled by character named Handy the Dog (Figure 96, top left). The back of each card displays a picture of the object to which it relates; the front contains data.

The programs are represented by laying cards on the table in the programming environment, and adding textual properties to them.

By using a natural-language textual syntax, similar to GRAIL (mentioned in this section, on p. 157), the programmer "tells" Handy the Dog about rules that govern the objects represented by the cards laid on the table.

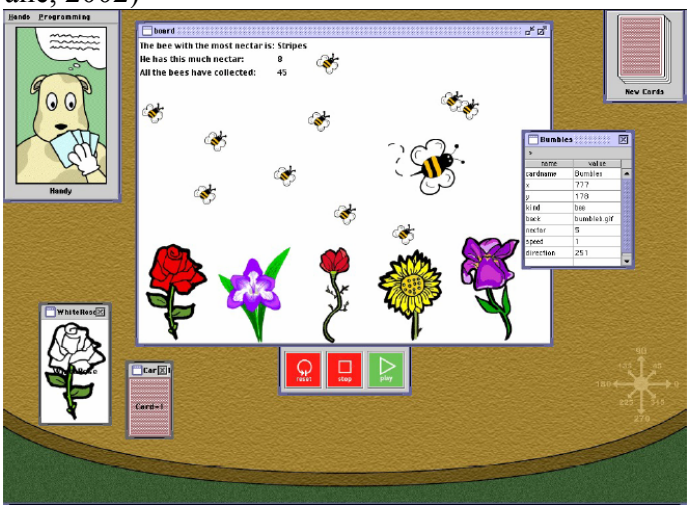


Figure 96 – HANDS programming system

From: Pane, 2001

« Open Handy's Thought Bubble. Use the ***New*** *button to make a new rule. Type* `when anything happens` *for the event. This way Handy will do the action* ***all the time*** *when anything happens on the game board. We know we want Handy to figure out the* `CardWithLeast nectar of all flowers`*. We would like him to put the answer into the back property of the* `leastFlower` *card. To do this, we will ask him to:*

`set` ***`leastFlower's back`*** `to` ***`CardWithLeast nectar of all flowers`***

Put that inside your new rule. The rule should look like this:

```
when anything happens
     set leastFlower's back to
     CardWithLeast nectar of all
     flowers
end when
```

Press ***Check****, to make sure the new rule is correct. If it is, close Handy's Thought Bubble. »*

(Pane, 2002, Appendix G, p. 11)

PROGRAMMING WITH SEQUENCES OF ICONS

These programming environments and languages try to be more accessible to children by employing graphic entities (icons) instead of textual ones. These icons are typically from a standardized set, but in some systems (Show and Tell, for instance), the children can define a new graphical symbol/icon for a program part he/she defined.

PLAY (1986)

(Tanimoto & Runyan, 1986, according to Kelleher & Pausch, 2003)

"*Play is a system designed to allow preliterate children to create graphical plays using an iconic language. Stories consist of a linear sequence of actions that is displayed at the top of the screen, above the story's stage, as a sequence of icons similar to a comic strip. The character, what the character should do, and one additional piece of information, typically a direction to move, all selected from menus, specify each action in the story.* (…) *Play does not allow children to use more complicated control structures such as loops and conditionals or define procedures*" (Kelleher & Pausch, 2003).

Function Machines (1988)

(Feurzeig, 1996; *id.*, 1999)

This visual programming language was developed with the specific aim of being used for mathematics education.

Functions, algorithms and models are all conceptualized as "machines", represented as icons resembling puzzle pieces, with "input hoppers" and "output spouts" (Figure 97). These machines can be connected among themselves or contain other machines.

The program in Figure 97 prints the sequence for the rule "n/2 if even, 3n+1 if odd" for a given number (the machine named "Type Line" prints the value it receives in an output window). The function machines "n/2 or 3n+1" does exactly that, and its contents are defined in terms of internal machines, as seen in the window at the bottom of that figure.

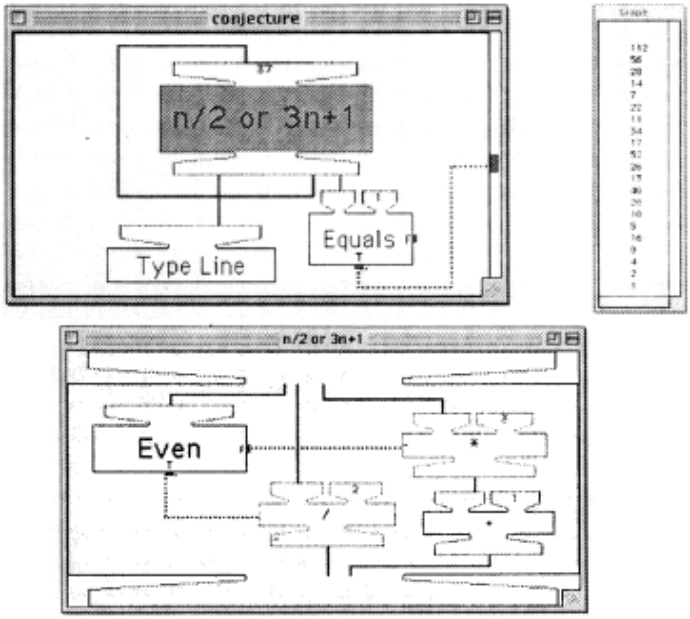


Figure 97 – Function Machines program

From: Feurzeig, 1999, p. 44

Show and Tell (1990)

(Kimura *et al.*, 1990)

This is a language based on dataflow graphs, using visual representations were arbitrary images can be used as function names (in Figure 98, the images are handwritten names). A program is a series of connected boxes (empty boxes represent variables). Some boxes represent operations for conducting arithmetic, input and output, etc.

Children can program by drawing icons for procedures and using boxes to define the operation of those procedures.

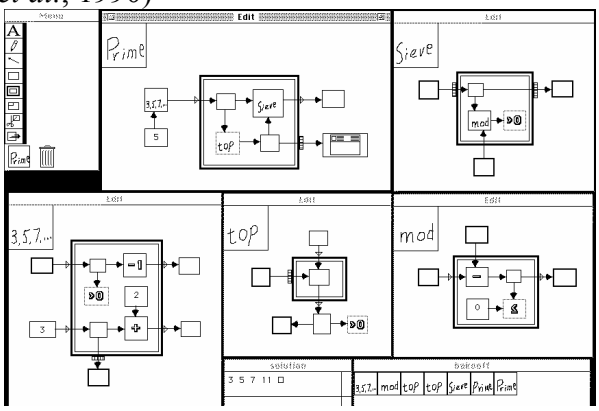


Figure 98 – Show and Tell programming

From: http://www-2.cs.cmu.edu/~wjh/papers/bakeoff.20.gif

PervoLogo (1995) / MicroWorlds JR (2004)

(Soprunov, 1996; INT, 2005; LCSI, 2004)

MicroWorlds is a popular version of the Logo language presented earlier on pp. 139-146. In 1995, the Russian institute INT developed a version called PervoLogo (ПервоЛого), specifically "*for the pre-school and initial school education (4 years and older)*" (INT, 2005). Having evolved across three versions, it was released in English in 2004 as MicroWorlds JR (LCSI, 2004).

It focuses on the control of Logo turtles, of which there can be various, at the same time, with different shapes (in Figure 100, three are visible: two bird-shaped "turtles" and a ship-shaped "turtle").

Children can open an icon list for each turtle and drag and drop icons into them, in a fashion similar to Magic Forest (*vd*. Figure 88).

"[Children] *control the turtle with iconic commands that tell the turtle to move, pivot, pause, put its pen down or pick it up, change and stamp its shape, play music, and much more*" (LCSI, 2004, p. 3).

Figure 99 – PervoLogo opening screen and newer versions

From: http://www.int-edu.ru/logo/pl.gif
http://www.int-edu.ru/logo/band.gif
http://www.int-edu.ru/logo/Pl3.gif

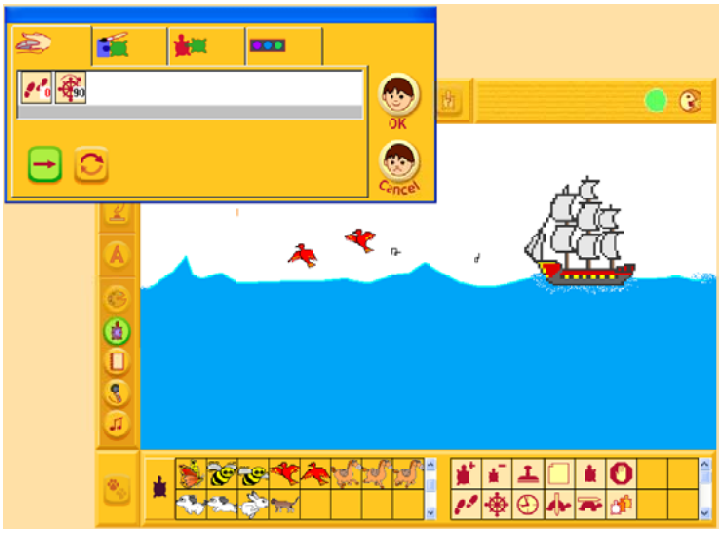


Figure 100 – MicroWorlds JR environment and programming

LogoBlocks (1996)

(Begel, 1996)

"*LogoBlocks is a graphical programming language intended for use with the Programmable Brick*[167] *developed at the Epistemology and Learning Group in the MIT Media Lab. The Programmable Brick is a small handheld computer that can control up to four motors and can read values from six sensors. Children can make LEGO creations like cars and robots, attach motors and sensors to them, and program the Programmable Brick to actuate and control their creation.*" (Begel, 1996, p. 6). LogoBlocks can also be used to control LEGO's RCX blocks (Figure 101 and Figure 102).

Figure 101 – Programming a LEGO motor in LogoBlocks

From: Walters & Beaver, 2004, p. 5

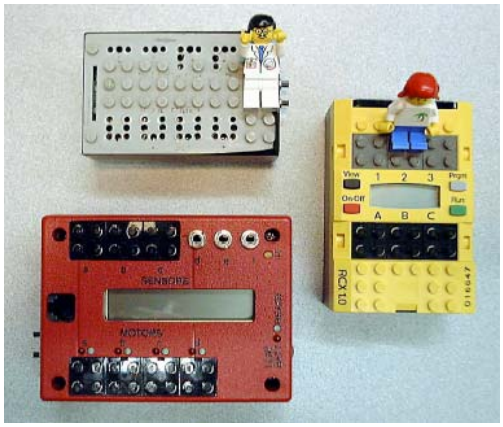


Figure 102 – Gray brick, Programmable Brick, Lego RCX

From: McNerney, 2000, p. 28

The language originally designed for control of these Programmable Bricks was a version of Logo, called BrickLogo. Previously, a prototype language called Youngster, based around a similar approach had also been developed (Studer *et al.*, 1995). LogoBlocks is a visual version of BrickLogo, and its programming environment is the one presented in Figure 103.

The left-side shape palette represents commands that can be dragged to the programming area. Some represent primitives of the language; others represent motors, sensors or numbers.

The code in Figure 103 includes a start sign ("inicio"), an "always" loop ("siempre"), if-then-else conditions ("si-entonces-no"), two blocks for reading sensors, comparisons with fixed numbers, and blocks for controlling motors **a** and **b**.

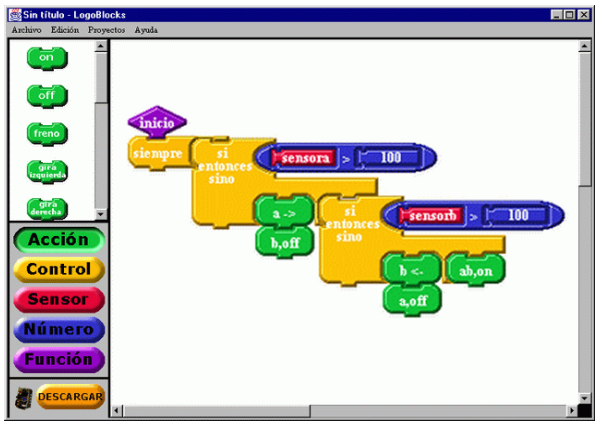


Figure 103 – LogoBlocks programming environment (Spanish version)[168]

From: http://www.teddi.ua.es/lineasTrabajo/verFoto.asp?pFoto=images/logoblocks.gif

[167] The Programmable Brick project led to the development of the LEGO company's RCX Mindstorms brick (http://www.lego.com/eng/education/mindstorms/home.asp?pagename=rcx&bhcp=1), and to a more recent MIT project called Cricket (http://llk.media.mit.edu/projects/cricket/about/index.shtml).

[168] "inicio" is "start"; "siempre" is "always"; "si entonces sino" is "if then else".

RCX Code / Lego Mindstorms (1998)

(LEGO, n.d.-1)

This is a language based on LogoBlocks (p. 161). It is supplied with Lego's Mindstorms construction kits, which incorporate RCX bricks (mentioned in the LogoBlocks entry in this section and presented in Figure 102).

"(…) *an RCX Code program consists of a collection of one-dimensional instruction 'stacks' made up of brick-like icons. Each 'brick' is a complete program statement* (…)" (McNerney, 2000, p. 25).

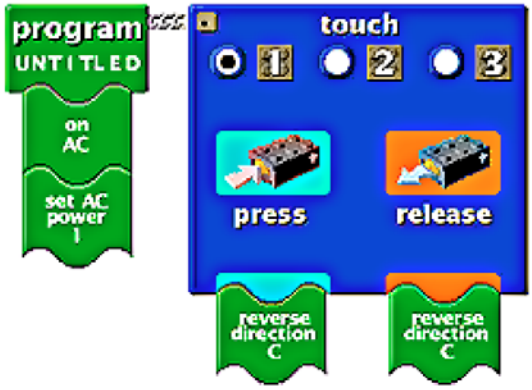


Figure 104 – Lego Mindstorms RCX Code

From: McNerney, 2000, p. 26

RoboLab (1998)

(Erwin *et al.*, 1999; Cejka, 2004)

RoboLab is an alternative language to program LEGO's RCX bricks. It is based on the LabVIEW visual language, with the goal of being "*an educational software for programmable robotics and data acquisition applications*" (Cejka, 2004).

RoboLab's basic principle is that all programming can be done by constructing a diagram, nothing else. That is, "*the user builds up a continuous thread of commands that defines the program*" (Erwin *et al.*, 1999, p. 7).

In order for the language to be usable and useful from preschool levels to college levels, RoboLab has different programming levels, with increasing complexity (*id.*, *ibid.*), as shown in Figure 105 and Figure 106.

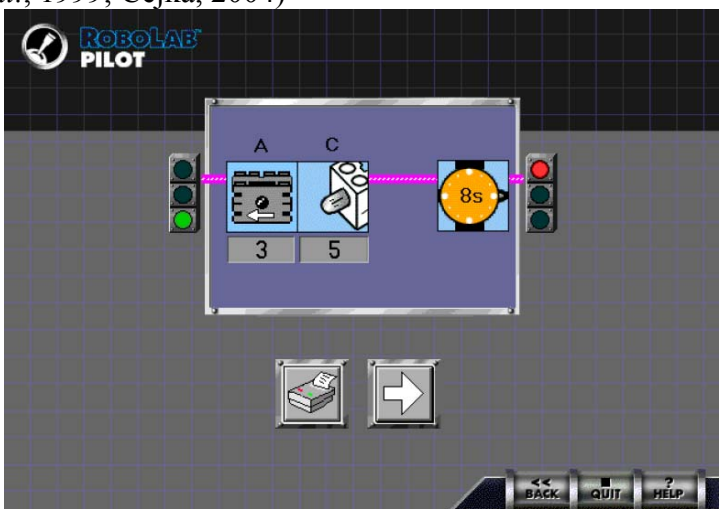


Figure 105 – RoboLab: "Pilot" programs allow users to select options from pull down menus; in this case it turns a motor and a light on for 8 seconds.

From: Portsmore et al., 2001, p. 3

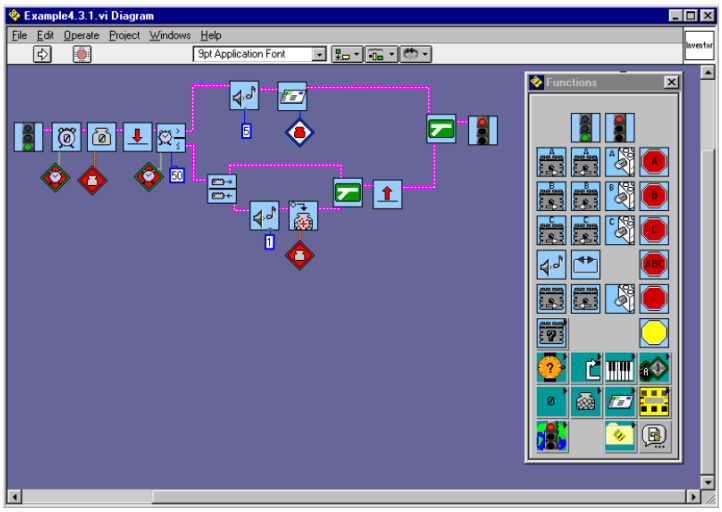


Figure 106 – RoboLab: "Investigator" programs, created by picking and wiring icons from a palette; in this case, an RCX records in a variable the number of presses on a touch sensor, and then sends the resulting value to another RCX

From: Portsmore et al., 2001, pp. 4

Magic Forest / Floresta Mágica (2000)

(Correia & Andrade, 2001)

Floresta Mágica is a programming environment and language developed at the Portuguese firm Cnotinfor, for the European project Playground[169]. The main developers were Tiago Correia and Dave Pratt[170] It is better known internationally by its English-language version, Magic Forest, commercialized by the UK firm Logotron (Figure 107).

Figure 107 – Floresta Mágica programming and play environment

From: Correia & Andrade, 2001

Its main aim is allowing children to change the programming of the games that are supplied with the product, or produce entirely new games.

Children program by combining icons in the form of "stones" inside "scrolls/papyrus" (Figure 109). A scroll belongs to one object, and the stones therefore affect the behavior of that object. There are blue, brown and beige stones, representing control, conditions and actions, respectively (Figure 108).

Figure 108 – Floresta Mágica control stone "I am controlled by the mouse", and condition and action stones "when the counter is less than 1, I play a sound, I make all objects explode, and I stop the game"

From: Correia & Andrade, 2001

The environment is the same for programming and playing: a control lever in the center of a "dashboard" bar, at the bottom of the screen (visible in Figure 107), allows the user to turn the programs "on" (to play) and "off" (to program).

Floresta Mágica also includes a method of communication between objects, which could be described as "broadcasting" messages. One of the action stones lets an object send a colored "envelope" without specifying the destination. All objects receive that message, and each can define its own processing rules for it, employing a condition stone with an envelope of the same color. When no valid rule exists, the received message is simply ignored.

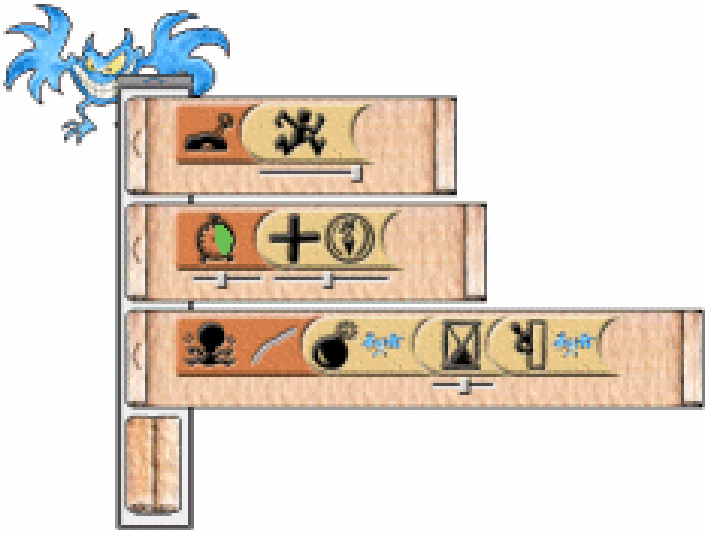


Figure 109 – Floresta Mágica: scrolls belonging to an object

From: Correia & Andrade, 2001

[169] http://www.ioe.ac.uk/playground

[170] Several other people collaborated in management, design, and technical advice, including Secundino Correia, Manuela Andrade, Peter Tomcsány, Ivan Kalas, and Andrej Blaho (Correia & Andrade, 2001).

Scratch (2004)

(Feinberg, 2004; Maloney et al., 2004)

Scratch is an adaptation of the Squeak programming language, presented earlier on pp. 146-151. It is presented as a toolkit aimed at simplifying the introduction to the use of programming.

The programming style of Scratch blends Squeak's drag-and-drop style of linking attributes (*vd.* Figure 82, on p. 147) with a graphical combination of statements evocative of LogoBlocks (p. 161).

Scratch includes a method of communication between processes that is similar to the one in Floresta Mágica (*vd.* p. 163). A process can "broadcast" colored messages, letting other processes decide whether they want to acknowledge messages of a given color or not, instead of explicitely specifying which processes should receive each message (Figure 111).

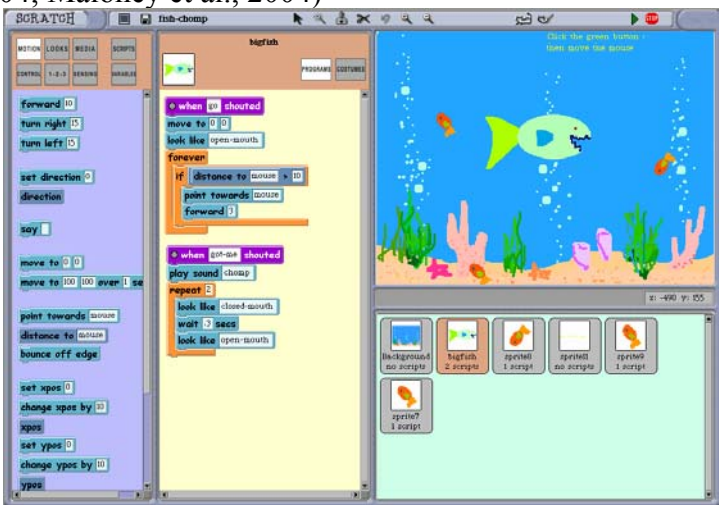


Figure 110 – Scratch environment and programming

From: http://llk.media.mit.edu/projects/scratch/pacfish.gif

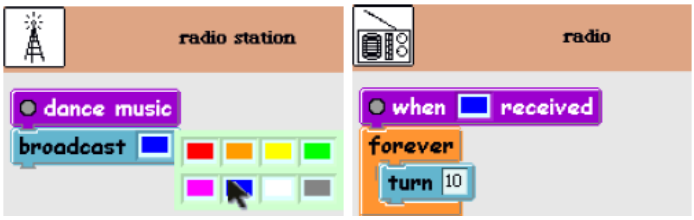


Figure 111 – Selecting a colored event to broadcast and responding to a blue event that was broadcasted

From: Feinberg, 2004, p. 27

PROGRAMMING INSPIRED BY LOGO'S TURTLE TALK

These systems focus on providing settings for using Logo's commands for controlling the screen turtle (forward, back, left, right, pen up, pen down, as explained in p. 144), even if they usually provide a few extra possibilities and features. In general, they provide a small taste of programming, or serve as entry points to the main concepts, by allowing children to focus on this small set of programming commands.

FASTR (1974) and TEACH (1975)

(Goldenberg, 1974; Solomon & Papert, 1975)

These systems provided "one-finger" versions of Logo's TurtleTalk commands, composed by single alpha-numeric commands. FASTR, written in Logo, "*scanned for characters typed to the keyboard. When certain ones were typed -- F, B, R, L, and a small number of others -- the turtle responded immediately, with default distances and angles built in. The program also kept a *record* of these moves, so that the young programmer could undo a move* (…)*, replay the set of moves* (…)*, and even name a procedure (after having executed all the moves)*" (Goldenberg, 2005). In the quite similar TEACH, "*numbers are used to set default values*" and "*the system remembers the steps of a procedure*" (Maulsby & Turransky, 1993).

Thinkin' Things Collection 3 – Half Time (1995)

(Edmark Corporation, 1995, according to Kelleher & Pausch, 2003)

"Thinkin' Things Collection 3" is an educational software product with different activities, one of which is "Half Time". Its announced aim is "*building programming skills*" (Riverdeep, n.d.).

The child must control the motion of up to 30 characters in a half-time show using commands identical to Logo's TurtleTalk, as can be seen in Figure 112, below the football field. There is also one control statement, the counted loop.

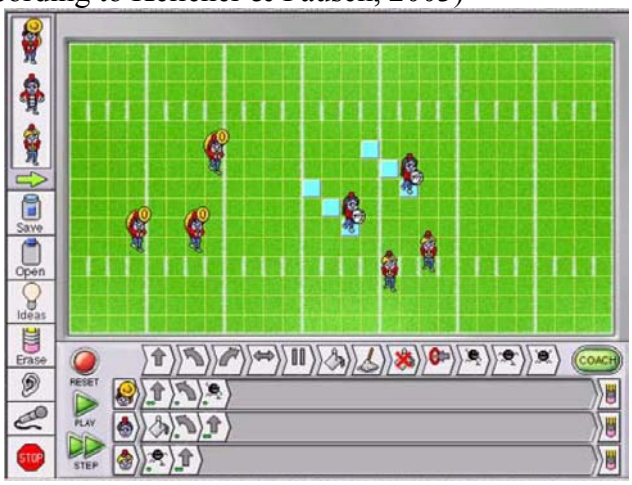


Figure 112 – Thinkin' Things Collection 3 – Half Time

From: Kelleher & Pausch, 2003, p. 15

Leogo and Cleogo (1997)

(Cockburn & Bryant, 1997; *id.*, 1998)

Leogo is a version of Logo that provides three different programming paradigms for the user to express programs or watch the results: "*direct manipulation*" (programming by demonstration); "*iconic*" (clicking buttons and dragging sliders) and "*normal text-based language*" (Logo).

Any interface action is simultaneously displayed across all three paradigms, in this way providing more than one way for the child programmer to interpret his/her actions.

Cleogo is a collaborative programming version of Leogo.

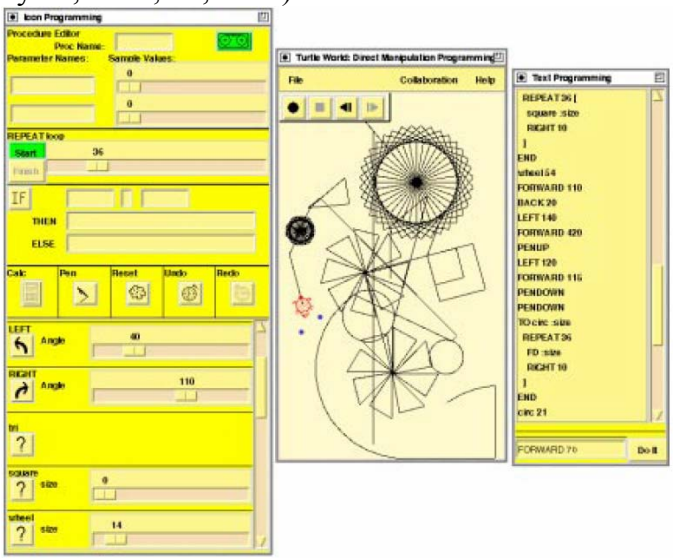


Figure 113 – Leogo/Cleogo environment

From: Cockburn & Bryant, 1998, p. 4

Drape – DRAwing Programming Environment (1999)

(Overmars, n. d.)

This system (Figure 114) aims to introduce children to programming by providing a graphical environment for exploration of commands similar to TurtleTalk. "*There are commands that draw lines and shapes, commands that move to a particular position, commands that set properties, like color, line width and size, and control commands, e.g. to repeat commands, to call procedures, and to test for mouse buttons*" (Overmars, n.d.).

Each picture represents a command, and a program is represented by a sequence of pictures. The example in Figure 115 sets the pen color to blue, draws a circle, moves to the right, draws another circle, moves to the right again, and draws a third circle.

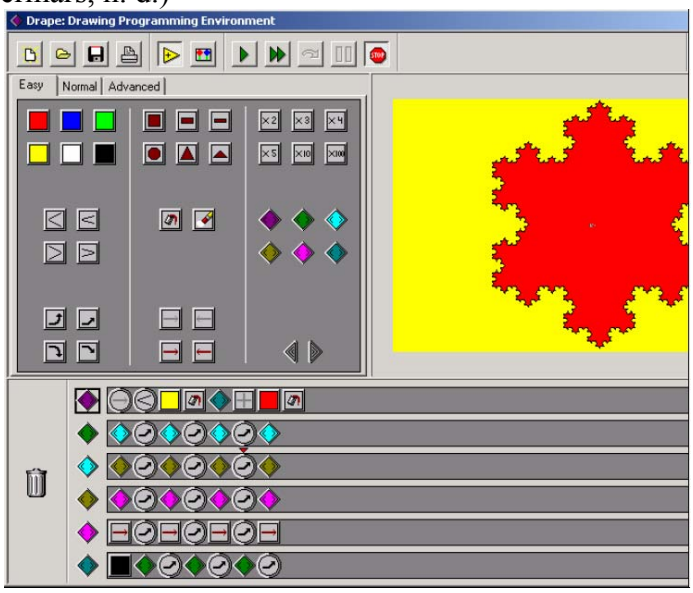


Figure 114 – DRAPE environment

From: Kelleher & Pausch, 2003, p. 17

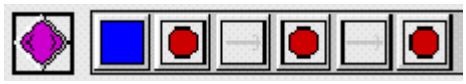


Figure 115 – DRAPE program

From: http://www.cs.uu.nl/people/markov/kids/drape/example2.gif

PROGRAMMING OF SOCIAL INTERACTIONS

These systems focus on the use programming techniques, such as event-driven activities and conditions, to control the interaction between characters: people, horses, dragons, etc. The range of uses for these systems can vary from a simple introduction of the power of expression achievable with programming techniques, to more advanced programming constructs such as loops and the planning and design of multi-object interactions.

My Make Believe Castle (1995) / My Make Believe Treasure Isle (1997)

(Bearden & Martin, 1998)

In this system[171], children can drag characters into several parts of a castle (entryway, bedroom, dungeon, etc.) and bring them to life using rules to specify actions and emotions such as jumping, dancing, getting angry, etc. The characters are among the traditional cast for castle-based fantasy stories: princess, dragon, jester, horse, etc. (Figure 116 presents the My Make Believe Castle setting and two characters).

Figure 116 – Two members of the cast of My Make Believe Castle

From:
http://www.microworlds.com/solutions/images/mmb_castle_screen.jpg

Even though text can be found, explaining a rule as it is programmed, it is only used for comments, and not necessary for programming. This is part of the features rendering it suitable for pre-literate children (the manufacturer recommends ages 4-7).

In Figure 117, the blue line shows the jester's path. Over it is a ballet slipper and a banana peel. The jester is programmed to dance when passing over the ballet slipper, and slip over the banana peel (TERC, 1999).

Figure 117 – A jester and its path

From:
http://www.terc.edu/mathequity/gw/gifs/ScreenShots/MycastleJester.gif

[171] I am referring to My Make Believe Castle. The "treasure isle" system is an identical one, only in the setting of an island, rather than a castle.

Concurrent Comics (2003)

(Kindborg, 2003)

This is a programming tool developed as part of Mikael Kindborg's PhD research. It employs a comic-strip metaphor, the overall aim being to use programming to "*represent the behaviour of concurrently acting characters, rather than actions in a linear story*" (Kindborg, 2003, p. 119).

The programming takes place using before-and-after rules, an approach similar to that used in Stagecast Creator (p. 151). Given the metaphor of comic-strips, Concurrent Comics employ voice balloons for textual output of social agents, frame labels for program information, "ghost images" to denote constant motion, and "moods" to denote changes in agent state.

The prototype tools developed by Mikael Kindborg employ two windows, one for programming (Figure 118 – a balloon pointer is visible on the left) and another for viewing the execution of programs (Figure 119). In these figures, a "medical doctor" is programmed to destroy "virus" it comes across. Live viruses could also have been rendered "virus corpses", for instance.

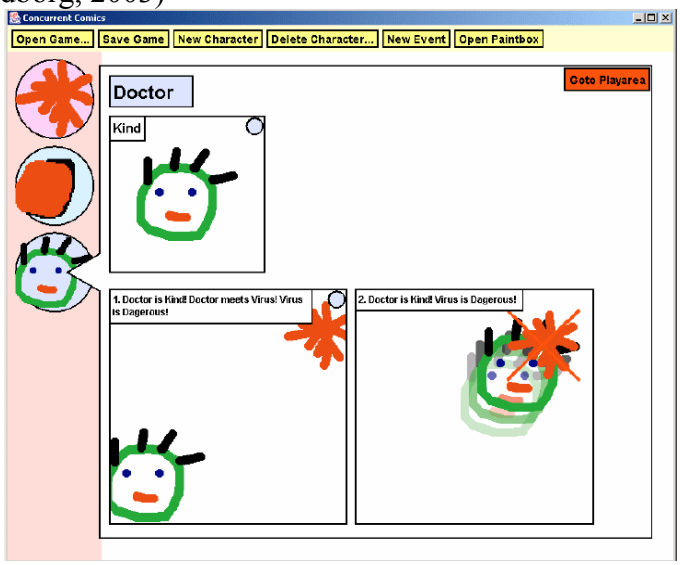


Figure 118 – Concurrent Comics: programming the virus destruction

From: Kindborg, 2003, p. 152

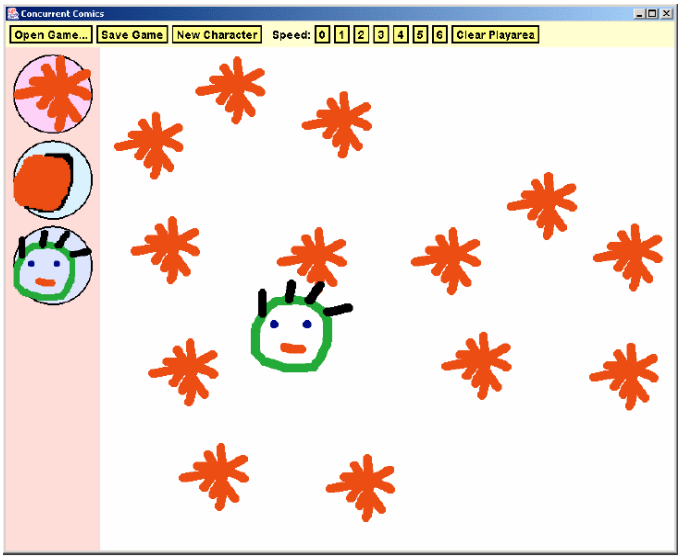


Figure 119 – Concurrent Comics – Playing the virus eater game

From: Kindborg, 2003, p. 151

Tableaux / MediaStage (2003)

(Immersive Education Limited, 2004)

MediaStage is a commercial educational tool for storyboarding (it was called Tableaux while it was only a research project[172]).

Its main idea is to "*use technology that is normally used to create typical, 3D viewpoint 'shoot'em-up' games for education* [, so that learners can] *create their own interactive visions*" (Burton-Wilcock et al., 2003). It employs modern techniques such as "*built-in real physics of objects, movement through 3D space, and the ability to animate virtual actors*" (*id.*).

Children can direct a theatre play or TV show, by defining script lines for each "actor", setting camera events, and so on. But more powerful interactions can be programmed: for instance, actors "*can be made to react to their surroundings*" (Cole, 2004), and facial and body expressions for feelings and emotions can be predefined, to express gestures and "*behavior (*[actors] *can be aggressive or confused, for example)*" (*id.*).

Figure 120 – MediaStage: placing props

From: http://www.kar2ouche.com/mediastage/images/grabs/grab03.jpg

Figure 121 – MediaStage: defining character's actions

From: http://www.kar2ouche.com/mediastage/images/grabs/grab01.jpg

[172] At the NESTA Futurelab, Bristol, UK (www.nestafuturelab.org).

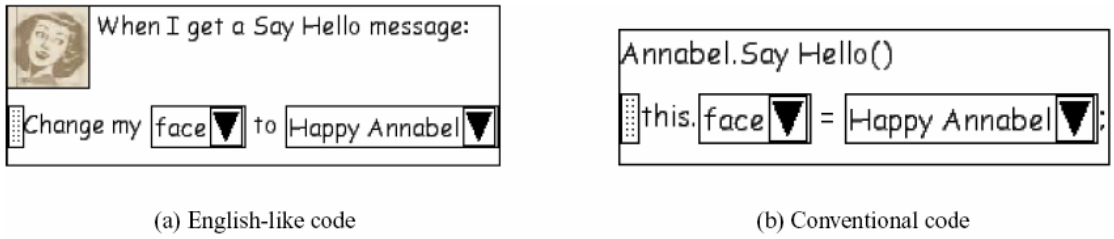
Mulspren – MUltiple Language Simulation PRogramming ENvironment (2004)
(Wright, 2004)

Mulspren is a language environment that aims to promote cooperative programming, *i.e.* having more than one person working on one program. It also provides two different notations, called "conventional-style" and "English-like", to cater for styles of programming of different users. Just as in Squeak Etoys, changes performed in one notation immediately impact the program representation in the other notation (Figure 123).

Figure 122 – Mulspren programming environment

From: Wright, 2004, p. 95

Users program with Mulspren by building 2D visual simulations, a common approach to programming for children, used in several other languages described in this dissertation. The programming area can contain several agents, which interact with each other; children program what is to happen when agents interact. According to Wright (2004, p. 96), "*children want to specify the direction and speed that an agent is moving in, where the agent is, and what it looks like. These requirements make the domain of programming visual simulations fit well with object–orientated event–driven programming paradigms*". Figure 122 shows rules for interaction and a list of mouse-related events (on the top-right pane).



(a) English-like code (b) Conventional code

Figure 123 – Programming notations in Mulspren

From: Wright, 2004, p. 102

PROGRAMMING WITH PHYSICAL OBJECTS

These programming languages and systems allow children to move physical objects to express programs, rather than type commands or manipulate pictures in a computer. In some cases, the final result of the programming must be watched on a computer screen, but in other instances, the results are available externally.

TORTIS (1976)

(Perlman, 1974; Perlman, 1976)

"*TORTIS* […] *is a system of special terminals together with software"* (Perlman, 1976), which aims to allow "*preschool children to communicate with and program the* [Logo] *turtle*" (Perlman, 1974, p. 2). It was "*designed so that only a few new concepts are introduced at a time but more can be added when the child becomes familiar with what he has*" (*id.*, *ibid.*).

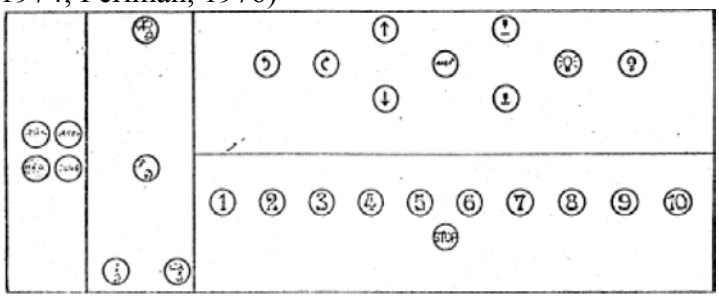


Figure 124 – TORTIS button box diagram (composed by 4 special-purpose boxes)

From: Perlman, 1976, p. 9

There are two kinds of terminals: Button Boxes had buttons for typical TurtleTalk commands (such as "forwards" and "right"), for parameters, and for control (buttons "start remembering", "do it", etc.); the Slot Machine is a set of several rectangular Plexiglas rows, each representing a procedure, "with slots in the top for the user to place cards" (*id.*, p. 4), each card representing a command.

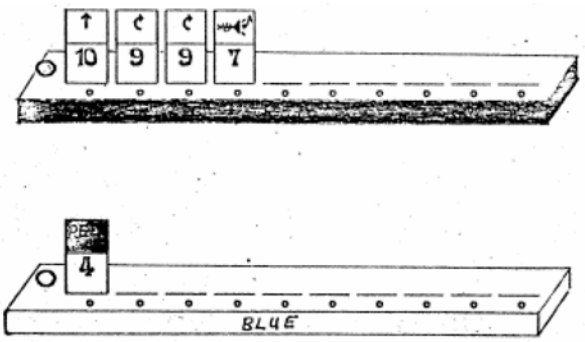


Figure 125 – TORTIS Slot Machine program for drawing a square and toot

From: Perlman, 1976, p. 19

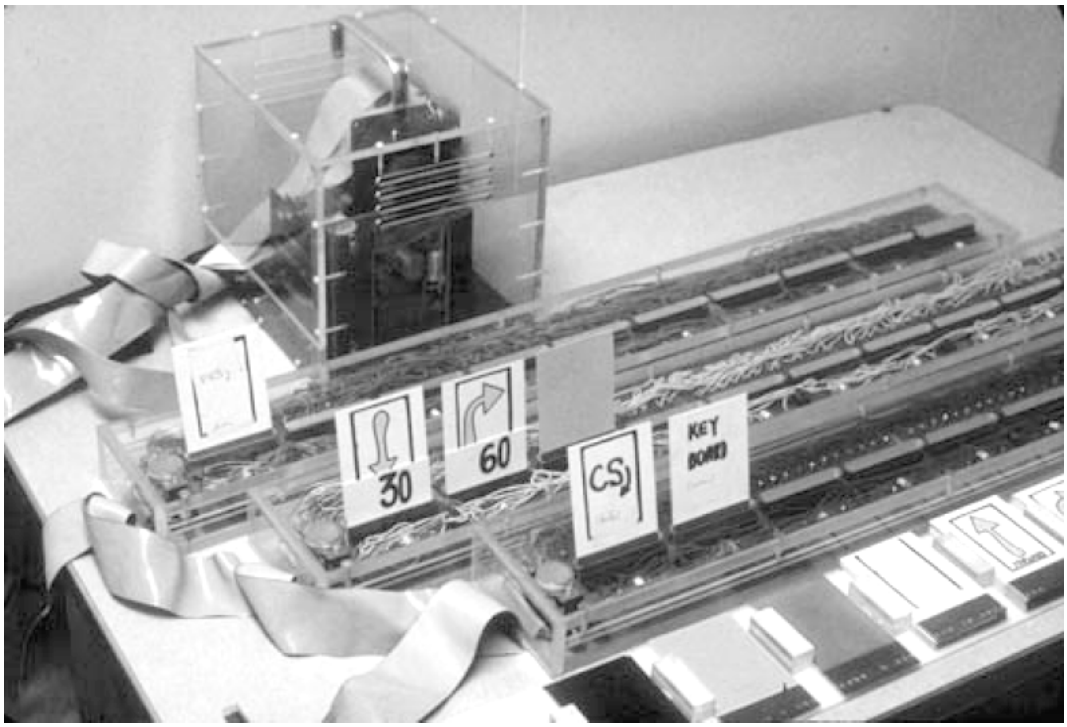


Figure 126 – TORTIS Slot Machine, by Radia Perlman

From: McNerney, 2003, p. 328

Valiant Roamer (1989)

(Valiant Technology, n.d.-1)

The Valiant Roamer is an autonomous and programmable robot (*i.e.*, it does not require a computer connection to be programmed or to execute a program). Children enter programs by clicking keys directly on the robot's "body" (Figure 127), within a control panel (Figure 128).

Commands are entered sequentially and the green "GO" key orders their execution. The available commands are mostly from Logo's TurtleTalk, but there is also a command key for playing musical notes and keys for definition of procedures and defining repetition of issued commands.

Sensors and actuators can also be linked to the robot, and replaceable "shells" can be customized, and several designs created, for better integration of the robot and programming activities to diverse educational settings (*e.g.*, João-Monteiro *et al.*, 2003, and Valiant Technology, n.d.-2).

Figure 127 – Children commanding a Roamer robot

From: http://www.utad.pt/~leonelm/papers/RobotinKindergarten/RobotinKindergarten_files/image026.jpg

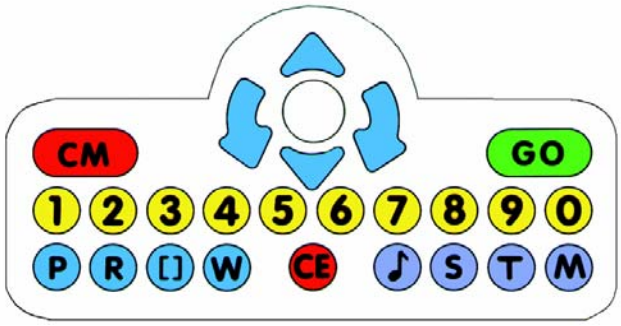


Figure 128 – Controls of the Valiant Roamer robot

From: http://www.valiant-technology.com/freebies/symbols/r-symbols.gif

AlgoBlock (1993)

(Suzuki & Kato, 1993; *id.*, 1995)

This system is composed by a set of physical blocks (Figure 129) that can be connected together, forming a program. This program can be linked to a computer, to watch its execution. Each block represents a Logo-like command in the AlgoBlock language. Children can follow a program's execution by noticing lamps lighting up on the blocks as their instructions are executed.

The most striking feature of the AlgoBlock system is its tangible nature[173] (the expression "tangible programming" is often attributed to the authors of this system, Suzuki & Kato). Two other novel features that arise as a consequence are: programs can be edited while they are executing (by removing and attaching blocks); and the execution can be traced by children, by noticing on which blocks lights turn on, indicating that the block's command is being executed.

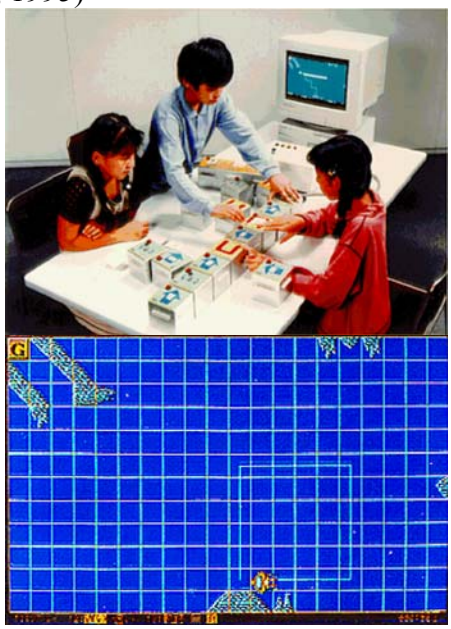


Figure 129 – AlgoBlock programming & screen

From: Kato et al., 1998, p. 144

[173] This was also a feature of the TORTIS slot machine (p. 171), 17 years earlier.

Logiblocs (1996)

(Logiblocs, n.d.)

These are building blocks with logical functions, sensors (inputs), and actuators (outputs), which can be connected to produce several constructions with varying degrees of complexity. "*From traffic lights to space stations, from cars to computers, everything in today's high-tech world is controlled in the same way as Logiblocs. Inside each Logibloc is a printed circuit board and every block is colour-coded to describe its function. By plugging together Logiblocs in different combinations you can create circuits and virtually write your own simple program to build your own new inventions*" (Logiblocs, n.d.).

Logiblocs are sold as special-purpose kits, but the blocks can be applied to a large number of projects involving design skills and problem-solving skills.

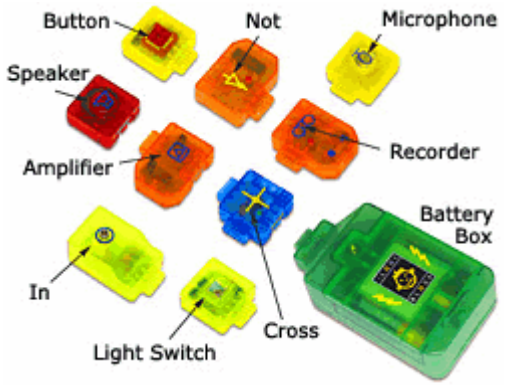


Figure 130 – Logiblocs ("Spytech" kit)

From: http://www.logiblocs.com/img/products/all.gif

AlgoCard (1997)

(Yamashita *et al.*, 1997; Yamashita, n.d.)

AlgoCard is a system based on AlgoBlock, which replaces blocks with cards.

"*AlgoBlock is a substantial programming language, but because each block consist*[s] *of electrical circuits, the system lacks flexibility* (…). *AlgoCard replaces blocks with cards. A computer recognizes cards and interpret*[s] *them as a program*" (Yamashita, n.d.).

There are two elements to the AlgoCard system: a software environment presents the position of the cards used to express the program and the corresponding program (Figure 131).

The cards themselves are tangible, placed on a table with a grid, and recognized by means of a computer-vision system (Figure 132).

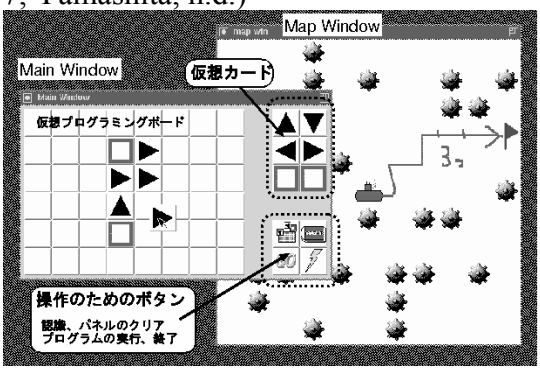


Figure 131 – AlgoCard software environment

From: http://jun.cyber.rcast.u-tokyo.ac.jp/RESEARCH/GRAD/WIN.GIF

Figure 132 – AlgoCard tangible environment

From: http://jun.cyber.rcast.u-tokyo.ac.jp/RESEARCH/GRAD/BOARD2.GIF

Tangible Programming with Trains (1998)

(Martin *et al.*, 1999)

This system was based around a train set, which included several active toys to influence the behavior of the train. The goal was for children to explore "*pre-programming concepts – causality, interaction, logic, and emergence*" (Martin *et al.*, 1999, as cited by K&P).

According to McNerney (2003, p. 331), the research and development by Martin *et al.* led to commercial products by several toy companies, such as LEGO's Intelli-train products.

"*Place the special Code Bricks along the track to make the train stop, change direction, or blow its horn. Want to change the train's speed? Put in a different engineer and watch it go!*" (LEGO, n.d.-2)

The age group suggested by this company for its train sets with programmable features is "3+".

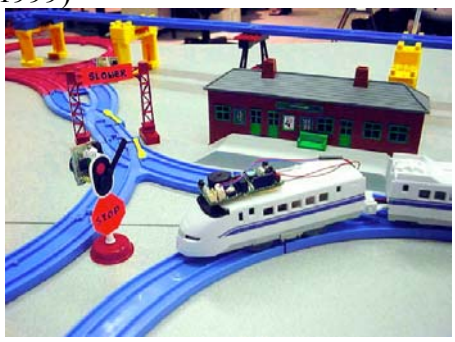


Figure 133 – Train set augmented with a Cricket programmable brick to heed infrared beacon signs

From: McNerney, 2000, p. 34

Figure 134 – LEGO's Intelligent Locomotive

From: http://cache.lego.com/images/shop/prod/10052-0000-XX-13-1.jpg

Curlybot (1999)

(Frei *et al.*, 2000)

"(…) *curlybot is an autonomous two-wheeled vehicle with embedded electronics that can record how it has been moved on any flat surface and then play back that motion accurately and repeatedly*" (Frei *et al.*, 2000, p. 129).

More than being a simple physical macro recorder, however, the curlybot can be used with sensors, to produce conditional behavior.

There is also the possibility of recording primitives, so that a child can record a circle, a box, or a line, and then use them as procedures. There is even the possibility of using a desktop computer to combine different "gestures" done with the curlybot into a program with a more traditional look, and then send that program to the curlybot for execution.

Figure 135 – curlybots

From: Frei et al., 2000, p. 129

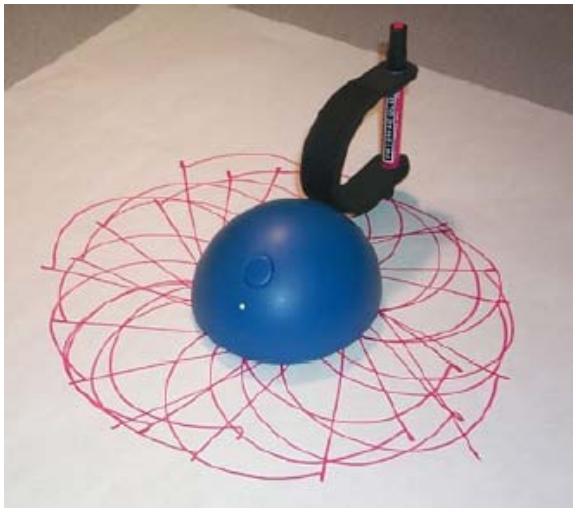


Figure 136 – curlybot with a pen attachment

From: Frei et al., 2000, p. 130

Electronic Blocks (2000)

(Wyeth & Purchase, 2000; Wyeth & Wyeth, 2001)

Electronic Blocks are "*blocks with electronic circuits inside them. By placing Electronic Blocks on top of one another young children build 'computer programs' – each stack of Electronic Blocks is capable of a different function*" (Wyeth & Wyeth, 2001). Programs performed by children aged between four and six include remote control cars and lights that flash responding to clapping.

Figure 137 – Electronic Block family (sensor, logic and action blocks)

From: Wyeth & Wyeth, 2001

Figure 137 show the existing electronic blocks: the three on the left are sensor blocks (back to front: *hearing*, *touch* and *seeing*); the four central ones are logic blocks (back to front: *delay*, *and* – the double block, *not*, *toggle*); the three right ones are action blocks (back to front: *light*, *movement* and *sound*).

Figure 138 – Remote control car with Electronic Blocks

From: Wyeth & Wyeth, 2001

Those are Lego Duplo Primo™ blocks with electronics inside, and can be stacked just as any ordinary Lego block. Figure 138 show a sample program: on the left, the touch sensor block is on top of the light action block, which therefore produces light whenever the sensor is touched. On the right, the movement action block has a seeing sensor block on top, which therefore moves whenever the sensor detects enough light. By placing the first set of blocks (remote control) near the second set (car), the car will move whenever the child put her/his hand in contact with the touch sensor. The logic blocks could also have been used. For instance, if the "not" block was placed between the seeing sensor and the movement block, the car would move whenever there was NOT enough light reaching the seeing sensor.

Tangible Programming Bricks (2000)

(McNerney, 2000)

"*Tangible Programming Bricks are physical building blocks for constructing simple programs*" (McNerney, 2000, p. 2). They were developed with the overall aim of allowing people without programming experience to control "*everyday objects, from toy cars to kitchen appliances*" (*id.*, *ibid.*).

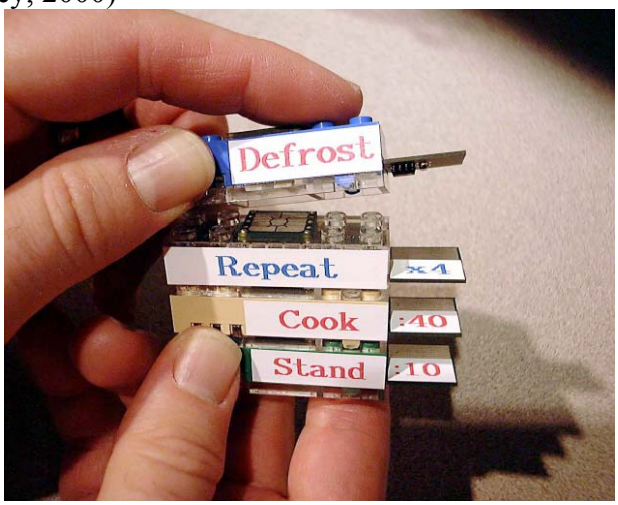


Figure 139 – Programmable Bricks program (the side-cards contain parameters)

From: McNerney, 2000, p. 15

Each brick has the electronics to behave in a specific manner, and can receive "parameter cards". For instance, Figure 139 show bricks to control the behavior of an oven, in a more detailed way than is normally possible: defrost (the number of minutes is not visible in the picture), then repeat 4 times: cook for forty minutes, and let it stand for 10 minutes.

AutoHAN/Media Cubes (2001)

(Blackwell & Hague, 2001)

This system aims to allow end-users to program appliances by using a language called Media Cubes. This language is based on associating specific cubes to specific appliances, and using them to specify intended behavior.

At their simplest, the cubes act as single-button remote controls, but they can be composed together to create programs (for instance, to start the coffee pot when the alarm clock goes off in the morning).

Figure 140 – AutoHAN cubes

From: Blackwell & Hague, 2001, p. 155

Tangible Programmable Blocks (2002)

(Mukherjee *et al.*, 2002)

This system aims to provide a mapping in the physical world of traditional algorithms from sequential programming.

The basic idea is that each line in a sequential program becomes a brick that can be connected to the following brick in sequence, and the resulting program result connected to actuators (*e.g.*, motors) with the proper connections. Programs with conditional control flow can be integrated in the model resorting to bricks with left, right, front and back connections, instead of just the top and bottom connections presented in Figure 141.

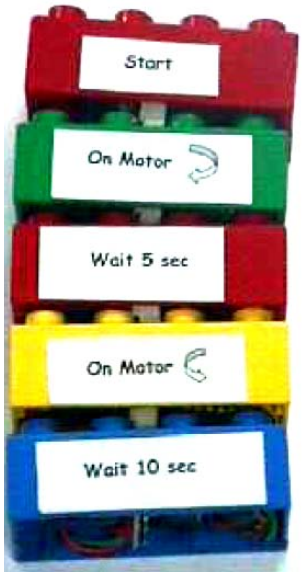


Figure 141 – Programmable Blocks implementation of Table 12

From: Mukherjee et al., 2002, p. 2

Start
↓
Run motor clockwise
↓
Wait 5 seconds for next task
↓
Run motor anticlockwise
↓
Wait 10 seconds for next task

Table 12 – Algorithm with motor behavior

From: Mukherjee et al., 2002, p. 2

Physical Programming (2002)

(Montemayor *et al.*, 2002; Montemayor *et al.*, 2004)

These authors envision physical rooms which interact with the people inside them, called StoryRooms. "*One might enter a room, press buttons, pull strings, or step on a rock, to hear music, watch a movie clip, or even feel the wind blow from a fan.*" (Montemayor *et al*. 2004).

Physical programming is a research project that aims to give children the power to participate in the creation of such interactive rooms, by using physical props. It is defined as "*the generation of physical interaction instructions by the physical manipulation of computationally augmented (or aware) objects in a ubiquitous computing environment*" (*id*.).

The physical props can be used, for instance, to define conditions such as "touched", and the actions to be performed when those conditions are met.

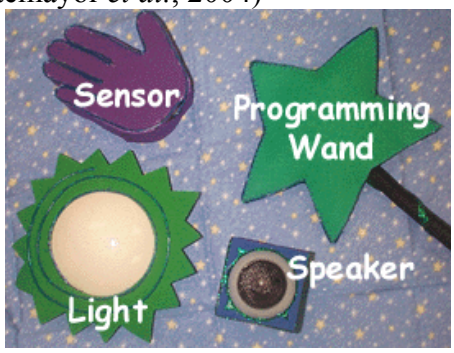


Figure 142 – Physical "icons" (props) for programming

From: http://www.cs.umd.edu/hcil/kiddesign/story_icons01_words.gif

Figure 143 – Using a Physical Programming wand

From: Montemayor et al., 2004, p. 18

System Blocks (2003)

(Zuckerman & Resnick, 2003)

This programming system was developed to allow children to explore concepts from system dynamics, such as feedback loops, time delays, levels, and rates. Children link together blocks with predefined behavior and in doing so produce systems, or programs, with distinct behaviors.

Available blocks include a **sender** (produces "1" when clicked), an **accumulator** (that updates its value by adding or subtracting its inputs), a **delay** (which holds its input for a number of seconds, before outputting it), a **converter** (which outputs "1" whenever it receives any value at its inputs), and a **MIDI-producing block** (that plays a MIDI musical note associated with its input value).

Figure 144 presents an arrangement of these blocks that includes a positive feedback loop, producing a linear growth in MIDI notes heard.

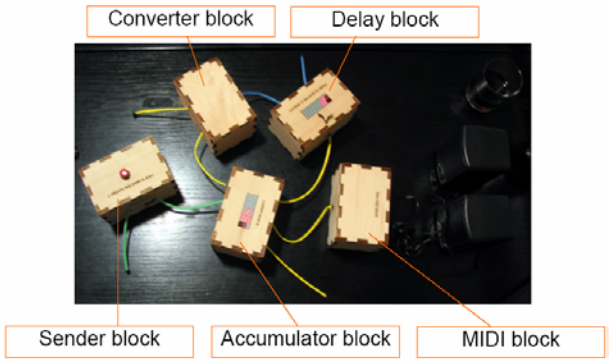


Figure 144 – System Blocks: simulation of positive feedback loop

From: Zuckerman & Resnick, 2003, p. 2

Figure 145 – Four-year-old girl playing with System Blocks

From: id., p. 1

Topobo (2004)

(Raffle *et al.*, 2004)

Topobo is a 3D constructive assembly system that can record and playback physical motion. A child can assemble several components, both static and motorized. The resulting constructions often resemble animals or skeletons (*vd*. Figure 146). These can be animated by "*pushing, pulling, and twisting them*" (Raffle *et al*., 2004), and the child can watch as the system plays back those motions.

The resulting behavior can be more complex than mere reproduction of the recorded motions, because one active element can be designated "queen" and linked to other active components, which then mimic whatever motions was "taught" to the "queen" element.

Figure 146 – Topobo: programming a horse to walk

From: Raffle et al., 2004, p. 648

Eco-Pods (2005)

(Kikin-Gil, 2005)

Eco-Pods are a programming system aimed at children aged between 4 and 6 years, which shares its mains goals with System Blocks (p. 177); namely, to "*help children learn systems thinking*" (Kikin-Gil, 2005, p. 1), defined as "*a holistic way of looking at the world (...) as an assemblage of interrelated components comprising a unified whole. The relationship of elements in the system facilitates the flow of data, matter or energy between them*" (*id.*, p. 4). The concepts whose learning the system aims to facilitate include "*feedback loop, interconnectedness and change over time*" (*id.*, p. 19).

In this system, children can indirectly influence the growth of a flower (Figure 147), by controlling the environment elements: wind, rain, light, heat. This control is done by acting (blowing, shaking, spinning, rubbing) upon tangible control elements, the Eco-Pods (Figure 148).

According to the researcher involved in the design and development of the Eco-Pods, "*children understood cause and effect, interconnectedness and change over time*" (*id.*, p. 50).

Figure 147 – Eco-Pods display interface

From: Kikin-Gil, 2005, p. 42

Figure 148 – Eco-Pods (water, light, heat, wind)

From: Kikin-Gil, 2005, p. 42

GRAPHICAL PROGRAMMING BY DEMONSTRATION

These systems employ techniques of programming by demonstration (a concept described in section 2.3.2). Programming is conducted by performing activities in a virtual world

AgentSheets / Visual AgenTalk (1993)

(Repenning & Perrone, 2001)

"*AgentSheets is an agent-based simulation-authoring tool that lets a wide range of end users (from children to professionals) create their own interactive simulations and games*" (AgentSheets, 2002, p. 5). The agents are objects programmable by end-users. They can "*react to mouse clicks and keyboard input, move around, change their appearance, play MIDI music and videos, speak, read Web pages, send email, and compute formulae*" (*id*., *ibid*.). The name of the programming language used for in AgentSheets is Visual AgenTalk (VAT).

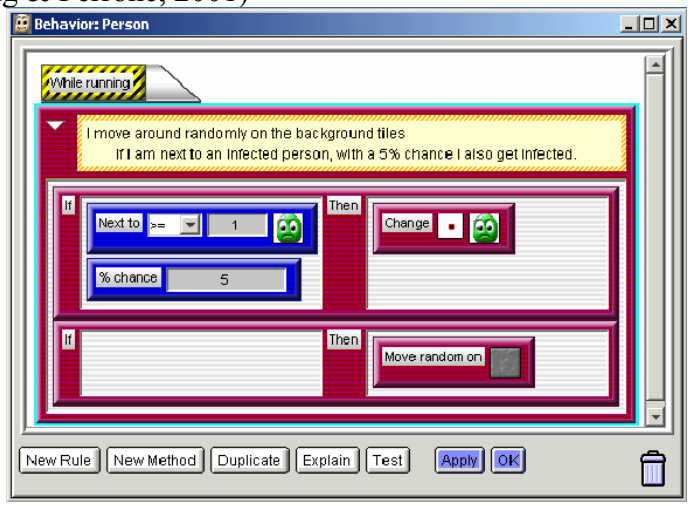


Figure 149 – AgentSheets rule definition

From: AgentSheets, 2002, p. 17.

VAT is similar to the method of rule-definition used in Stagecast Creator (*vd*. 3.3.4, on p. 151). However, its palettes of conditions and actions make more options available than in Stagecast Creator.

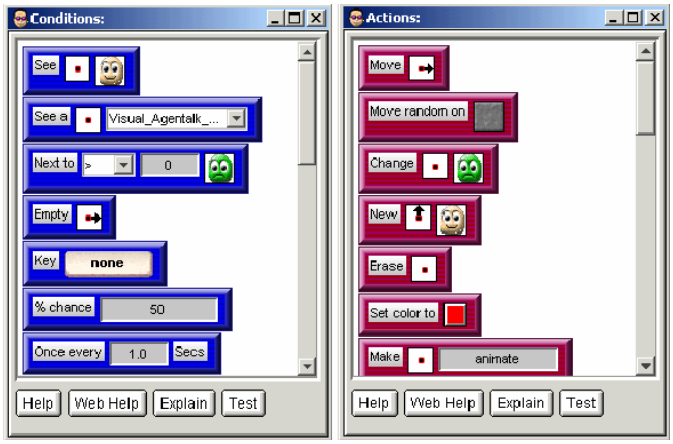


Figure 150 – AgentSheets conditions and actions palettes

From: AgentSheets, 2002, p. 15

ToonTalk (1995)

(Kahn, 1995)

ToonTalk is an animated programming language and a 2.5D[174] programming environment: the code itself is animated, like an animation cartoon. Figure 151 shows the programmer's hand holding a bird, while the help system (the Martian) provides information in a speech balloon.

This language was employed in the field work supporting this thesis, and is described in detail in section 3.3.5.

Figure 151 – ToonTalk environment

From: http://www.toontalk.com/inside.gif

[174] In ToonTalk, one can drop some objects on top of others, and also re-order the resulting stack, achieving different results. The programmer is therefore working with more complexity than just 2 plain dimensions, because the stacking order affects visible results. But it is not a fully 3D environment, because one cannot really produce 3-D objects, just stacks that remain two-dimensional stacks of 2-D objects (*i.e.*, the stacks themselves have no height, either visibly or interpreted from effects produced by sideway contact with other objects). This use of 2D objects whose design gives the viewer the illusion of working in 3D is commonly referred to as "2.5D".

Icicle (2002)

(Sheehan, 2004)

Icicle is a programming system inspired by Stagecast Creator (*vd.* section 3.3.4, p. 151), but without the restrictions of an underlying grid. In Icicle, "*objects, or performers, can be placed anywhere in an Icicle world with any orientation and size and they change smoothly from one shape to another by morphing*" (Sheehan, 2004, p. 147).

Icicle includes further technical improvements, such as a "*continuous model of parallelism*" (*id.*, *ibid.*), regarding the execution of rules (Figure 152 shows the rules for moving a train along a track). By allowing rules to be executed in parallel, rather than one at a time in sequence, several problems in game-programming are simplified.

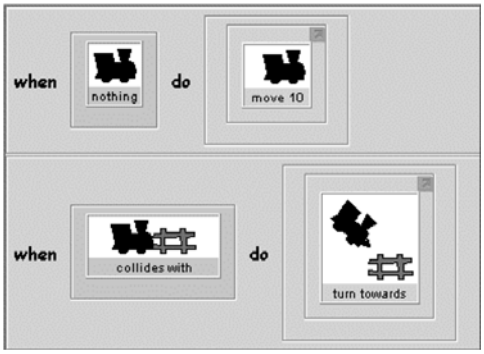


Figure 152 – Icicle programming environment

From: Sheehan, 2004, p. 147

PROGRAMMING IN A SOCIAL NETWORK CONTEXT

The main distinctive feature of these systems is their emphasis on peer programming. A child can use them to perform programming in an environment where that programming is visible to other children, who can be reached to request help, advice or comments in general. While several other systems include networking capabilities (*e.g.*, Squeak and ToonTalk), the systems included here have networking as their main feature.

MOOSE Crossing / MOOSE (1997)

(Bruckman, 1997)

This is a text-based virtual world (a.k.a. MUD[175]), which can be constructed by children together, "*making new places, objects, and creatures* (…) [such as] *baby penguins that respond differently to five kinds of food, fortune tellers who predict the future, and the place at the end of the rainbow—answer a riddle, and you get the pot of gold*" (Bruckman, 1997, p. 3). In order to do these things, children must use the system's language, called MOOSE.

MOOSE is a textual language that follows the path of simplifying the syntax, like other languages already described in this section. In the case of MOOSE, the starting point for the simplification was the set of already-existing languages for programming virtual worlds, such as MUSE and MOO.

MUSE

```
@va Rover=$PET ROVER:@pemit You pet
     Rover.; @emit Rover wags his
     tail.
```

MOO

```
@verb Rover:pet this none none
@program Rover:pet
player:tell("You pet Rover.");
this.location:announce_all("Rover
wags his tail.");
.
```

MOOSE

```
on pet this
     tell player "You pet Rover."
     emote "wags his tail."
end
```

Table 13 – Petting a dog called Rover, in three languages, including MOOSE

[175] Multi-User Dungeon. The name comes from the early use of the concept of text-based virtual worlds: a hosting place, where several people could play fantasy adventure games together. The original fantasy adventure games usually took place inside dungeons, hence the name.

Pet Park / YoYo (1998)

(DeBonte, 1998)

Pet Park builds on the ideas of MOOSE Crossing, by using a two-dimensional graphical world, instead of a textual one. The main goal is the creation of new "pets", with specific behaviors.

"*The vision of Pet Park is a kind of graphical MOOSE Crossing where kids can design and build things in a graphical virtual world* (…) *the focus of Pet Park is on building the virtual world by programming animated behaviors for creatures and other objects in the world.*"

(DeBonte, 1998)

While the environment is graphical, and therefore creating a new pet with new behaviors requires inclusion of graphics, the language used for programming is also textual, a script language called YoYo. The instructions can request for specific animations to be displayed or sounds played.

Figure 153 – PetPark environment and selection of behaviors

From: http://llk.media.mit.edu/papers/archive/deBonte-MEng/Figure5.gif
and
http://llk.media.mit.edu/papers/archive/deBonte-MEng/Figure7.gif

" (…) *using a property to store whether or not Grover is in a dancing mood,* (…) *Grover's dance script might look like this:*

```
if [wantToDance = "yes"]
[
      wagTail
      walk
      turnAround
      walk
      repeat 3 [jump]
      wagTail
]
[
      say "Sorry, I'm not in a
      dancing mood right now."
]
```

Building blocks might also include sounds along with the animation."

(DeBonte, 1998)

GAMES THAT INCLUDE PROGRAMMING ACTIVITIES

These systems were not necessarily designed as programming systems or programming environments. Rather they are games, some with educational intent, some entirely devoted to entertainment, where programming – or at least programming skills – is a central activity in gameplay[176].

RobotWar (1981)

(Warner, 1981; Home of the Underdogs, n.d.)

This was a game written by Silas Warner for MUSE Software in 1981, for Apple II computers.

Users must design and program robots to fight in an arena (Figure 154), using a language similar to BASIC to control several registers. These had names such as AIM, SHOT, and DAMAGE. The following code is an example from the manual (Warner, 1981, p. 36): a routine to determine if the robot has spotted another.

```
LOOK
  AIM + 5 TO AIM
  AIM TO RADAR
  IF RADAR<0 GOTO SHOOT[177]
  GOTO LOOK
```

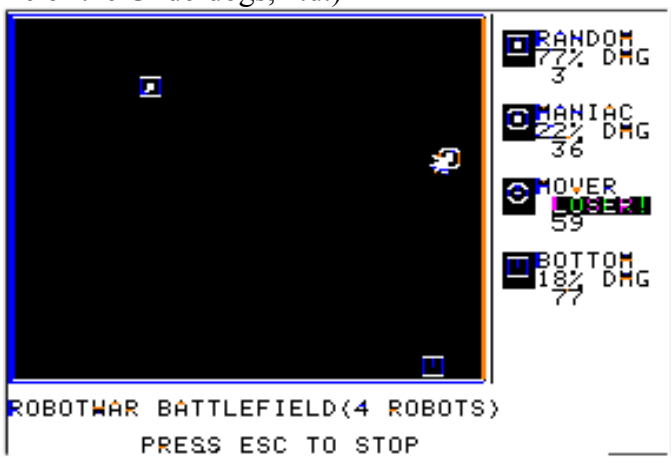


Figure 154 – RobotWar

From: http://robotwar.wikiverse.org/media/9/95/robotwar.png

Rocky's Boots (1982), Robot Odyssey (1984)

(Robinett, n.d.; Foote, n.d.)

These two games shared the idea that programming techniques from logic circuits had to be employed for the player to progress from level to level. They were co-authored by Leslie Grimm (the other authors were Mike Wallace, for Robot Odyssey, and Warren Robinett, for Rocky's Boots).

In Rocky's Boots, children could design "*simple digital logic circuits, using a joystick to move around circuit symbols on the screen and plug them together. The circuit components were AND gates, OR gates, NOT gates, and flip-flops*" (Robinnet, n.d.).

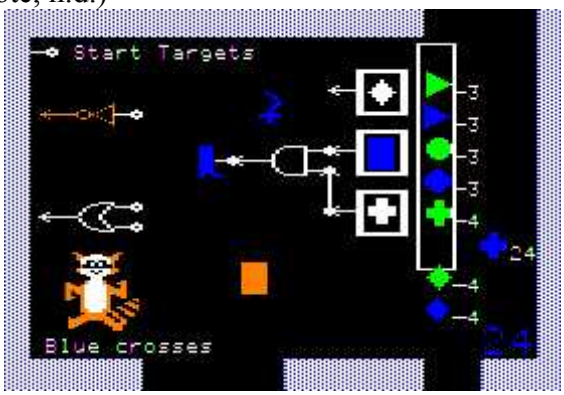
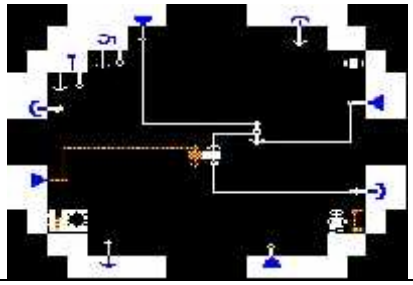


Figure 155 – Rocky's Boots

From: http://www.warrenrobinett.com/rockysboots/rocky_screen.jpg

Robot Odyssey built up from this idea: in each level, the player must program the behavior of one or several robots (in Figure 156, the there are two: a blue and a white). This programming was achieved by connecting together several circuit symbols and sensors inside the robots (Figure 157).



[176] Just like for game-editing software, this is a large genre. For a starting point reference, I suggest http://www.find-site.com/index/Games/Video_Games/Simulation/Programming_Games.html.

[177] This is the name of a user-written procedure for shooting. It is not a reference to the SHOT register, mentioned previously.

Figure 156 – Robot Odyssey level

http://mywebpages.comcast.net/tomfoote3/DQ/0b3c7890.jpg

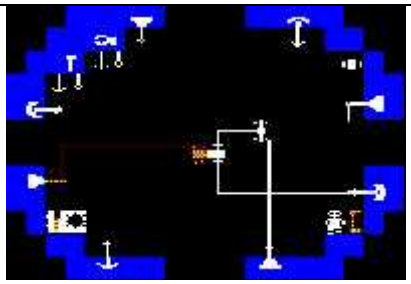


Figure 157 – Robot Odyssey

From:
http://mywebpages.comcast.net/tomfoote3/DQ/0b4c7890.jpg
http://mywebpages.comcast.net/tomfoote3/DQ/0b5c7890.jpg

Lemmings (1991)

(Meyer, 2004)

In this game, originally published by Psygnosis[178], lemmings are small creatures that plunge into each game level from a top trapdoor (Figure 158) and start walking until they fall or bump into a wall, turning around.

The player has no direct control over the path of each and every walking lemming, so his/her task is to assign specific tasks to a few of those lemmings, in this way programming a path for the remaining ones to follow.

Eventually, the lemmings will reach the level exit in enough numbers. The available tasks include digging, building ladders, exploding, blocking a passage, parachuting, etc.

Figure 158 – Lemmings level in mid-play

From:
http://www.mobygames.com/images/shots/original/1055171936-00.gif

In the level of Figure 158, for example, the blocking task must be assigned to lemmings placed at specific spots in the platforms, so that the remaining lemmings only fall small heights. This prevents them from splatting on the lower floor, so that they can reach the exit in enough numbers. In the figure, the leftmost and rightmost lemmings have already been assigned the blocking task.

[178] Now part of Sony Entertainment.

The Incredible Machine (1993) / The Incredible Machine 2 (1994) / The Incredible Machine 3 (1995) / Return of the Incredible Machine: Contraptions (2000) / The Incredible Machine: Even More Contraptions (2001)

(Reyes, 2000)

These games were inspired by a kind of comic gag common in several animated features: one event causes another, and another, and another, in a hilarious sequence[179]. Another inspiration source derived from the absurdly-connected machines known as Rube Goldberg inventions[180].

In these games each level has a specific goal, and provides several objects that the user must place and combine, so that the resulting contraption produces the intended result.

For instance, in Figure 160, the goal is for all three guns to fire. For this, several objects have already been positioned. The bowling ball, upon rolling, will hit the punching glove (containing a spring). This will tilt a bucket tied to a rope. Upon falling, the bucket will pull the rope, which in turn will pull the trigger of the first gun. The shot will blow the balloon, and the iron weight tied to it will fall. The bucket that was falling will also hit one end of the seesaw, whose other end is tied to the trigger of a second gun. In this figure, the player is currently positioning the second gun, and still has to come up with a solution for firing the third gun.

Figure 159 – The Incredible Machine I & II cover art

From:
http://www.mobygames.com/images/covers/large/970790813-00.jpg
and
http://www.mobygames.com/images/covers/large/972067977-00.jpg

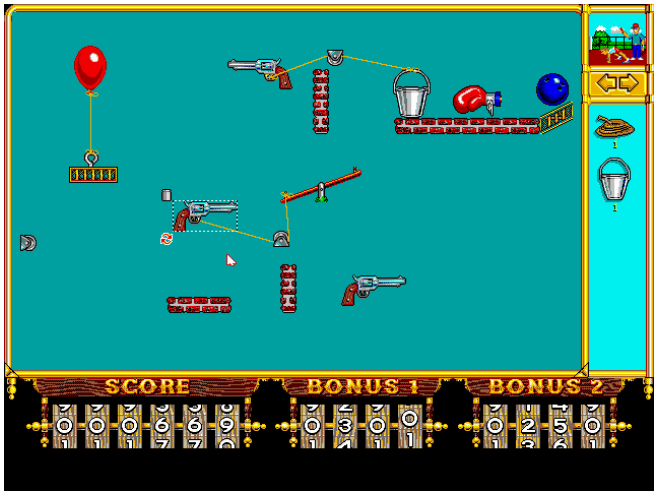


Figure 160 – The Incredible Machine screenshot

From:
http://www.mobygames.com/images/shots/original/1082679853-00.png

[179] Examples include Warner Brother's Road Runner & Coyote cartoons, Hanna-Barbera's Tom & Jerry cartoons, and more recently the 1988 movie "Who Framed Roger Rabbit", which opens with a tribute to this kind of gag.

[180] "*While most machines work to make difficult tasks simple, his inventions made simple tasks amazingly complex. Dozens of arms, wheels, gears, handles, cups, and rods were put in motion by balls, canary cages, pails, boots, bathtubs, paddles, and live animals for simple tasks like squeezing an orange for juice or closing a window in case it should start to rain before one gets home*" (Rube Goldberg Inc., n.d.).

Widget Workshop (1995)

(Cunningham, 1997)

This game combined ideas from the other games already presented in this section, particularly the series of "The Incredible Machine" games. But its goal was "*to turn science from a dull, boring, too hard subject into a fun, magical environment by looking at it from a mad scientist's point of view*" (Cunningham, 1997).

The game focused more on building contraptions using actual laboratory tools and equipment, rather than just cartoon-style gags.

Figure 161 – Widget Workshop

From: http://www.worldvillage.com/wv/school/images/scrnshot/widget4.gif

AlgoArena (1995)

(Kato & Ide, 1995)

"*AlgoArena is a simulation of Sumo, the traditional Japanese form of wrestling. Learners are supposed to program the actions of their own wrestlers with a LOGO-based programming language* (Table 14) *so as to defeat other wrestlers*" (Suzuki & Kato, 2002, p. 280).

A particular important feature of AlgoArena is that it was created to facilitate cooperative learning through programming: the player programs a wrestler to "*engage in bouts with opponents programmed by other learners or by the teacher*" (*id.*, *ibid.*). After programmed bouts have taken place, "*learners are supposed to analyze the results and incorporate solutions into their own programs*" (*id.*, *ibid.*).

Figure 162 – AlgoArena in game-play

From: Suzuki & Kato, 2002, p. 281

Figure 163 – AlgoArena programming environment

From: Suzuki & Kato, 2002, p. 282

```
REPEAT (30) [move_forward Defense]

TO Defense
IFELSE (:_his_hand = 2)[disturb_hishand][
IFELSE (:_my-posture <= 2) [bend_forward][
IFELSE (:_my-posture =4) [bend_back][
IFELSE (:_my-posture= <=2) [move_forward][Offense]
]
]
]
END

TO Offense
IFELSE (:_his_hand = 2) [disturb_hishand][
IFELSE (:his_posture =4) [slap_down][
IFELSE (:_my-hand =2) [throw][
IFELSE (:distance= 1) [grasp_mawashi][push_foward]
]
]
]
END
```

Table 14 – Sample AlgoArena program (English-language version)

From: Suzuki & Kato, 2002, p. 286

Mindrover (2002)
(CogniToy, n.d.)

MindRover is classified by its maker, CogniToy, as a "*3D strategy/programming game*" (CogniToy, n.d.). The player is a researcher on Europa, a moon of Jupiter. In order to play the game it is necessary to program mobile robots, called "rovers", to "*race around the hallways, battle it out with mini lasers and rocket launchers, and find their way through mazes*" (*id.*).

Programming takes place in graphical manner, by wiring rover components together and setting properties (Figure 165).

Figure 164 – MindRover gameplay

From: http://www.mindrover.com/images/battle2.jpg

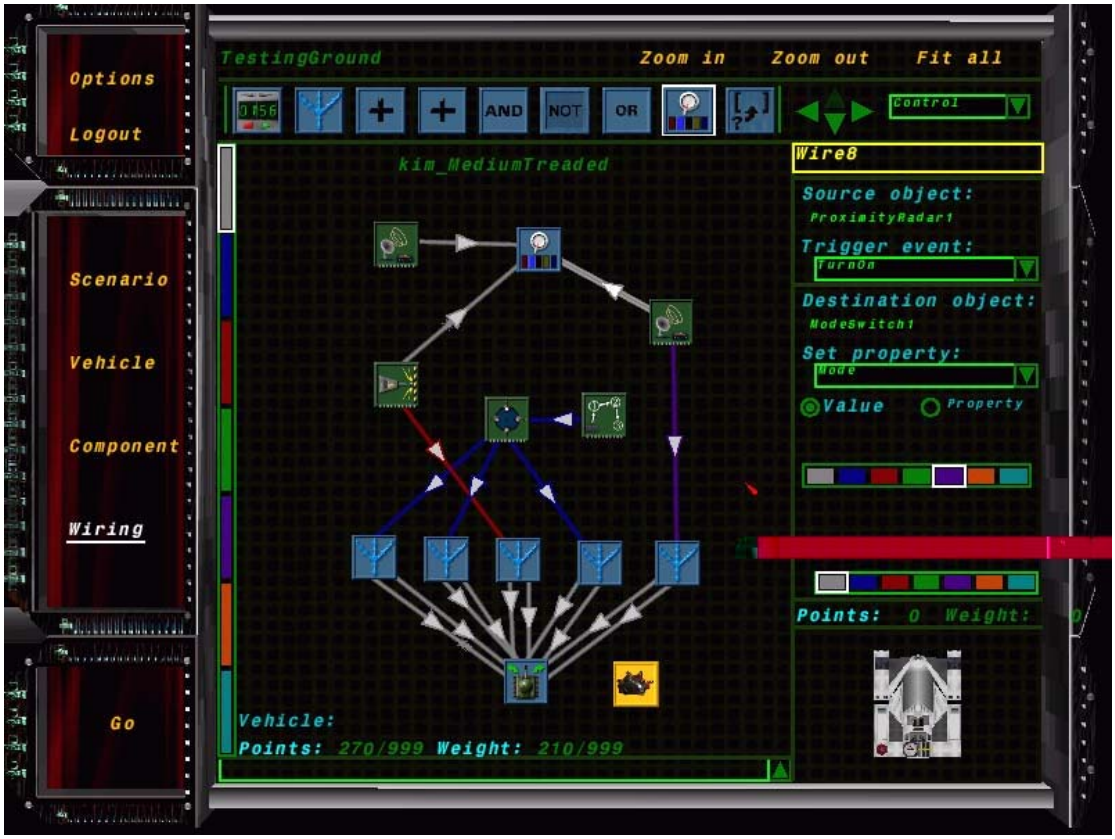


Figure 165 – MindRover programming environment

From: http://www.mindrover.com/images/console2.jpg

GAME EDITORS AND GAME-CREATION SYSTEMS

These programming systems are application-specific. They allow children of different ages to edit game-operation parameters, the results varying from producing games with similar operation but different context (edited graphs, maps, organization), to entirely new games. Since there are many such editors[181], here I present only a few examples of the genre from historical and modern periods.

Games Designer / Games Maker[182] (1983)

(Shaw, 1985)

Games Designer was a 1983 program for the ZX Spectrum, written by John Hollis, which allowed the users to edit parameters that determined the operational aspects of 8 included games, all of the "shoot'em up" variety (Figure 167). As Figure 166 shows, users could program the movement pattern for the "invaders", but also other elements such as the looks of the sprites, the sounds for the events or the sequence of "attack waves".

Figure 166 – Games Designer editing screens

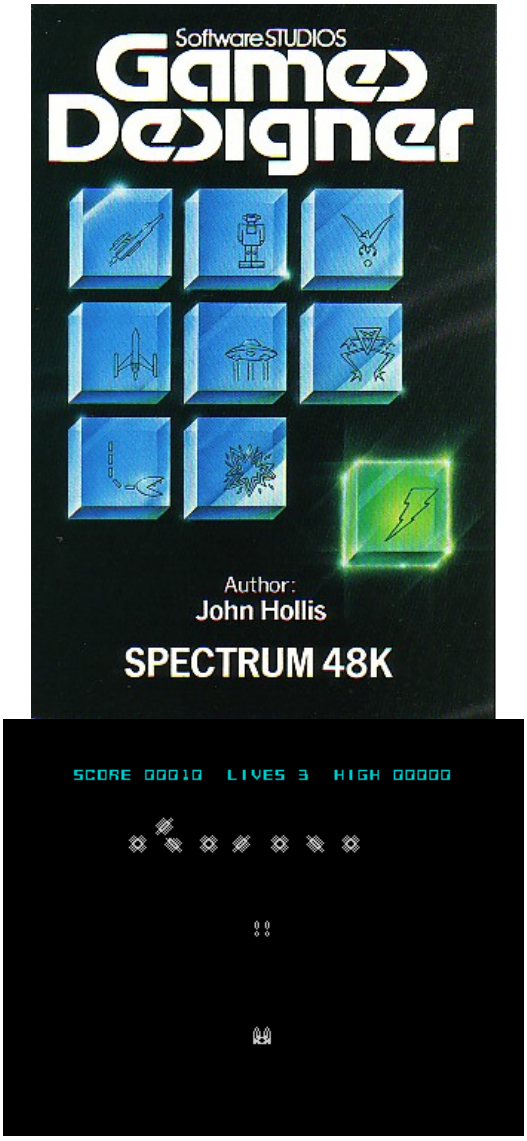


Figure 167 – Games Designer cassette inlay and sample game

From (cassette inlay): ftp://ftp.worldofspectrum.org/pub/sinclair/games-inlays/g/GamesDesigner.jpg

[181] For instance, browsing for game-design tools at http://www.the-underdogs.org/ provides no less than 214 different titles, which don't even include tools developed in Europe, such as Game Designer and HURG, presented in this section. Tools for creating entirely new games, however, are only 10.

[182] Released in the UK in 1983, under two different names: "Games Designer", edited by Quicksilva, and "Games Maker", edited by St. Michael. The only difference is the loading screen, otherwise the software is the same (after loading, "Games Maker" clearly displays the title "Games Designer").

Pinball Construction Set (1983)

(Cassidy, 2002)

The Pinball Construction Set was a 1983 program developed by Bill Budge for the Apple II and Atari 800 computers (it was launched in the same year as the Game Designer for the ZX Spectrum), which allowed users to create and play entirely new pinball games. It included a graphics editor and sets of parts for placement on the table, and the user could define the assignment of points and bonuses, and physics effects such as gravity and elasticity.

The Pinball Construction Set and Games Designer were pioneers in the genre of game editors.

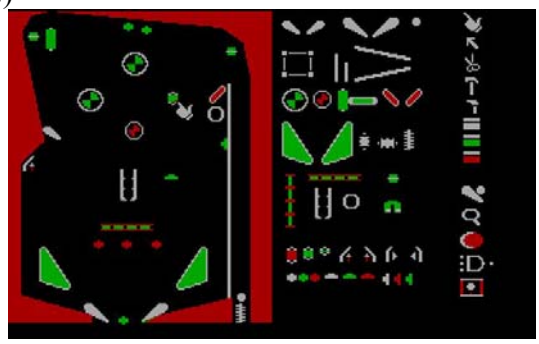


Figure 168 – Pinball Construction Set editing

From: http://archive.gamespy.com/halloffame/september02/pcs/4.jpg

HURG (1984)

(Shaw, 1985)

HURG (for "High-level User-friendly Real-time Games-designer"), was a 1984 program for the ZX Spectrum computer, written by William Tang, which allowed users to create level-based games of several genres, not just one (Games Designer only allowed for creation of "shoot'em up" games, Pinball Construction Set was only usable for creating pinball games).

Users could draw characters and screens (or load externally-drawn screens) and define game-playing parameters using just a joystick, through graphical schemas such as the one presented in Figure 172 (for defining the behavior of sprites upon collision: "no go", "eat", "crash", "go"). Rotating counters allowed number-entry with the joystick.

Events and conditions could be defined for updating of variables such as score or energy levels, control the behavior of elements such as "shots", specify events for the appearance or disappearance of objects, set the presence or absence of "gravity", and many other elements defining game-play.

HURG's variety of configuration and programming options allowed it to produce entirely new games, not just variations of existing ones. Much of its influence was lost with the demise of the ZX Spectrum computer.

Figure 169 – HURG – High-level User-friendly Real-time Games-designer

From: ftp://ftp.worldofspectrum.org/pub/□inclair/games-info/h/HURG.zip

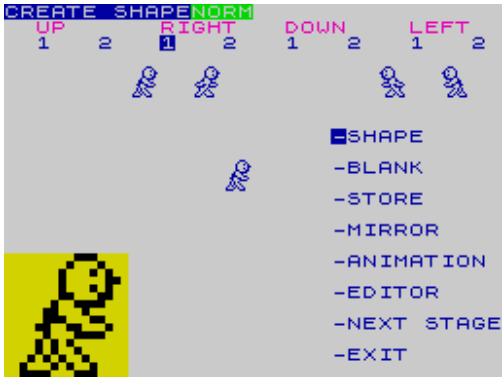


Figure 170 – HURG: defining sprite movement

From: http://www.users.globalnet.co.uk/~jg27paw4/yr17/yr17_24c.gif

Figure 171 – Screenshot from a game created in HURG by Tony Samuels for the "Your Spectrum" magazine

From: http://www.users.globalnet.co.uk/~jg27paw4/yr17/yr17_24.htm

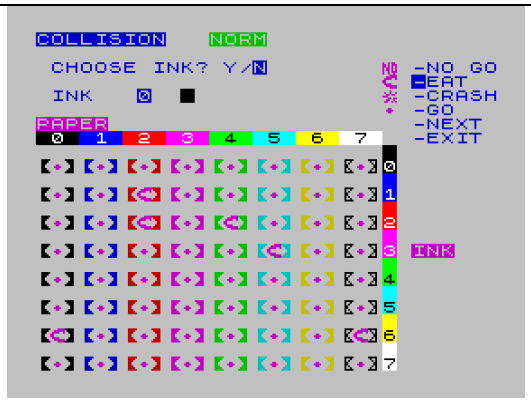


Figure 172 – HURG: defining behavior on object collisions

From: http://www.users.globalnet.co.uk/~jg27paw4/yr17/yr17_24d.gif

ARK: Alternate Reality Kit (1986)

(Smith, 1986)

The Alternate Reality Kit does not fit exactly within this category (game-creation systems), since its aim is to create interactive simulations, not just games (a goal it shares with Stagecast Creator, presented in p. 151). I have opted to include it here nonetheless, since this is more of a distinction in aim, rather than in actual programming features, which are similar to other products in this genre.

"*ARK is built upon a physical-world metaphor: all objects have an image, a position, a velocity, and can experience forces. Users manipulate objects with a mouse-operated "hand" which enables them to carry and throw objects, to press buttons, and to operate sliders*" (Smith, 1986, p. 61).

ARK included several innovative ideas. One was conceiving the metaphor of a virtual world, with a hand to grasp objects, and dragging command buttons to assign behaviors to pictures – these are techniques also used in ToonTalk (section 3.3.5, p. 195). Another was the idea of dragging parameters to buttons, which is similar to what is now used in Squeak Etoys (p. 146).

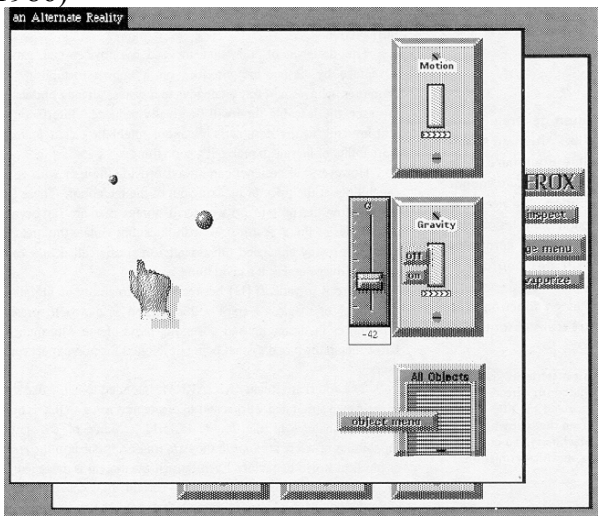


Figure 173 – Alternate Reality Kit environment

From: Smith, 1986. p. 62

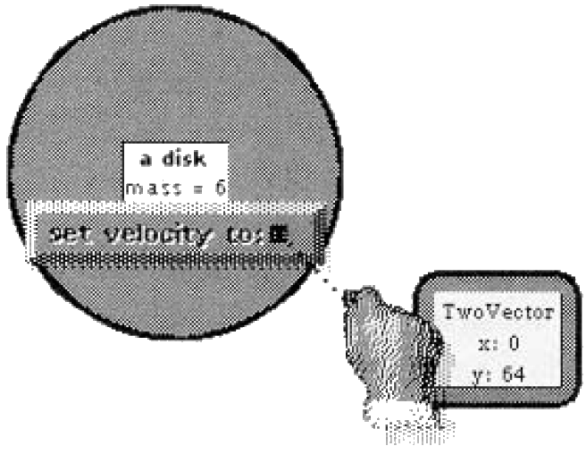


Figure 174 – ARK Dragging to assign a parameter to a button

From Smith, 1986. p. 63

Klik & Play (1994), Clik & Create (1996),
The Games Factory (1996), Multimedia Fusion (1998)
(Lionet & Lamoureux, 1996)

Klik & Play was the first "modern" generic game-making tool (*i.e.*, for Windows-based PCs). It was developed by François Lionet and Yves Lamoureux, and was published in 1994 by Europress Software Ltd, in the UK. These two developed then created more advanced versions of this tool, first at Corel Corporation (Clik & Create), then at their own company, Clickteam (The Games Factory and Multimedia Fusion[183]).

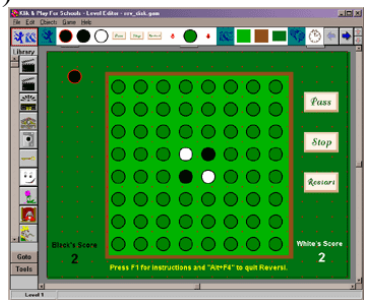


Figure 175 – Klik & Play level editor

From: http://www.gamelearning.pwp.blueyonder.co.uk/clubs/tools/images/KlikandPlay.gif

These systems allow the user to develop a game working in different editors and settings: a storyboard editor allows management of "levels" and their transitions; a level editor is available to edit each specific level of a game, defining graphics, animating them, and setting their properties; an event editor allows the definition of all events in the game, *e.g.* "*What happens when the alien collides with the spaceship? What happens when your hero eats the hot porridge?*" (Lionet & Lamoureux, 1996, p. 11), based on the state of variables and objects. There is also a tool for pre-debugging, to help the user detect undefined likely events, called "Step Through Editor".

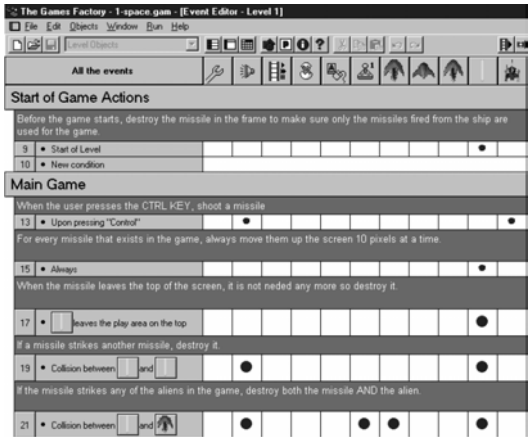


Figure 176 – The Games Factory event editor

From: (Lionet & Lamoureux, 1996, p. 23)

[183] Multimedia Fusion is aimed at adolescents and adults, rather than children.

Game Maker (1999)

(Overmars, 2004)

Game Maker is a modern game editor, developed by Mark Overmars. Like HURG or Klik & Play, it can be used to create entirely new games. Users can define sprites, objects, rooms, events and actions, employing features common in modern windowed environments, such as menus, tree structures for organization, tabbed dialogs, and drag-and-drop elements.

> *« You can import and create images, sprites (animated images) and sounds and use them. You easily define the objects in your game and indicate their behavior. And you can define appealing rooms with scrolling backgrounds in which the game take place. »*
>
> *(Overmars, 2004, p. 6).*

A particular feature of Game Maker is that it includes its own textual programming language, GML, similar to the professional programming language C. This feature is not child-oriented, but advanced users can have full control of the game-making interface: scripts in GML can be assigned to game elements, to finely define behaviors and events.

Game Maker has an interesting track record in terms of educational use, including technology summer camps and elementary schools, but also high schools and universities. The official site includes a page with teaching materials, an introduction for children aged 11-13, and other reference material, at http://www.gamemaker.nl/teachers.html.

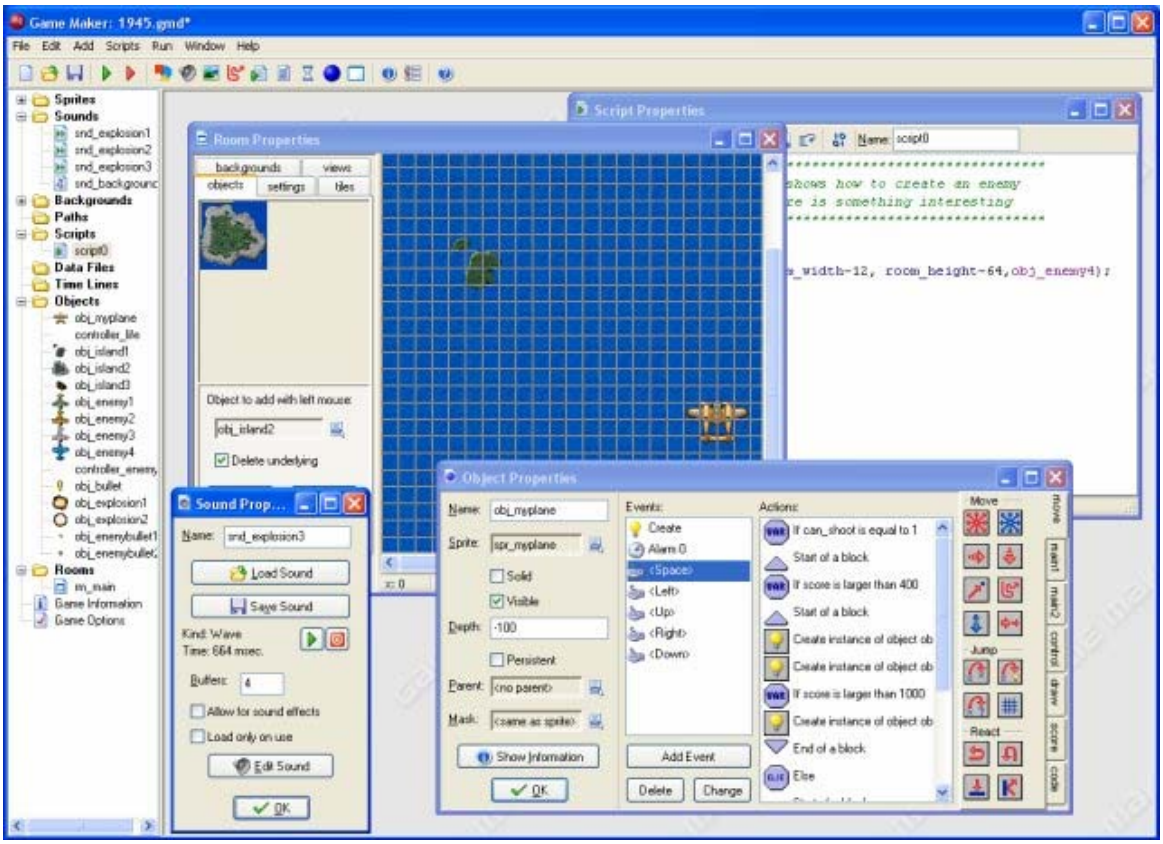


Figure 177 – Game Maker environment

From: http://www.gamemaker.nl/images/interface_large.jpg

Point & Click Development Kit (2004)

(Maas, 2004)

This programming environment is quite recent, and is presented here in representation of a long tradition of game editors devoted to the production of adventure games. This particular environment is devoted to the genre of 2D graphic adventure games (Figure 178).

Figure 178 – 2D adventure game

Users can define several positions and states for each object, and define events and actions using a simple scripting language. Following techniques from adventure creation, the player actions are anticipated and the programming involves producing responses to specific actions, such as "Use key with door".

Small action games are also creatable, using the interaction between graphical objects. For instance, Point & Click Development Kit is shipped with an action-game example: trying to hit as many rats as possible with a hammer (Figure 179).

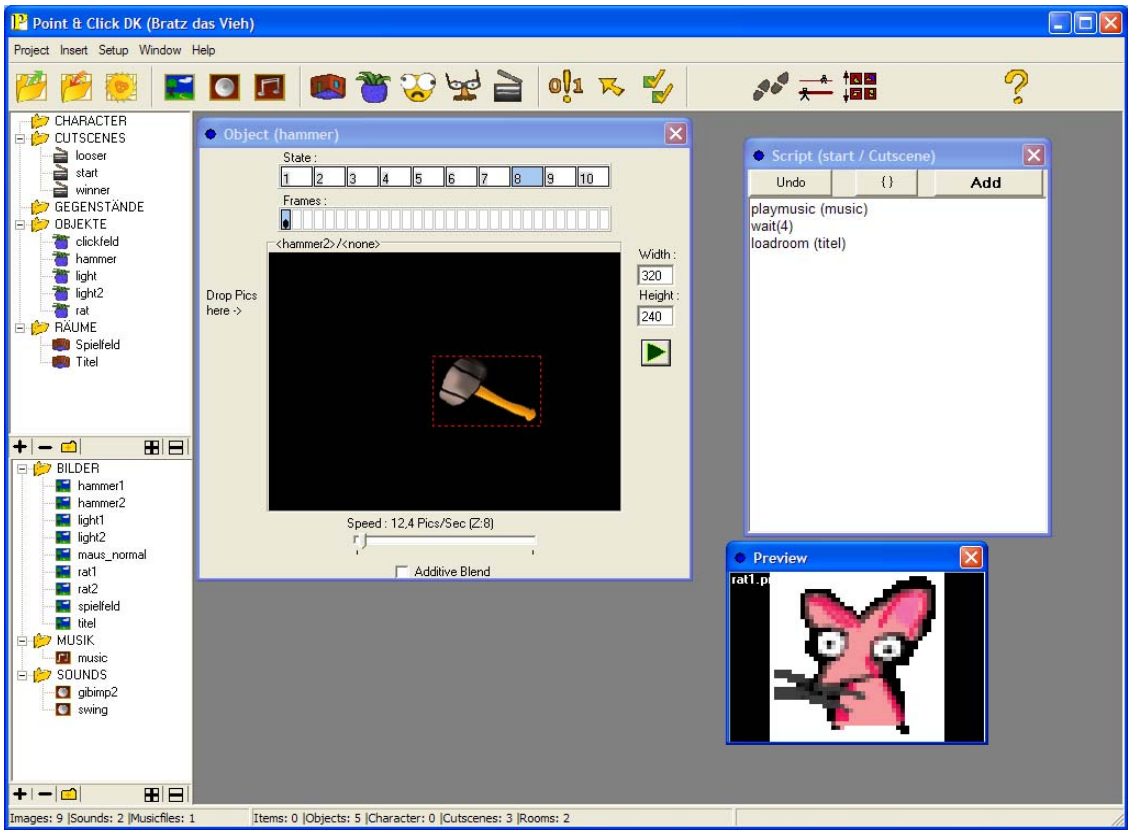


Figure 179 – Point & Click DK environment: object status, frames, image resource, action script

3.3.5. Animated programming and ToonTalk

As mentioned previously, the programming language ToonTalk was used to conduct the field work supporting this thesis (the reasoning for this choice is presented in section 5.1). This language embodies several concepts discussed in the previous sections of this chapter: concurrent programming (3.2.1), constraint programming (3.2.2), visual programming (3.3.2) and child-oriented features (3.3.3).

But ToonTalk (Kahn, 1995; 1996a) is also the language which implemented a novel idea in visual programming: **animated programming**. That is, using animation to express the code of a program, thus going one step beyond static visual programming. This is an original idea of Kenneth M. Kahn (Figure 180), who designed and developed ToonTalk as a proof-of-concept and commercial implementation of it. To the best of my knowledge and that of its creator[184], ToonTalk was the first language to employ this concept and is still the only known case (except for a few special cases in the physical world, as I'll mention shortly). Therefore, the presentation of the concept of animated programming, in the following paragraphs, is intertwined with the description and presentation of ToonTalk.

Figure 180 – Ken Kahn
From: http://www.digicon-inc.co.jp/img/ken.jpg

Being a novel concept, it helps to state what animated programming isn't: it is not the programming of animated characters or plays by means of static code; it is not the animation of the execution of static code; it is not the usage of animated icons or objects in place of traditional static icons.

Animated programming is the **usage of animation to express the code itself**. In order to help picture this concept, one can think of a programmer saying "watch what I do" and of a system that records and interprets the visual (animated) actions of the programmer. This is just a simplification, for complete expression of code must also include methods for other programming constructs, such as specifying conditions and generalizing the actions of the programmer.

> *« The fundamental idea behind ToonTalk is that source code is animated. (ToonTalk is so named because one is "talking" in (car)toons.) This does not mean that we take a visual programming language and replace some static icons by animated icons. It means that animation is the means of communicating to both humans and computers the entire meaning of a program. »*
>
> *(Kahn, 1995, p. 5)*

In this regard, one can think of cases of "animated programming" in the physical world: for instance, when a shooting a movie or rehearsing a play. Sometimes a director explains his/her intentions to an actor by performing actual body movements, gestures, facial expressions, etc. In this sense, there is "animated programming" of the actor's performance. By considering the physical world, a few of the systems described in section 3.3.4, under the category "Programming with physical objects" (p. 128), can also be considered instances of animated programming, at least partially. Specifically, the main feature of programming the Curlybot (p. 174) is reproducing physical motion; and programming in Topobo (p. 178) is entirely done by moving the programmable pieces[185].

[184] "*It seems that no one has ever tried to do this. (And when we figured out how to, we applied for a patent of the invention.)*" (Kahn, 1995, p. 7).
[185] In these cases, the "execution" of the program by the actor or robot is also an animation – an interpretation of the animation provided by the director/programmer. A generic animated programming system needs to go beyond mere cases of "animation to program animation".

However, programming a human actor, a Curlybot or a Topobo object all fall short of being general-purpose animated programming. For instance, while demonstrating something to an actor, a director often needs to resort to spoken speech to explain details that cannot be easily conveyed by movements of the human body or facial expressions[186]; the Curlybot needs to resort to non-animated programming for constructs such as conditions and procedures; and Topobo programming is limited to the specification of motion for its programmable elements.

Furthermore, physical programming systems such as these are also limited in terms of resources (space occupied, availability of enough programmable elements, and cost of such elements). The specification or execution of a program may require 100 actors, or 100 curlybots, for instance – a complex implementation situation.

A virtual animated programming language, *i.e.* one based within a personal computer, not requiring physical props, can be limited only in the same ways as other programming languages (computability, processor performance, available memory and hard disk space, etc.).

ToonTalk is such a language, with the added benefit of being a powerful system, not just a "toy" system. It was based upon concurrent constraint programming languages, and programs written in languages such as "*Flat Guarded Horn Clauses, FCP, Parlog, Strand, and PCN can be straight forwardly constructed in ToonTalk*" (Kahn, 1995, p. 4). "*Conversely, ToonTalk programs* (…) *can be translated to textual equivalents in these languages*" (*id.*, *ibid.*).

> *"We chose concurrent constraint programming as the underlying foundation of ToonTalk* (...)*. One reason is that over ten years of use at many research centers has demonstrated that there is no risk that the language will be inadequate for building a wide variety of large programs. The languages are small yet very powerful."*
>
> *(Kahn, 1995, p. 4)*
>
> *"ToonTalk was built to be both very easy to learn and to be a very powerful and flexible programming tool.* (...) *Kids can pretend to help in the garden with toy shovels and rakes, but if they really want to do gardening they should have real tools that have been adapted to their special requirements."*
>
> *(id., p. 3)*

But why would someone wish to program with animation? Several usability benefits are expected from employing animation, the most important one being the expectations that animation can help the users/programmers better understand the transitions between visual elements in visual programming languages. Another expected benefit is that animation can help provide better metaphors, and consequently better mappings between programming concepts and those of users.

> *"*(...) *in order for a static picture to represent the dynamic behavior of a program, it needs to rely upon a rich set of encodings. Control and data flow need to be encoded in abstract diagrams. Abstract diagrams are arguably easier for people to deal with than symbolic formalisms, but they are still very difficult. Why not take the next step from visual programming languages and begin to use dynamic images, i.e., animation, to depict the dynamic processes of a program?"*
>
> *(Kahn, 1995, p. 5)*

[186] In Sweden, Jakob Tholander and Ylva Fernaeus have conducted programming activities with children, without a computer, in a setting similar to this "movie director" situation (Fernaeus & Tholander, 2003). Each children would be assigned (or define) a small program by a combination of body motion, spoken explanation and written notes. Just like in the "movie director" example, part of this programming is animated, but not its entirety.

Given the novelty of the field itself, there is no available research on usage of animated programming, by controlled comparison with static programming, to support or disprove these expectations, but informal anecdotal evidence, favorable to them, is mounting up (*e.g.*, Playground Project, 2001). However, research regarding the impact of animation on users' understanding of visual information, albeit limited, provides positive indications:

> *"If a task requires a subject to know something about objects' spatial position, and the viewpoint is changed, then animating that change in viewpoint appears to help users.* (...) *This has direct implications for many applications that present linear data – including word processors, spreadsheets, and Web browsers."*
>
> *(Bederson & Boltman[187], 1998)*

Before proceeding with the description of specific features of ToonTalk, I need to address two problems with the implementation and description of animated programs. The **creation of animation** by the user/programmer and **providing a static description** of the programs (for instances such as their presentation on printed matter).

The creation of animation is a basic problem for any animated programming language: since the source code itself is animated, the user must be able to produce the required animations. But producing animations is often a complicated task in itself[188]. The solution provided by ToonTalk is to make the programming environment resemble a video game, where the user controls and manipulate game elements while programming.

> *"(...) constructing animation is generally difficult and time-consuming. Good animation authoring tools help but it is still much more difficult to animate an action than to describe it symbolically. Luckily, there is one sort of computer animation that is trivial for a user to produce – video game animation. Even small children have no troubles producing a range of sophisticated animations when playing games like Mario Brothers. While the range is, of course, very limited relative to a general animation authoring tool, video game style animation is fine for the purposes of communicating programs to computers."*
>
> *(Kahn, 1995, p. 5)*

[187] This paper also contains a nice summary of studies on how animation affects user performance.
[188] To the point that there are programming languages specifically targeting the creation of animations, such as SAM (Geiger *et al.*, 1998) or Alice (Conway, 1997).

Even considering that the animation is that of a traditional video game, the other aforementioned problem remains: animated code, be it ToonTalk code, filmed gestures or some other situation, is troublesome to describe in a static medium such as this written document. Sometimes a simple screenshot will suffice, but not always.

Circumventions for this have been used, and throughout this document I will employ an approach already used by Ken Kahn (1995) and Mikael Kindborg (2003): capturing pieces of the action as still frames and arranging them in comic-book style.

Ken Kahn calls these pieces "snapshots" (*vd.* Figure 181), and Mikael Kindborg (*vd.* Figure 182) refers to a set of them as a "comic strip".

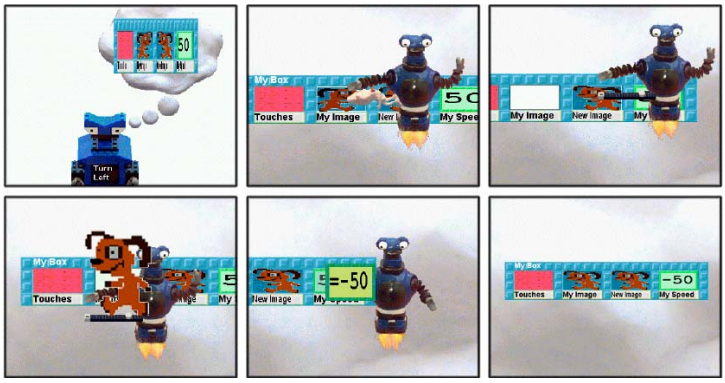


Figure 182 – Comic strip: ToonTalk program for making a puppy turn around

From: Kindborg, 2003, p. 107

In some circumstances, however, the lack of motion in these images may pose doubts to the reader. In those cases, my personal contribution to the problem is to add textual comments below each frame (a full example is available in Annex I, on p. 441), a technique almost as old as comics themselves (a classic example is presented in Figure 183). These comments will convey to the reader some of the information unavailable due to lack of motion.

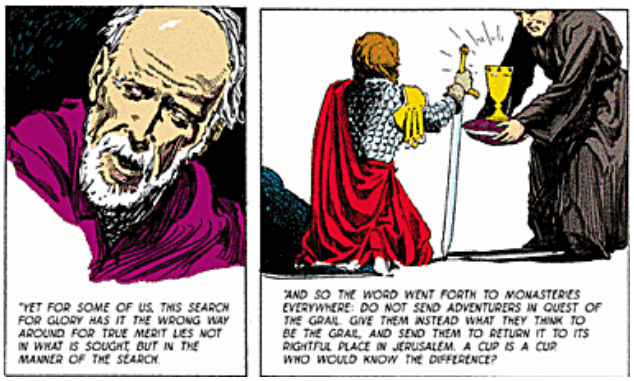


Figure 183 – Caption style used in Prince Valiant comics

From: http://est.rbma.com/content/Prince_Valiant?date=20041107

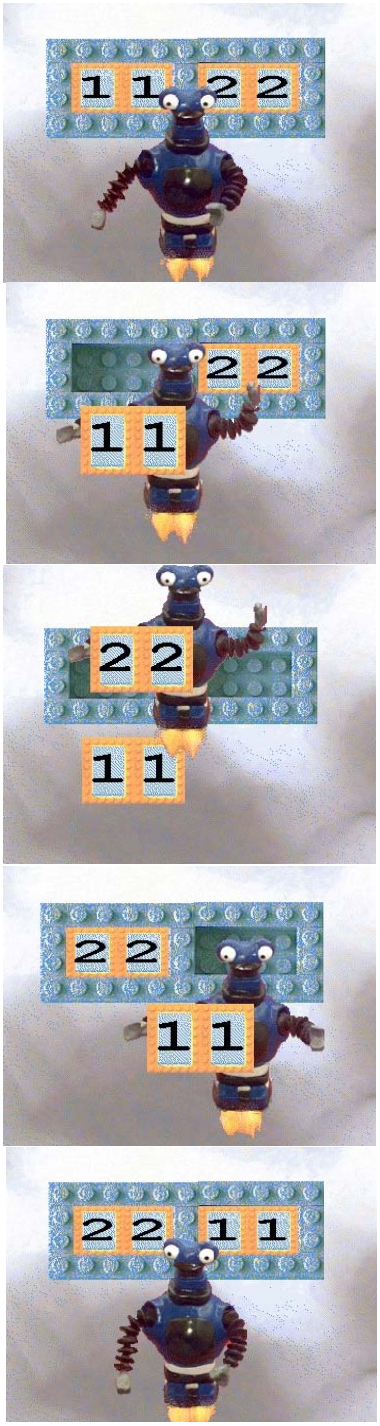


Figure 181 – Snapshots from swapping two elements

From: Khan, 1995, p. 6

Ensuing with the description of ToonTalk and animated programming, it is necessary to say that ToonTalk is not just a programming language, but also a programming environment where that language is used to create programs. For this reason, I'll employ the analysis categories already used in section 3.3.4 for other programming languages and environments: expressing a program, understanding the execution of the program, and child-orientation of the programming environment.

In ToonTalk, the most striking feature associated with the latter category is the emphasis on providing a global metaphor for conducting programming activities. As much as possible, all the programming activities take place in a video-game world resembling a city. The user (programmer) controls a "doll" or "figurine", seen as his/her programming "persona" or "avatar" (Figure 184).

Controlling that programmer doll, in video-game fashion, the user can enter and leave houses, where computations are taking place, observe and debug ongoing computations, and create new computations using objects representing tools and primitives, stored inside a legged, trailing, toolbox. Since the city can grow large, a helicopter is provided, to allow a bird's-eye view and simplify navigation in the city (the toolbox and the helicopter are visible in Figure 184).

Figure 184 – ToonTalk programming environment: programmer and toolbox outside

The programming takes place within this global metaphor. By clicking the mouse the programmer "kneels" on the floor of a house, to program or debug. This makes the toolbox open, displaying the programming primitives: numbers, letters, and objects to use for performing comparisons, process[189] spawning, process termination, etc. The metaphor is maintained because now the mouse movements still control the programmer's persona, visible as a hand and an arm (*i.e.*, not just a cursor shaped as a floating hand). Figure 185 presents this hand, the open toolbox on the floor, and several tools and primitives.

[189] The term "process" is used here just for the sake of simplicity in this short description. "Agent", "actor" or "object" could also have been used.

Figure 185 – ToonTalk programming environment: programmer's hand and toolbox contents

By "tools", I am referring to elements whose only purpose is to allow the programmer to manipulate the environment and the visual elements used in programming: producing copies, zooming in and out, deleting (vacuuming), erasing (creating a generic version of an element), saving to disk, etc. They are extensions to the programmer's persona. All other ToonTalk elements are primitives. In this taxonomy, the programmer's hand and the programmer's persona aren't classified at all (neither as tools nor as primitives); they are meant to be seen as extensions to the human programmer's own body and mind. One should have in mind that since tools are extensions to the persona, a different "programmer" might not need them to achieve the same purpose: for instance, if the programmer persona was a winged dragon, rather than a human, there would be no need for an helicopter to fly around, nor for a vacuum cleaner to get rid of things (albeit charring things down to the ground might not be a good idea should one need them later)[190]. In this sense, the elements which I classify as ToonTalk tools are the ones presented in Table 15.

All elements in ToonTalk are "cartoons", even the tools. In line with the language and environment metaphor, they possess animations. Table 15, which details the tools, presents examples of such animations.

[190] An alternative computer science perspective could be to disregard the existence of tools altogether, since all their purposes are potentically replaceable by a programmer persona. As I describe further ahead in this section, the syntax of ToonTalk is not dependent on the tools themselves but on the static primitives (which can be seen as values) and actions upon them, which I call "animated primitives" (and can also be seen as operations). Conceivably, any animated primitive can be replaced by a completely different one, as long as its effect and applicability remain identical, and the language would still be ToonTalk, even if the overall appearance changed a lot: " *The ToonTalk world resembles a twentieth century city. There are helicopters, trucks, houses, streets, bike pumps, toolboxes, hand-held vacuums, boxes, and robots. Wildlife is limited to birds and their nests. This is just one of many consistent themes that could underlie a programming system like ToonTalk. A space theme with shuttle craft, teleporters and the like would work as well. So would a medieval magical theme or an Alice in Wonderland theme*" (Kahn, 1995).

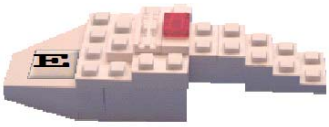
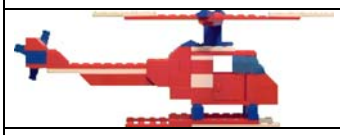
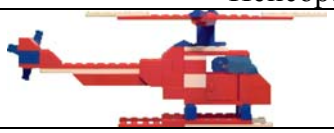
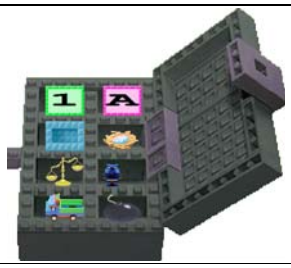
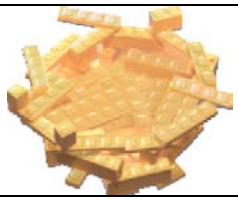
ENVIRONMENT TOOLS			
Visual (when not used)	**Visual (in use)**	**Modes of operation[191]**	**Location**
Dusty the Vacuum			
		S – Suck up element R – Reverse (spit out) E – Erase surface	Inside Tooly the Toolbox. Pops out when the programmer kneels.
Helicopter			
		Flying	Landed on city streets. Can be summoned with the “F1” or “H” keys.
Maggie the Magic Wand			
		C – Copy element S – Copy itself O – Copy original picture of an element	Inside Tooly the Toolbox. Pops out when the programmer kneels.
Marty the Martian			
N/A		Providing help with text balloons Providing help with audible speech	Shows up whenever the programmer is inside a house. Can be sent away or summoned with the F1 key.
Pumpy the Bike Pump			
		B – Make big L – Make little W – Make wide N – Make narrow T – Make tall S – Make short G – Make “good” size[192]	Inside Tooly the Toolbox. Pops out when the programmer kneels.
Tooly the Toolbox			
		Following the programmer Open on the floor or in hand	Trails behind the programmer when he/she is walking or standing up. When the programmer kneels, Tooly opens up.

Table 15 – ToonTalk tools and their purpose

[191] The modes represented by letter buttons are presented as they appear in the US English version, but are found localized in other language versions of ToonTalk.
[192] In ToonTalk, this means each object’s default size.

The remaining visual elements in ToonTalk are primitives, but they do not constitute the entire set of primitives. As expected in an animated language, several primitives are animated actions, not just visual elements. They give the programmer the power to express a program in ToonTalk. Table 16 lists the static primitives and Table 17 the main animated primitives available in ToonTalk.

STATIC[193] PRIMITIVES		
Primitive	**Visual representation**	**Purpose**
Bird		Message[194] sending.
Bomb		Termination of a set of programming elements (see row on "house") or of a container (see row "container").
Container	See rows on "Image", "Text pad", and "Number pad".	Holding a set of programming elements, running concurrently with access to common properties such as "distance from the left side", "speed to the right", "collision state", or even other primitives. All images, text pads and number pads are containers[195].
Cubby box		Organization of elements in arrays.
House		Holding a set of independent programming elements. In computer-science terms, it can be seen as an object, a process, an actor, etc.
Image		Representation of graphical data.
Infinite stack	Identical to the static primitive used in its creation.	Template for instantiation of a static primitive.
Nest		Message reception. FIFO[196] container of received messages

[193] Most of these primitives are not visually "static", they possess animations. The term "static" is used because those animations serve no specific programming purpose, and are employed solely for aesthetic or metaphoric reasons.
[194] In the ToonTalk software environment, a message can be one of the square-shaped static primitives or a scale, but a broader view of just the ToonTalk language, not its current environment, could envision it as being any static primitive.
[195] This is the current status of the ToonTalk language and environment, which does not possess an actual specification document. Envisioning such a specification, one could consider all static primitives as containers, and the current ToonTalk software simply as a language implementation were only some of those containers are usable as such.
[196] First-In, First-Out.

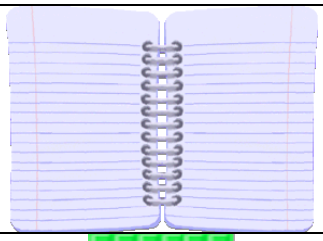
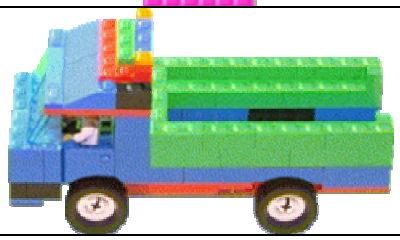
Notebook		Organization of elements in collections[197] of templates.
Number pad		Representation of numerical values (static data or otherwise), as numerical constants, variable values, or arithmetic expressions. See also the row on "container".
Robot		Representation of a sequence of operations (*i.e.*, a procedure).
Robot thought bubble		Representation of constraints on the operation of the robot.
Scale		Representation of the result of comparisons (static to specify a result, animated to specify indetermination).
Multimedia pads (sound pad, force-feedback pad)		Representation of audio (both sound data and text-to-speech) or multimedia effects (currently, only joystick force-feedback).
Text pad		Representation of alphanumerical strings (static data or otherwise). See also the row on "container".
Truck		Spawning of new sets of programming elements (see table row on the primitive "house"). Container for primitives related to spawning.

Table 16 – ToonTalk static primitives

[197] The term "collection", used here, aims to convey a distinction between notebooks and arrays (which are represented by cubby boxes). While ToonTalk collections are ordered (*i.e.*, notebooks have page numbers), just like arrays, they are distinguishable from ToonTalk arrays in several aspects. Firstly, they are indexed: in a notebook, a program can directly access an element, using a value that only becomes defined at runtime (either by page number or textual content of odd-numbered pages); cubby box arrays must be manipulated or iterated. Secondly, notebooks do not have a defined size: they hold a finite set of elements, but the only way to specify a number of "empty" cells/pages is to have a non-empty cell after them in numerical page order. Thirdly, notebooks are not divisible: there is no primitive to split the contents of one notebook over two notebooks, whereas with cubby boxes one can readily do that operation.

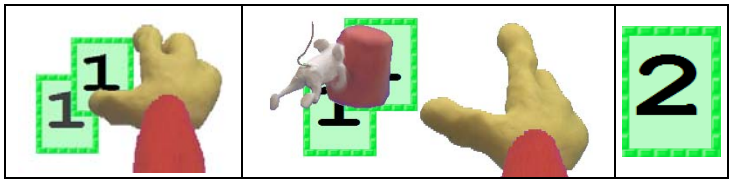
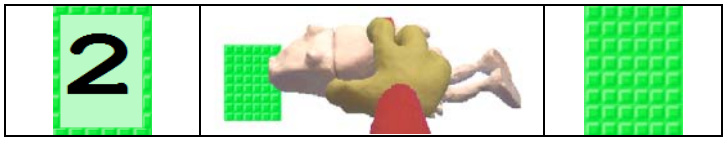
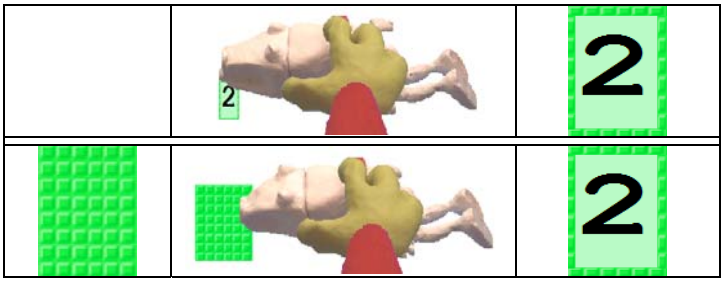
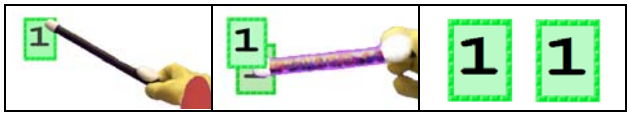
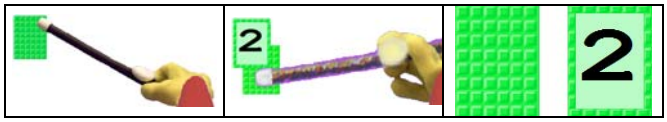
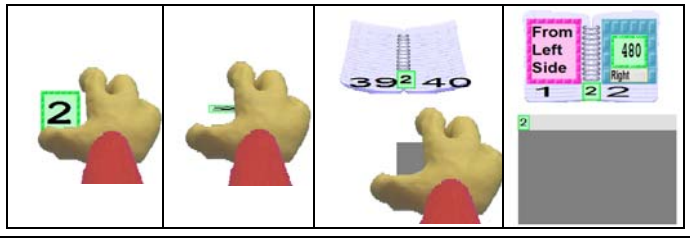
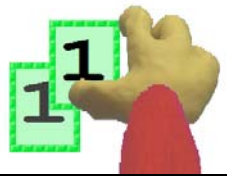
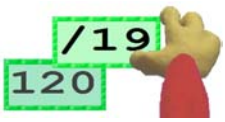
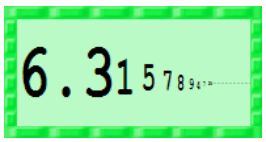
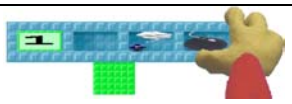
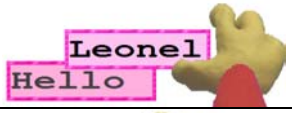
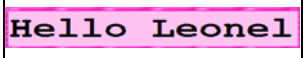
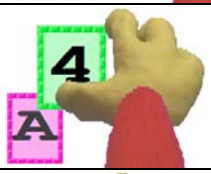
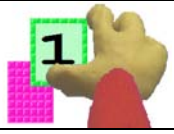
MAIN ANIMATED PRIMITIVES		
Primitive	**Animation snapshots**	**Purpose**
Grabbing and dropping		Repositioning of static primitives – OR – removal from a container or array – OR – instantiation of a static primitive from a template.
Dropping on		Combination of static primitives. The actual operation depends on the nature of the primitives being combined. Not all combinations are valid.
Vacuuming		Elimination of a static primitive.
Erasing		Generalization of a static primitive. Not applicable to all static primitives.
Spitting		Recovery of a static primitive that was recently vacuumed – OR – recovery of a defined state from a recently generalized static primitive.
Magic wand copy		Duplication of a static primitive.
Magic wand copy of original		Creation of a duplicate with a defined state, from a generalized primitive.
Flipping a container		Access to its contents, for addition, removal, commenting, and labeling.

Table 17 – ToonTalk main animated primitives

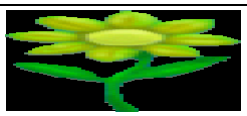
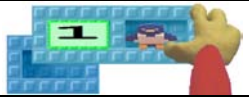
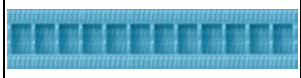
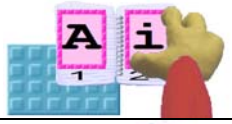
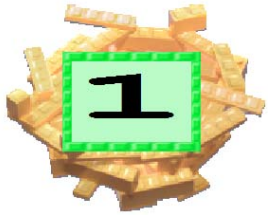
The tables in the last few pages presented detailed lists of ToonTalk environment tools and language primitives, but they still fall short of providing an actual picture of its programming. A crucial element is the behavior of the "dropping on" primitive, included in Table 17. As I stated there, "*the actual operation depends on the nature of the primitives being combined*". In the visual example included in that table, for instance, a number pad for "1" was dropped on another "1", and the result was that they were added, thus resulting in a number pad for "2".

However, for other primitives, the result can be quite different: for instance, dropping an array on a robot without constraints results in the initiation of the programming of the operations of that robot. (This particular case is a basic building block of ToonTalk programming.) Yet other combinations of primitives produce results such as sending a message, spawning a process, or defining a template. For this reason, the following table details the results of the combination of different static primitives, by dropping one on top of another.

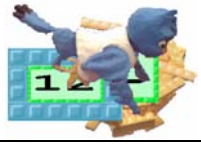
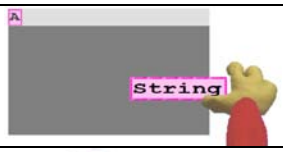
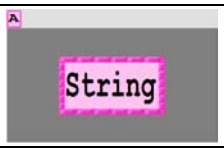
"DROP-ON" COMBINATION OF STATIC PRIMITIVES – Main results			
Involved primitives	**Visual combination**	**Visual result**	**Result description**
Number pad on number pad			Result of the arithmetic operation[198]. Default operator is "+".
			
Array on number pad			The array is split in two: the first *n* elements in one (*n* is the value of the number pad) and the remaining elements in another.
Array on erased number pad			Number pad with the number of items in the array (number of cubby boxes).
Text pad on text pad			Concatenation of text strings.
Number pad on text pad			The letter index is increased.
Number pad on erased[199] text pad			Alphanumeric representation of the numerical value in the number pad.

[198] ToonTalk arithmetic supports arbitrarily large numbers, exact fractions, and irrational (approximate) numbers. It provides some interesting solutions for displaying numbers with tens of thousands of digits, or even with an infinite number of digits (Kahn, 2004). For instance, the rational number 120/19, above, is displayed with an ever-shrinking sequence of digits. If the programmer enlarges the size of the number pad, more digits become visible.

[199] *I.e.*, a generalized text pad.

Image, number pad or text pad on image			The dropped-on image or pad becomes an independent part of the bottom image.
Image on erased image			The bottom image acquires the visual looks of the dropped-on image.
Static primitive on cubby box	 	 	Placement of the dropped-on primitive inside the cubby box.
Cubby box on side of cubby box			Concatenation of the two arrays.
Text pad on erased cubby box			Text converted into an array with a single-letter pad in each hole.
Number pad on erased cubby box			The erased box becomes an array with as many empty boxes as the dropped-on number.
Notebook on erased cubby box			The notebook templates are instantiated. Each element is placed in sequence in the resulting array.
Robot team on erased cubby box			The erased box becomes an array with as many holes as there were team members. The robot team is split, and each member is placed into a hole.
Static primitive[200] on bird			The bird drops[201] the static primitive (and copies of it, if necessary) on all its associated nests[202].
Static primitive[203] on empty nest	 		Placement of the dropped-on primitive inside the nest.

[200] In the current implementation of ToonTalk, only rectangle-shaped static primitives and scales can be used (see footnote 194, on page 202).
[201] *I.e.*, the animated primitive "drop on" results in another animated "drop on" primitive.
[202] If there is more than one nest associated with a given bird, then the static primitive is duplicated (copied), and each nest receives a copy. A different perspective on this could be that the static primitive is turned into a template upon being dropped on a bird, and from that template several instances are created, one for each nest.
[203] As mentioned in footnotes 194 (page 202) and 200 (page 206), in the current implementation of ToonTalk only rectangular static primitives and scales can be used.

Static primitive, carried by bird on non-empty nest			Placement of the dropped-on primitive inside the nest, below[204] existing contents.
Nest on empty nest			The two nests are joined. Birds associated with either nest become associated with the resulting nest.
Static primitive on flipped container			Placement of the static primitive inside the container.
Array on untaught robot			Change of setting to the robot's thought bubble; until exit from this setting, all actions (animated primitives) are recorded as that robot's sequence of operations. The status of the dropped-on array is recorded as that robot's constrains.
Array on taught robot			The robot's constraints are compared to the dropped-on box. If they match, the robot executes its sequence of operations repeatedly, until the constraints cease to be valid.
Array on thought balloon of taught robot			Assignment of new constraints to the robot's sequence of operations.
Array on taught robot whose thought balloon has been vacuumed			Retraining. The setting changes to the robot's thought bubble, where the robot replays its sequence of operations, either until the end or until the programmer interrupts it, by moving the mouse. Any subsequent actions are recorded following the replayed ones. (Unreplayed actions are discarded.)
Robot on robot			Robot team. From the viewpoint of all other primitives, a team is a single robot. Within the team, if a dropped-on array fails to meet the constraints of a team member, the array is passed on to the next team member in line.

[204] The picture in the "visual result" column was edited for the benefit of printed clarity. In ToonTalk, the presence of items "behind" the first is not visible until the first one is removed.

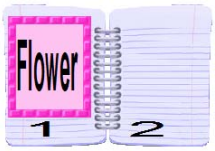
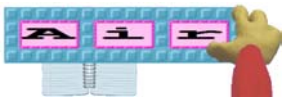
Robot AND array on truck (in any order)			Spawning. A new house is created. The truck drops off the robot there and drops the array on the robot.
Static primitive on empty page of a notebook			The static primitive becomes a template, stored in that page of the notebook.
Number pad on base of notebook			The notebook opens on the page matching the dropped-on number.
Text pad on base of notebook			The notebook opens on the left page whose textual contents match the dropped-on text.
Array on erased notebook			The array elements are converted into templates, one in each page of the resulting notebook.

Table 18 – Main results in ToonTalk of using the "drop on" primitive

Examples of programs performed using the primitives and combinations just presented can be found in Figure 181, on p. 198 (the "exchange" program by Ken Kahn), in Annex I, on p. 441 (the "Writes Maria" program, by a 4-year old girl named Maria), and in Figure 77, on p. 137 (a program to spawn a child-process and send it a message). An example using two robots that communicate amongst them is presented further ahead in this section, on Figure 194 (page 213).

In summary, a ToonTalk program is built by programming robots. This programming is achieved by executing a sequence of actions such as in this example, for exchanging two numbers:

1. Create an array with a starting example: elements that define the robot's constraints (Figure 186).

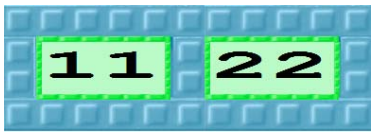


Figure 186 – Starting example for a sample robot

2. Drop that array on an untaught robot (Figure 187).

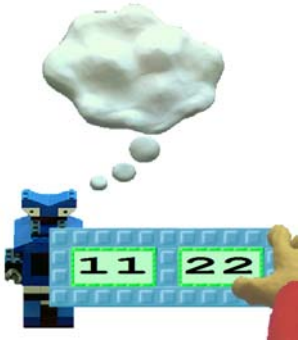


Figure 187 – Dropping on a robot an array with the starting example.

3. Upon entering the robot's thought bubble, use the mouse to control the robot, and perform the desired set of actions on the constraints (see Figure 181 on page 198).
4. The result is a robot with a set of constraints on its operation (Figure 188).

Figure 188 – Taught robot, thought bubble identifies constraints.

5. The constraints of this robot can be generalized[205], allowing it to work on any numbers or even any other array contents. This is achieved by erasing the number pads (meaning "any number") or vacuuming them (the empty-hole constraint means "any content"), as can be seen in Figure 189.

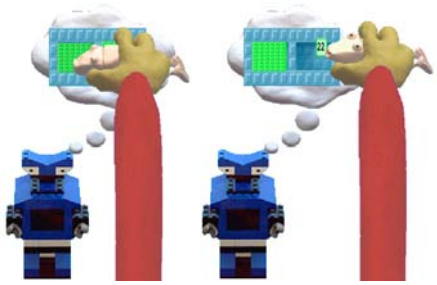


Figure 189 – Generalizing constraints in ToonTalk: erasing and vacuuming

Such robots can be activated by dropping on them an array with which to work. For instance, a robot such as the one in Figure 189, with generalized constraints, will accept any of the arrays in Figure 190, but not the arrays in Figure 191.

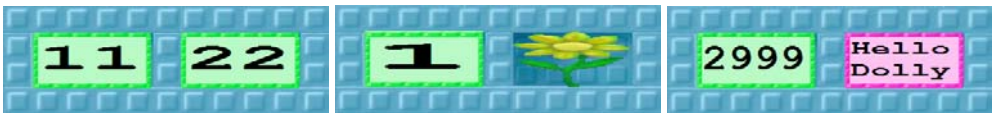


Figure 190 – Arrays acceptable for the robot of Figure 189

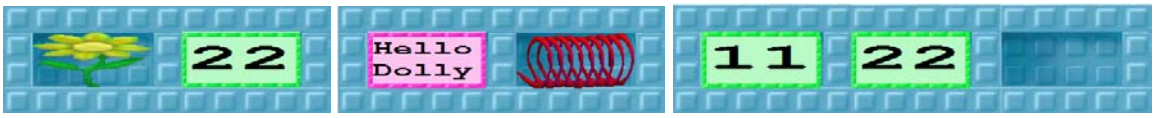
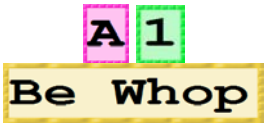
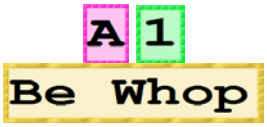
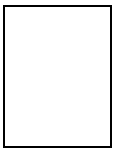
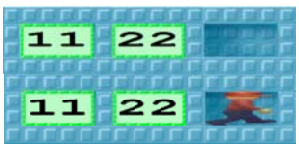


Figure 191 – Arrays unacceptable for the robot of Figure 189

In Figure 190, all the arrays have a number on the first hole, matching the first-hole constraint of the robot in Figure 189 ("any number"). The contents of the second hole don't matter, because there isn't any constraint on the contents of it.

In Figure 191, the first and second arrays don't have numbers in the first hole, and so don't match the first-hole constraint. The third array does have a number in the first hole, but the box is a three-hole box, and the robot's thought bubble constraint on box size specifies a two-hole box. Table 19 presents a summary of these matching rules. A point to note in matching is that labels, comments, and names are never considered. In ToonTalk, all these are simply provided for the convenience of the programmer (or anyone curious about what the program does), and do not constitute programming elements (*i.e.*, one cannot lookup a bird named "Lolly", a box labeled "X", nor a notebook labeled "Samples").

[205] If the robot constraints are not generalized, then it will only act on an identical array, which renders the programming little more than macro recording (in the presented example, the robot will swap "11" with "22", but not "22" with "11", nor any other combination of numbers). The generalization of constraints is how "*any program can be created by working with examples*" (Kahn, 2001b).

SUMMARY OF CONSTRAINT-MATCHING RULES IN TOONTALK			
Constraint in thought bubble		**Valid matches (in boxes handed to robot)**	
Any constraint but empty nests.		An empty nest is not checked to see if it matches; rather, the checking is suspended until an object arrives in the nest, and that object is then checked.	
Empty nest		Empty nest.	
Ungeneralized pads (text, number, sound)		Pads which present the same exact values (the contents on the back are not checked). Numerical formats do not affect the matching (*i.e.*, 1.5 matches 1.5, 1½ and 3/2).	
Generalized pads (erased pads)		Same-type pads regardless of appearance or contents on the back.	
Ungeneralized images		Images whose base pictures are identical, regardless of width and height (the contents on the back are not checked).	
Generalized images (erased images)		Any image, regardless of the appearance or contents on the back.	
Empty boxes		Boxes with the same number of holes (regardless of contents).	
Boxes with contents		Boxes with the same number of holes, where the matching applied to each hole is valid.	
Mixed boxes (some holes filled, some empty)		Boxes with the same number of holes, where the matching applied to each hole is valid (any content is valid for the empty holes).	
Generalized boxes (erased boxes)		Boxes with any number of holes, regardless of content.	

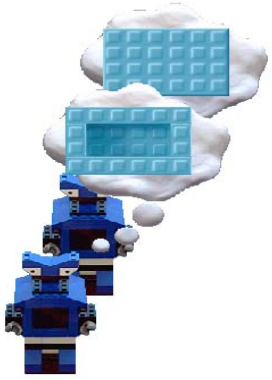
Scales		Scales in the same state (balanced, left-tilting, right-tilting, or wobbling)	
Robot or robot team		Any robot team whose first members are copies of the same robots, in the same sequence, regardless[206] of their names, constraints, or secondary members of the team. (Robots and teams can be erased to match any other robot or team.)	
Birds, bombs, or notebooks		Respectively, any bird, bomb or notebook (notebook contents and state are not matched).	
Empty trucks		Any truck, regardless of contents.	
Trucks with cargo		Trucks whose cargo matches.	

Table 19 – Summary of constraint-matching rules in ToonTalk

ToonTalk provides visual cues to help the programmer detect these constraint problems. When a constraint is not fulfilled, the robot stops and that failing constraint is highlighted in red (Figure 192).

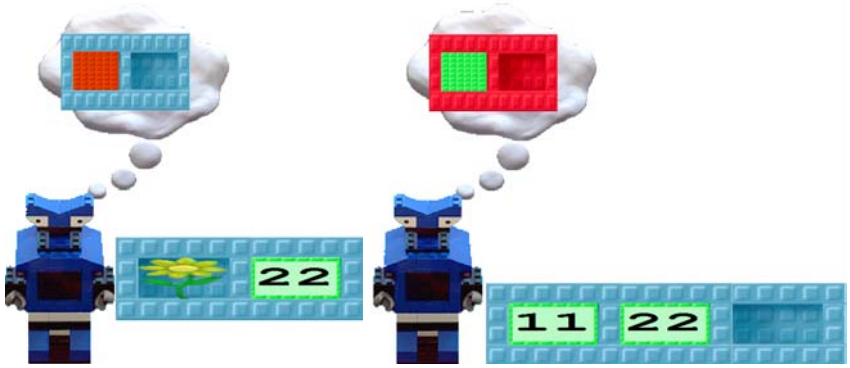


Figure 192 – ToonTalk robots after constraints failed to match

These visual cues are one of the features of the ToonTalk system that help the programmer understand the program execution. But perhaps the most important feature on this regard is the possibility of observing the execution of a program at various speeds and levels of complexity, as I'll now explain to conclude this section.

[206] In this row, the rightmost cell shows two sample matching teams. Firstly, a team that is identical to the constraint. Secondly, a team where several changes occurred, but that still matches the constraint: a new robot was added to the team, but since it is located behind the first two, it is not considered for matching purposes; and the constraint of the second robot was changed, but the robot itself is the same.

After programming a robot, there are four different ways to make it execute its sequence of operation. These provide for different program requirements, but also contribute to help the programmer understand the execution of the program. They are:

1. Put the robot on the floor and directly drop on the robot the box for it to work with.
2. Put in a truck both the robot and the box with which it will work (the truck will do the dropping-on, inside a new house).
3. Put the robot on the back of a picture and directly drop on the robot the box for it to work with. The robot will start working when the picture is flipped over.
4. In any of the three previous methods, replace some of part of the box contents by nests; the robot will only start working when static primitives are eventually dropped on those nests.

The **first method**, presented in Figure 193, is a typical test (or debugging) situation. Upon being given a box matching its constraints, a robot will proceed with the execution of its sequence of operations, under the gaze of the programmer, at a speed identical to that used by the programmer to perform the same actions (albeit without pauses).

This method allows one to finely observe the behavior of a robot with a specific array, to determine whether its actions are the intended ones. However, since the execution proceeds at human pace and with the robots in view, it is usually not suited to most execution situations. But this not a real limitation, because the programmer persona can get up, and leave the house where the robots are working. By doing so, the robots will execute at full speed. Therefore this method can be suitable for all situations in which the visual presence of the robots does not place a problem. The programmer can let the computation proceed at full speed inside the houses, and simply enter a house to see the robots working in detail. There are two levels of detail and execution speed for such observations: while the persona is kneeling, the robots execute with full detail, at a speed similar to that of a human programmer; when the persona is standing up, the detail level is reduced and the execution speed is increased.

Figure 193 – First execution method: dropping the working box on a robot and watching it work.

The **second method** means that the moment the truck departs, the computation is running, but not under the eye of the programmer. This is an adequate situation in several cases. For instance, if one already debugged the program reasonably well and simply wants it to execute and terminate autonomously; or if one wants it to run continuously and simply observe the computation results indirectly, as messages being delivered to a container or some other remote effect (changes to images in view for example).

The **third method** is similar to the second, in that the robots will be executing at full speed, and the programmer can only observe indirect effects. Its difference lies in practical issues. For instance, if one wants to inspect all robots manipulating an image's properties, it is easier to find

them stored behind that image, rather than having to determine which houses hold them, amidst many others. Also, sets of robots working behind pictures can be placed as a group behind yet other pictures, or stored together in a notebook, along with their assigned boxes. This simplifies the combination and reuse of different programs. Finally, a new set of robots can be activated rapidly, by simply creating a copy of a picture with robots behind (by copying an existing picture or by instantiating a template), rather than having to spawn a new house for each robot in a group.

The **fourth method**, as stated in its description above, is a modification to the three previous methods. It is in fact the technique employed in ToonTalk programming to connect robots: sending message through birds. In Figure 194, for instance, the robot on the right is working, but doing nothing: it is waiting until something arrives in the nest, for then to check it against its constraints (it has to be a number). The nest belongs to the bird on the left[207]. The left-side robot, called "Adds 1", has been taught to increase its left-side number by 1 and drop a copy of the result on the bird. The bird will take it to the nest in the right-side robot, which will then proceed[208].

Figure 194 – Two robots connected through a bird-nest pair.

An example can clarify the way in which this technique may also be used to help the programmer understand the behavior of a program. The robot on the right accumulates the received values in the hole labeled "sum". Supposing that the number it holds is not just any number, but a control for a picture's width, then running this robot on the back of a picture (while the other, in another house, provides numbers), will simply make it grow large very fast (Figure 195), because the left-side robot will provide new numbers quickly and the right-side will update the picture at an identical speed.

Figure 195 – Dispatching one robot, with a small figurine width, and six seconds later.

Should the programmer find this behavior odd, he/she may want to inspect the impact of the width-changing robot. But waiting for its animation to unroll for each received number may be tedious. Using the bird and nest can be helpful. The robot can stay behind the image, working on its width, and the programmer can hand new numbers to the bird directly, watching the resulting effect on the picture width, rather than having to wait for the animation of the robot (Figure 196).

[207] As mentioned before, the name of the bird and the label on the nest take no part in this whatsoever: the connection of the bird with the nest is an intrinsic (and immutable) property of both.

[208] This pair of robots produces the following results: on the left, at each iteration the number will increase by one, so the number will be 0, 1, 2, 3, 4, ...; on the right, the number dropped on the nest is added to the value already in the hole labeled "sum", which will be "0, 1, 3, 6, 10, …", *i.e.* the result of 0+1+2+3+4+…

Figure 196 – Sending one number to the robot on the back of the figurine, using the bird.

The manipulation of controls for a picture, as mentioned here, is perhaps easier to understand if detailed a bit further. The "controls" themselves are regular primitives (text pads, number pads, boxes, images) that present special behaviors. This is how ToonTalk links itself to the state of the programming system. Terminology-wise, these primitives with special behaviors are only called "controls" when they provide access to the state of an internal ToonTalk container element (a picture, for instance); they are called "sensors" instead, if they provide access to the state of the computer system or the overall ToonTalk programming environment (mouse position, pressed keys, looks of the current ToonTalk house[209], etc.).

The sensors can be found in the "Sensors" notebook, within ToonTalk's main notebook. The controls, being specific to a container, can be found in the back of each container, which can be accessed in several ways, while the container is being held: pressing "F" for "flip"[210]; holding down a shift key and left-clicking the mouse; and, if the user activates the appropriate program customization option, right-clicking the mouse. Figure 198 present the resulting animation, with a picture container being flipped to reveal its contents (in this case, it's empty) and the notebook of controls flying out of it.

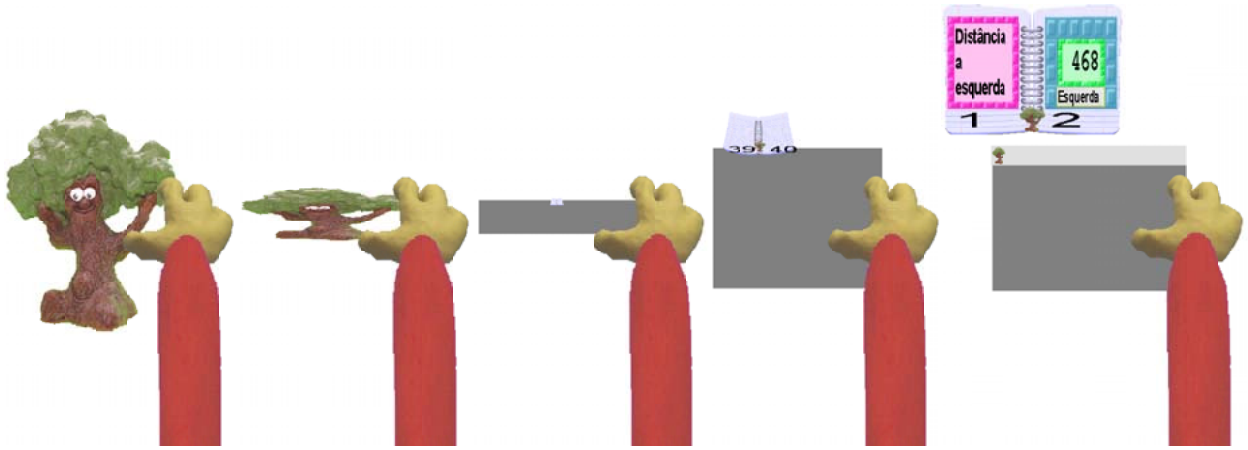
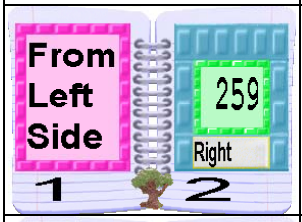
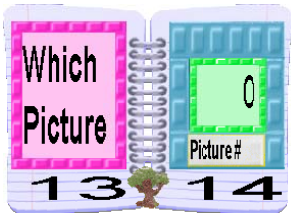
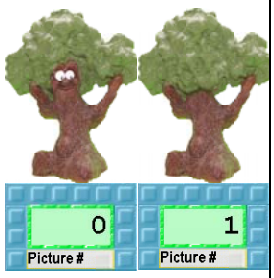
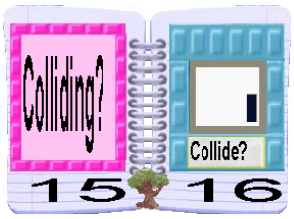
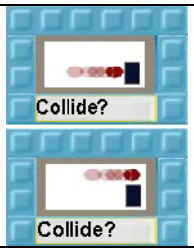
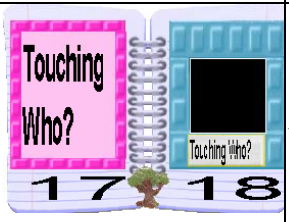
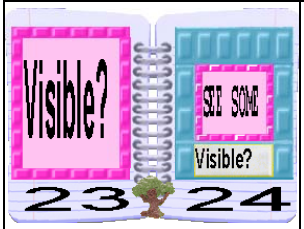


Figure 197 – Flipping a container to access its contents and notebook of controls.

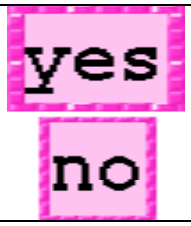
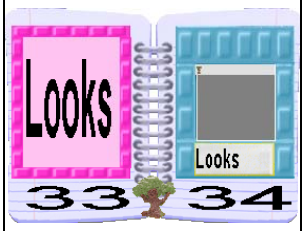
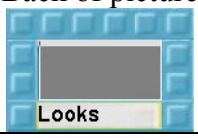
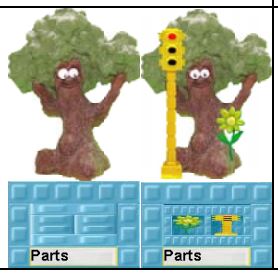
Repeating the flip operation returns the container to its original state, and the notebook of controls returns to its back.

Table 20 presents an assortment of sample controls and sensors, selected for the purpose of presenting the varieties of this kind of elements. It should be noted that sensors and controls commonly present some animated cue, in order to render them distinct from regular, static-looking objects. Pads have a "light marquee" outline, and picture controls flash regularly, for example.

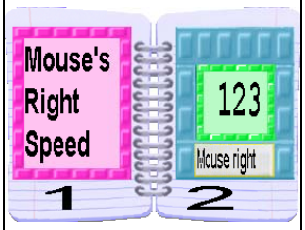
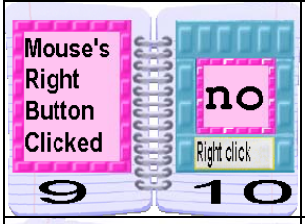
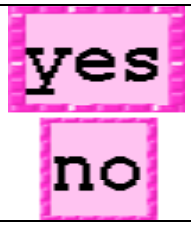
[209] There is a possible inconsistency in this distinction in that some sensors, such as the looks of the current house, are in fact local to a ToonTalk element.
[210] Localized versions of ToonTalk employ different keys. For instance, the European Portuguese version uses "V" since the word is "Virar".

SAMPLE PICTURE CONTROLS			
Image and name	**Description**	**Readable / Writable**	**Possible values**
	Distance from the left side of the containing image or viewport[211], measured from the center. Writing to it (dropping a number pad on top or editing with the keyboard) changes the horizontal position of the picture.	R + W	Rational numbers.
	Width of the picture, relative to the containing image or viewport.	R + W	Positive rational numbers.
	Index of the image providing the current looks of the picture (a ToonTalk picture can be a sequence of images). For synthetic pictures, and text and number pads, it controls the foreground color. 	R + W	Positive integers, from 0 to the number of internal images.
	Shows whether an image is "colliding" with another ToonTalk element[212] (*i.e.*, if some part of it occupies the same coordinates as another element). Can be changed to make them "uncollide", which causes the picture to move slightly aside, to end the collision.	R + W	
	If a picture is colliding with another, this control acts like a surveillance camera and shows that other picture. Dusty can be used to vacuum it from the control and spit it, so that a program can access that other picture's contents and controls.	R	Any picture. 
	Determines whether a picture is fully visible, invisible or partly transparent[213]. The programmer can change the current value by pointing to or holding the sensor and pressing the "+" and "-" keys.	R + W	

[211] When a picture is on the floor, rather than on top of another, ToonTalk coordinates are relative to the area visible to the programmer, from 0 (leftmost side) to 1000 (rightmost side), and 0 (bottommost side) to 1000 (topmost side). Images placed outside of the currently visible area have coordinates that are either negative or greater than 1000.
[212] The "collision" and "no collision" pictures are animated icons, displaying a ball hitting or missing a wall, respectively.
[213] If the format of the original picture file does not support transparency, ToonTalk uses black as the "transparent color" for this sensor.

	Shows whether a picture is being held by the hand of the programmer's persona or not.	R	
	Provides access to the appearance of the picture/pad, which can be edited as any picture/pad (erased, things can be dropped on it, etc.). It can also be flipped to provide access to the contents of the back.	R + W	Front of picture  Back of picture 
	Shows which subpictures are part of this picture. Dusty can be used to access these subpictures and manipulate their contents or controls. 	R	Array of containers: pictures, number pads, text pads.

SAMPLE TOONTALK SENSORS

Image and name	Description	Readable/ Writable	Possible values
	Displays the current horizontal speed of the computer system's mouse. 	R	Integer numbers.
	Indicates whether the right button of the computer system's mouse was just clicked or not.	R	
	Indicates the key that was most recently pressed on the computer system's keyboard.	R	A text pad indicating the letter. 
	The value of this sensor is constantly changing, in pseudo-random manner.	R	Any integer number between 0 and 999.

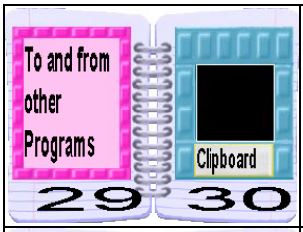
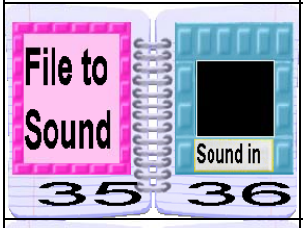
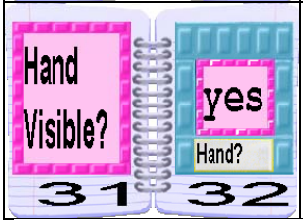
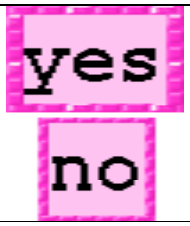
	Presents, as a picture, the current contents of the MS Windows system clipboard. Dusty can be used to vacuum it and spit, in order to access these contents. ToonTalk elements can also be dropped on it, to copy them to the clipboard.	R + W	Any ToonTalk static primitive (contents from other programs are converted).
	The programmer can drop a filename or URL on this sensor, which will respond by producing a sound pad with the sound found in the matching location.	R + W	Any sound pad.
	Determines whether the hand of the ToonTalk programmer's persona[214] is visible on-screen or not.	R + W	

Table 20 – Sample ToonTalk controls and sensors

[214] If the persona is standing up, this affects the entire persona; if the programmer's persona is flying in the helicopter, this affects the entire helicopter.

3.4. Cookbook of computer programming topics

Computer programming is a vast area, and while using it to learn other subject matters, be they mathematics, social science, biology, language – or whatever – one undoubtedly also learns about computer programming itself. But this haphazard learning of computer programming, not a conscious focus of either the learner or the teacher/educator, is often limited in scope. Frequently, both students and teachers end up using and re-using the same limited set of programming techniques, and going beyond them requires deep personal involvement, in terms of both time and mental effort. As a consequence of this, teachers and students quite often end up unable to take intended paths of study using computer programming (Hoyles & Noss, 1999).

In my view, the ideal situation would be to have a computer programming consultant available to teachers/learners. Should a teacher/learner come across a situation where his/her programming skills were insufficient, the consultant could suggest an adequate new programming technique or approach. This way, even the learning of programming would occur with a specific goal, embedded in the overall educational goals, not just for its own sake. This bears some similarities with the approach used in preschools of the Italian city of Reggio Emilia, where art educators, not just teachers, are involved in the educational activities (New, 2000), as described in section 4.1.3, p. 273.

For most educational settings, this ideal situation is not an available option, either now or in the near future. This section therefore aims to provide some help to teachers and learners using computer programming on their own. It is a list of various programming techniques, their translation in ToonTalk terms, and ideas for educational activities – *i.e.*, uses beyond learning programming. They may, however, include activities to learn about computers in general: computers being an important part of everyday living, learning about them fits within a content area for preschool education commonly known as "knowledge of the world" (Ministério da Educação, 1997).

I've called it "cookbook", because that's how I see its usefulness. As explained in detail in section 3.1, a teacher should not feel the obligation to try all these techniques. Rather, my intention is that this resource allows teachers to try out a new computer programming concept in the context of everyday educational activities. Most people that read cookbooks don't feel pressed to cook all the recipes in them. One browses cookbooks in search of dishes that one may "feel like" trying at a given occasion, either because they appeal to one's tastes or suit the ingredients (or time) available.

By trying a new "programming dish" from time to time, or at least being aware of the number of "dishes" that he/she still didn't "cook", a teacher/learner may use this section to advance programming knowledge with hands-on uses beyond programming itself: "*making learning worth while for use now and not only for banking to use later*" (Papert, 1999, p. xvi).

3.4.1. Computability

Computer programming concept	Preschool education connection
Not all conceivable functions can be implemented with a computer. Some are mathematic impossibilities[215], others can only complete in a time longer than the life of the universe (Mitchell, 2003, ch. 2), or require more storage space than the number of particles in the universe[216].	At times, the computer may seem to be a "magical" object. Obviously, it is not such a thing, but a technological object. Children can learn about inherent limitations in computation, realizing that the computer is just a machine, with real-world limitations.

Activity guidelines

By involving children in the design and planning of activities that employ the computer, opportunities are likely to arise for addressing the matter of what a computer can, and cannot achieve. But this is a tricky matter, particularly for preschool teachers or any educator without a strong computer science background. There is the serious risk of confusion between what a given computer cannot do, for lack of resources, and no computer at all can do. In the physical-world field, preschool teachers risk electing to state that computers cannot do something like, for instance, turn on and off the lights – while, in fact, they can, with adequate interfacing materials. Even things like deciding if a person is "pretty" or not are not safe to address: they include a large degree of subjectivity, but also a large degree of measurable real-world phenomena, such as facial proportions, symmetry, etc.

One option for educators to avoid this problem is to focus computer limitations entirely on the field of human cognition: a computer can't answer questions such as "am I a good person?", "do I mean well?" etc.

As an example, the following events took place in December 2004, while one of my college students, from studies in Early Childhood Education, set out to program an activity where children would have to match generic geometrical shapes to those composing a picture. For instance, in Figure 198 the arms are rectangles, so green rectangles from the left should be dropped on top of each arm. Halfway through, she was troubled that children might want to drop a circle on the clown body, but by accident drop it on the head instead, and since both shapes are circles, a match would occur. She had partially worked around the situation by using circles of different colors, but still, there were times when that solution wouldn't suffice (identical shapes for the left and right hands or arms, for instance). As things stood, dropping a shape on an unintended place might cause positive feedback, something she didn't intend to happen.

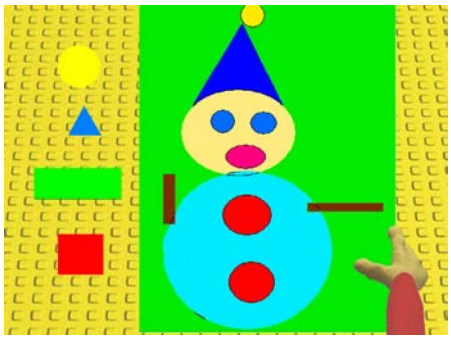


Figure 198 – Matching shapes but not intentions.

[215] Like the well-known example known as the "halting theorem": creating a function to determine whether any given program will terminate or not (Mitchell, 2003, pp. 14-15).

[216] It can be argued that this limitation (storage space) is not independent of the previous one (execution time), because in order to use a storage location, a computer system must also access it. However, in common computer systems, storage space must be allocated before being accessed, and the complexity of the execution of an allocation operation is often independent of the actual amount of storage being allocated. Therefore, one can imagine a situation where the execution would take place in a viable amount of time, but the storage requirements are a physical impossibility.

But she had a situation ripe with opportunities to address the matter of what a computer can and cannot do. Rather than trying to create a play environment where children couldn't find anything odd at all, she should instead focus on the learning opportunity that such an event (dropping a shape on an unintended place and getting odd congratulations) would pose for a preschool-age child.

I said[217], "*Look, this is in fact OK as it is. There is a good chance a child may tell you something like «teacher, I did it wrong and the stupid computer said I was right».*"

She replied, "*But that's exactly my problem, how can I get out of such a situation?*" She was a reasonably experienced computer user, but wanted to ignore the non-magical nature of the computer operation: she would have it working flawlessly, not bothering with its behavior. The shapes and colors mattered, little else.

"*That's a great situation*", I said, "*because the computer didn't fail or behave mysteriously. It behaved exactly as it should*." Then I proceeded, "*You can ask the child why she thinks the computer is wrong, and together realize that indeed, both the body and the head of the clown are circles*." But this shouldn't stay at that, I continued, because "*you and the child should be able to reach a point where she will realize that she thought the computer was wrong because it didn't interpret her intentions. This is the crucial element: realizing that computer will only interpret what she **does**, not what she **means to do***."

[217] The following dialogue is presented from my personal recall. It was not recorded or accurately transcribed in any way, and as such only reflects the content of the dialogue, not the actual exchange of words that took place.

3.4.2. Programming environment

Computer programming concept	Preschool education connection
Programming takes place in an actual computer system, usually using pre-existing programs. Every such system and program has particular features and requirements, which the programmer must be at least cursory aware of. A notion of particular importance is that the elements of programming (subprograms, variables, etc.) are stored somewhere. In order to develop a program, the programmer must be able to access them once stored.	In order for a child to employ a programming language, he/she must be able to use and navigate a programming environment for that language. But the environment features vary tremendously among computer systems and programming languages, and therefore actual activities focusing on the programming environment are likewise diverse, specific to each environment and language.

Activity guidelines

One can consider that even just playing around or tinkering with the features of the programming environment can be useful, in that it can allow a child to gain familiarity with it. However, my programming sessions with children led me to believe that it can also be cause for frustration and annoyance, if a child feels a lack of control, a lack of the ability to actually decide what to do and follow up on that decision.

This means that it is important to ensure that the playing and twiddling with the environment is sufficiently supported, in order to provide the child with the leverage to conduct ever more capable actions. Such support can be provided by means of specific advice, but also by setting larger goals or objectives.

An example (ToonTalk environment), is drawing the child's attention to the wiggling: when the hand of the programmer's persona points to an object, it wiggles. That provides a cue: clicking the mouse will pick up THAT object (*vd*. Figure 199). The same occurs while dropping: if something is wiggling, the drop will be on top of what is wiggling, not on top of something else or on the floor. Drawing the child's attention to that wiggling cue, straight from the beginning, can be crucial for that child's enjoyment of the programming environment.

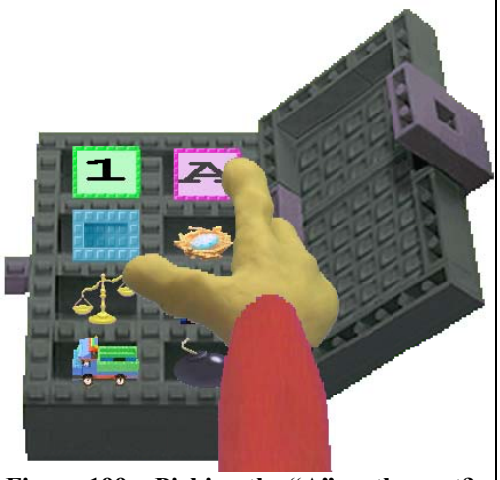


Figure 199 – Picking the "A" or the nest? The wiggling animation provides the answer.

In general, rather than keep the focus of programming activities just on the program itself, and ignore the programming environment features altogether, most activities for children can make heavy use of those – and even have some goals for discovery and exploration of such features.

For instance, ToonTalk programming activities can approach a "city" as an actual city, a place of building, decoration, destruction, and change. Rather than being just metaphors for programming constructs, all environment elements can often be used as virtual analogs to their real-world counterparts. Houses can be decorated; they can even be transformed into something not resembling a house at all, by changing the looks of the wall and roof! Birds can be used as "pets" or "dolls", not just as communication channels. And robots sending a ball to each other, or back to the child, are not just "transmitting an image", but rather "playing ball", with each other or with the programmer.

Many programming examples presented in chapter 1 address features of the programming environment, and particularly so in the two first sections of that chapter, 7.1 and 7.3.

3.4.3. Syntax

Computer programming concept	Preschool education connection
Programming languages, as all others, are not a random assortment of symbols. They must be used in proper form, within specific rules of grammar and usage – the language's syntax. Under a formal computer science point of view, the syntax of a program is its code (Mitchell, 2003, p. 48). Specifically, it is the "*various kinds of expressions (each with its associated evaluation rule)*" that compose the code of the program (Abelson *et al.*, 1996, p. 11).	A child may be able to use the programming environment (section 3.4.2, p. 221), but in order to construct a program, he/she must be able to use the syntax employed in that language. In ToonTalk, to connect two boxes together[218], for instance, one must be dropped on either the left or right-side of another; any other combination will result in dropping a box inside or alongside the other (ToonTalk's syntax was presented in section 3.3.5).

Activity guidelines

In a virtual animated world such as the ToonTalk programming environment, there would seem to be little difference between mastering the environment itself and mastering the language's syntax, since the borderline between syntax and environment seems to be tenuous. But even a little experience in ToonTalk shows that it is no so. Seemingly "strange" limitations are revealed: joining boxes, as mentioned above, can only be done by left or right sides, not by the top or bottom; birds only take rectangular objects and scales; robots only accept boxes; etc. Seemingly "strange" events occur: if one drops a picture on top of another, the dropped picture is clipped at the borders of the bottom one. Why shouldn't they both be visible on the floor? And while flipping a notebook, when does one reach its end? (I once witnessed a child flipping pages endlessly, trying to find "the end" of the notebook.) The reason these distinctions exist is the same: syntax. They came to be in order to limit potential problems and allow a richer expression of complex ideas. But they pose a challenge for educational use with young children: most of them have apparently no reason to be: *if a bird can carry a box with a truck, why can't it carry just the truck*?

These "rules of the game" can be hard to grasp in purely "that's how it is" form. But educational activities should make an effort to bring metaphoric, everyday logic to these rules. For instance, stating that "*robots like to tidy up everything, keeping everything nicely stored in boxes*", provide a reason for them to accept only boxes. That boxes piled up high can fall and cause injury may be a reason for ToonTalk not allowing them to be linked on the top or bottom; and pictures get clipped at the bottom picture's edge because that's where the paper sheet ends, and ToonTalk wouldn't want the floor all cluttered with paper snippets. Syntax, however, doesn't end there. At a more elaborate level, it dictates more subtle results. For instance, when teaching a robot, one must learn that two onlookers of the same box may not see the same thing: in Figure 200, under the ToonTalk syntax, one is teaching the robot to pick up the fourth element, not the last one nor the one containing the value "2". Understanding syntax at this level (which can also be seen as the meeting of syntax and semantics, not just a pure syntax issue) can also be useful, in general educational terms, precisely because it provides a window into the notion of how different people can interpret the same thing differently: "*The robot thought you meant fourth*."

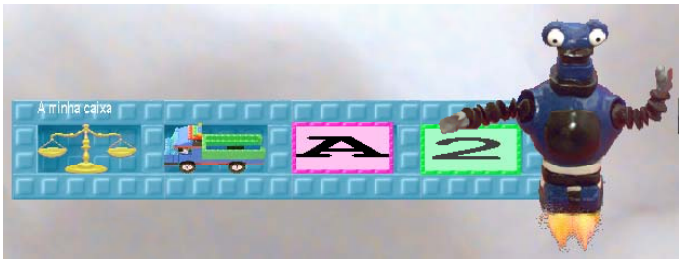


Figure 200 – The robot is picking «the fourth element», not «the "2"» nor «the last».

[218] Taking a box out of ToonTalk's toolbox (a.k.a. Tooly) creates an array. Connecting boxes is ToonTalk's syntax for creating larger arrays or joining arrays. A box inside another represents an array within a cell.

3.4.4. Declarations, expressions, and statements

Computer programming concept	Preschool education connection
These are a programming language's "*parts of speech*" (Mitchell, 2003, p. 25), as nouns, verbs and adjectives are in human languages: **declarations** create new identifiable elements; **expressions** can be evaluated to determine their value; and **statements** explicitly change the state of the system (*id.*, *ibid.*). In ToonTalk, declarations are the creation of something from a template[219] or by copying[220]; an expression is any group of primitives, and a statement is an action upon a primitive.	Children can realize that what is often simply referred to as "doing" may imply different things: **entirely mental processes** (observing, evaluating, deciding), **thinking about actions** (thinking about doing), and also actually **acting upon something** (doing). And therefore, that explaining to the computer "how to do", implies realizing that part of what is to be done is not visible, but happens nonetheless and needs to be thought of; and that "thing to do" is a separate idea, to be equated with "do to thing", as further mentioned in the coming sections.

Activity guidelines

Activities can be organized so as to allow children to realize these different parts of "doing". In child-play, for instance, a child can be a "referee", "judge" or "evaluator", and another can be the "keeper of secret gestures", "instructor" or "law agent". Upon being given a rule, the referee must decide whether it holds or not; if it does, the keeper can show what is to be done, just using gestures – in fact, the keeper may have to reason and decide what the gestures will have to be. Finally, a third child would use the keeper's gestures to actually do what the referee allowed.

In such a game, there is no mandatory need to use the computer, but also no reason not to. The referee can use ToonTalk scales to evaluate the relative size of two numbers, for instance; the keeper's gestures can also be the teaching of a ToonTalk robot or the drawing of a picture; and the subject of those gestures can be the computer or its software.

However, computer programming provides a play setting with important differences: rules are not easily bent; actions are not readily "adapted" to what they should have been; and results either occur or not, depending on what was acted upon.

The evaluation of expressions by robots determines whether they act or not (or, in a robot team, which robot will act). There is however a remarkable difference between executing actions and performing an evaluation: the former is visible, for all to see that it is happening; the latter is not visible[221] at all, one can only deduce that it happens. Educational activities can involve a degree of attention, inquiry, and even debate regarding the behavior of robots, when they work or don't.

For instance, a robot taught to write "MARIA" and give that text to a bird, starting just with a box with a bird, will work over and over again, sending an endless sequence of "MARIA" texts. However, if a robot is taught starting with the letter "M", completing it to read "MARIA" and

[219] In my ToonTalk ontology, a template is an infinite stack from which one can pick up newly-created objects. Infinite stacks are commonly found in notebooks and in Tooly the Toolbox, but the programmer can also create them using a keyboard shortcut.

[220] ToonTalk provides three methods for copying: using Maggie the Magic Wand to copy; dropping the object to be copied on a bird linked to several nests; and clipboard copy using keyboard shortcuts.

[221] In more general terms, one could however imagine a version of ToonTalk were the matching of a robot's thought bubble becomes visible: for instance, a robot could go to each hole in turn and use scales or some other method to explicitly compare the contents of the box with its thought bubble. However, such approaches cannot ultimately render everything visible as unequivocal events for a human to interpret. The broader concept is that the entire meaning perceived in communication involving human beings cannot be determined solely by the elements of syntax that compose that communication; the meaning of an expression employed in human communication depends on interpretation – semantics – and context – pragmatics (for a summary of these concepts, *vd.* Brown, 2001).

giving it to the bird, then it will only work once.

Also, when teaching a robot one has the opportunity to realize that showing the robot what to do is not the same as doing it with the robot. This becomes apparent from a common situation with novice ToonTalk programmers – both children and adults – which often are puzzled that after teaching a robot, it stays quiet, the job left undone (because one must hand it a box to work with). This situation is also an opportunity to think about what it means to "do to".

Yet another common situation constitutes an opportunity for realizing the difference between acting upon existing things, and just imagining/conceiving an action. When a person is teaching another, such differences are often simply assumed, rather than considered explicitly. For instance, one child might pick up two pebbles and swap them, at the same time saying "this is how you swap things" and leave the generalization of pebble-swapping to thing-swapping for the person that is watching. In ToonTalk, I've witnessed even adult novices trying to teach a robot about "swapping two objects" by giving the robot an empty box, teaching it to create another box with two holes, placing an object in each, and swapping their places. In doing so, the actions are indeed being done, but nothing can come out of this. The way out implies the realization that actions are only meaningful when applied to something, and that this application must be considered explicitly.

3.4.5. Conditional expressions

Computer programming concept	Preschool education connection

Conditionals are expressions whose value depends on the evaluation of conditions, formally called "predicates" (Abelson *et al.*, 1996, p. 19). The typical example is:
`If <predicate> <consequent> <alternative>`
This evaluates to `consequent`, if the predicate is true, otherwise to `alternative`. There can also be more than one predicate, such as in the common `case` expression:
`case { predicate1 consequent1`
`predicate2 consequent2`
`else alternative }`
This evaluates to $\texttt{consequent}_1$ if the first predicate is true; if not, it evaluates to $\texttt{consequent}_2$, and so forth. If all predicates are false, then it evaluates to `alternative`, here associated with the typical keyword `else`.
	The use of conditional expressions is associated with activities dealing with definition and assessment of rules. The two styles of conditional expressions on the left translate into child-play with simple rules or tabular lists of rules. But other forms can also be considered, particularly by recalling conditional expressions employed in the human languages (and in fact, also in various programming languages, with several subtle differences), which take into account notions of time and repetition: while, when, and until, for instance.
	Whatever the conditional form, its use implies not only the definition of the condition ("the rule"), but also: – what is to occur when it holds; – what (if anything) is to occur in other cases; – and what cases should be considered.

Activity guidelines

As was also the case in section 3.4.4, activities including this theme can be organized without using computer programming or even computers at all. What computer programming imposes is that the rules must be stated in a strict manner, within the boundaries of the programming language syntax. This usually implies thinking over the rule in detailed manner, lest it is meaningless.

In ToonTalk, the automatic evaluation of conditions only occurs for the constraints determining whether a robot runs or not. However, children can start to explore rules and the way to express them, without having to program a robot. Such activities are focusing on the expression and evaluation of rules (by the children themselves) rather than on expressing actions for their consequences.

As an example[222], one can consider a house functioning as a hardware store or sports shop. Other houses could be decorated as a swimming pool, a stadium, a tennis club, etc. – these could have "access rules", stating the necessary equipment: for instance, the tennis club could require a tennis racket, a tennis ball, white shorts/skirt, white shirt and white sneakers. A child playing the role of club caretaker could define and state such access rules, by assembling a poster with pictures of the necessary equipment and posting it on the wall. Another child, wanting to enter the club would look at the poster, and go to the sports shop to get the necessary equipment, putting it inside ToonTalk boxes. Upon returning to the tennis club with some equipment, the child-caretaker could be summoned to check if the rule was fulfilled, and if so, the consequence might even be some occurrence in the physical world, outside of ToonTalk.

Activities focusing on conditionals can be summed up as: the computer is used to state rules and check them; children, not the computer, are the evaluators; results can be defined and performed off the computer. This way, quite complex activities can be devised, involving various kinds of conditionals, and children focus on how to express conditions and perform their evaluation.

Using consequences external to computer programming has the advantage that they can be defined at will: children don't have to acquire the skills to program them as a prerequisite.

[222] This example was built around a "pair matching" game designed by two of my college students.

3.4.6. Compound procedures

Computer programming concept	Preschool education connection
Combining of operations into a more complex, compound operation, which "*can be given a name and then referred to as a unit*" (Abelson *et al.*, 1996, p. 12). The internal operations can be performed on static arguments, or on arguments provided to the procedure as a unit, called "parameters" (*id.*, *ibid.*).	As mentioned above (section 3.4.4), "thing to do" is different from "do to thing". The development of the notion of "thing to do" as a separate entity is also associated with the ideas of that "thing" being a collection of smaller "things to do", and the whole set having a name. Being able to define a set of actions as an identifiable procedure is connected to a helpful ability in trying to understand/or solve a problem: **problem decomposition** (being able to divide a problem in smaller sub-problems).

Activity guidelines

Children can define and use procedures, without a computer, through simple games or other activities where the procedures and their names are such as "*when I clap, we all touch our ears and touch our hips; when I stamp we all spin around and sing the merry tune; …*"

However, beyond simple defining or executing such procedures, a further step is achieved by starting with a problem, goal, or action and proceeding by dividing it into smaller problems and actions, identifying them as isolated pieces, by naming, branding, or some other identification method, such as "the left one", "the right one", "the second one", "the one with birds", etc.

For instance, in ToonTalk, it's easy to build new houses (one simply loads a truck with a robot and a box). But if two children build houses together, it's hard to tell which house was built by whom. But children can also easily put their name on the roof of a house (pick up the roof sensor, write their name with letter blocks and drop the name on the roof sensor). By teaching a robot how to put a child's name on the roof of a house, and sending that robot in the truck to build a new house, the new house will have the child's name (Figure 201). This robot can be identified in various ways, such as "that one", "the one on page 20", "the one called M", "the one near the door", etc. This usage of a robot to put one's name on the roof serves a purpose ("*whose house is that?*") but also requires understanding that the actions taught to that robot will take place as a block, and being able to refer to them as such.

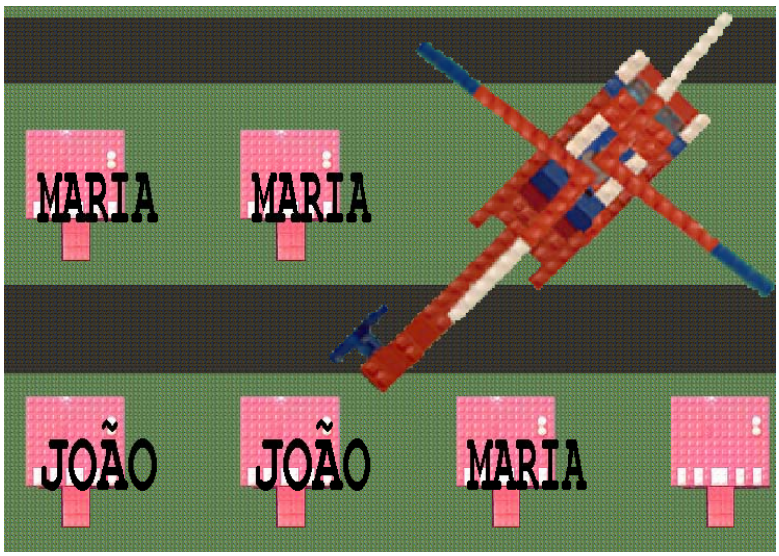


Figure 201 – In this city, children can tell which houses are "theirs".

3.4.7. Parameter passing

Computer programming concept	Preschool education connection
The operations in compound procedures (*vd*. section 3.4.6) can be applied to specific static arguments. However, general arguments can be assigned to a procedure (*i.e.*, arguments available for all the operations within the procedure); these are called the procedure's "parameters" (Abelson *et al.*, 1996, p. 12). When elements/values are assigned as arguments to a procedure, that assignment is called "parameter passing", since one considers that the parameters are being handed over to the procedure's operations.	The distinction between "to do a thing", and "to do to a thing" is completed with the realization that "to do a thing" can be done to different things. For instance, "turn left" can be applied to the top of a jar or to a door knob, with different consequences. This comprises two cognitive steps: deciding what is needed to perform a procedure, *i.e.*, what will constitute the object(s) of the procedure; and realizing that such object(s) can be different from what was originally planned (generalization).

Activity guidelines

In programming (but also in non-programming activities), rather than define a procedure by focusing the activity just on "what to do", one can also focus it on "what is needed".

As an example, one can consider the case mentioned in the previous section where a robot would have to be taught how to put a name on the roof of the house. Instead of teaching a robot to pick up text pads, write "João", go to the notebook, pick up the roof sensor, and drop the name "João" on the notebook, one could conceive a robot that would place on the roof any name it was given. But there is a large cognitive leap from teaching a robot how to do a specific thing (write "João" and put it on the roof) to teaching it about "receiving" parameters.

That leap can be eased by presenting the activity in a different manner: "what do you want to do?"; "– to put my name on the roof."; "what do you need?"; "– I need my name and the roof". This approach provides a motive for the existence of parameters even in the simplest cases.

Figure 202 – Assembling "what is needed", before teaching a robot

The use of generalization, in ToonTalk, is achieved by removing detail about the parameters from the robot's thought bubble, as described in section 3.3.5 (Figure 189, p. 209). The experience from working with preschool/kindergarten children (described in section 5.2) points to the possibility of presenting this by producing several similar robots (for instance, one to write "João", another to write "Maria", another to write "Miguel", etc.) and present generalization to the children as a matter of making the robots less "picky".

3.4.8. Type checking

Computer programming concept	Preschool education connection
A type is "*a collection of computational entities that share some common property*" (Mitchell, 2003, p. 129). In computer programming, entities have a type, such as "integer number", "floating-point number", "list", "atom", "number between 1 and 10", etc. Entities must be used consistently with their types. The programming languages may or may not enforce this, but inconsistent use of types usually results in program errors. ToonTalk enforces types, and wrong matching of types, when detected, causes the program to stop or pause.	Types are a genre of hierarchical classification: number "1" is also "a number"; the picture of a ball is also "a picture", and both are "an object". In this sense, matching a value to its type is a hierarchical matching of patterns. Activities which acknowledge types can be approached from the point of view of type-definition (which in ToonTalk is done by generalization of strict values), or from the opposing point of view of type-matching.

Activity guidelines

As mentioned in section 3.3.5 (Figure 189, p. 209), robots are generalized by "erasing" or "vacuuming" the constraints in their thought bubbles. Vacuuming is the simplest yet more general approach: by vacuuming a constraint it no longer restricts anything; erasing is more complex, in that only some bit of information is generalized. Figure 203 presents an example of this.

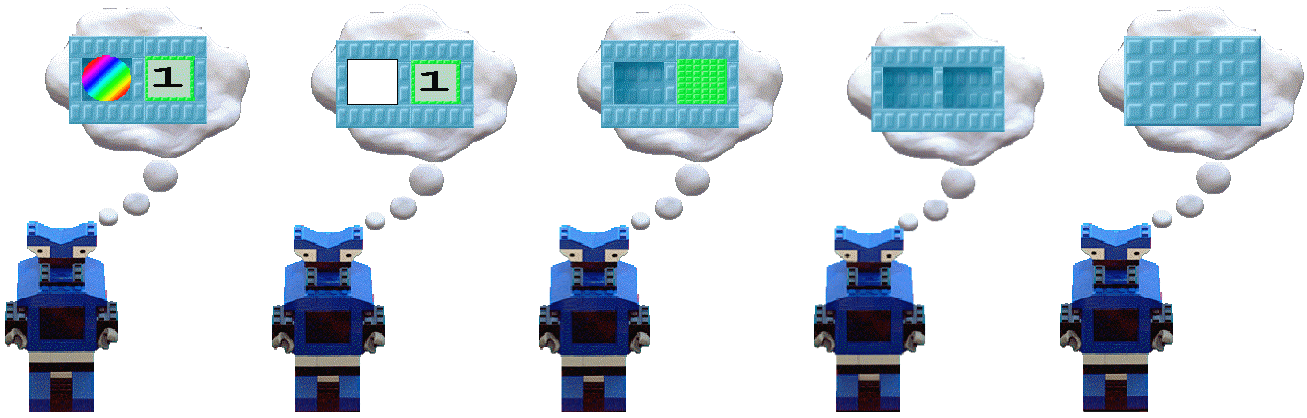


Figure 203 – Successive generalizations of the original types.

From the left, the robots demand: a two-hole box with a picture of a ball and a number pad with the number 1; a two-hole box with a picture and a number pad with the number 1; a two-hole box with anything (or nothing) in the left hole, and a number pad in the other hole; a two-hole box with anything (or nothing) in either hole; a box with any number of holes, regardless of their contents.

As mentioned in the previous section, the activities I conducted with children aged 4 and 5 years old (described in section 5.2) point to the feasibility of presenting generalization techniques to children at this age level, by producing several similar robots, under the same goal, and present generalization to the children as a matter of making the robots less "picky".

Pictures also provide nice possibilities for more complex type-matching activities. However, since ToonTalk has no provision for conducting such complex picture-matching at the level of the robot constraints, such activities have to be conducted with the support of specific user programs.

Examples imagined and developed by my college students and myself include: recycling containers that will accept only "paper", "glass", etc.; farm animals that enlarge if fed with adequate feed; a child picture that fattens when fed with candy and grows taller and stronger when fed with healthy food; etc.

3.4.9. Higher-order procedures

Computer programming concept	Preschool education connection
A procedure (or function) is considered first-order when it works directly on arguments which are not functions and higher-order when it combines the operation of other procedures or functions (Abelson, 1996, pp. 56-60; Mitchell, 2003, pp. 34-35).	The definition of procedures (*vd*. section 3.4.6) requires one level of detachment from the actual operations being performed on objects. By using procedures from within other procedures it is necessary to go further another detachment, one where sub-procedures are used not to attain immediate results, but in view of their expected actions. This is associated with a problem-solving technique that is often useful, albeit not readily for all thinking styles (as put forward by Sherry Turkle and Seymour Papert, in 1990): **black-boxing**, the thinking about the general reactions and impact of a small problem or solution, not bothering with its internal implementation.

Activity guidelines

In ToonTalk, there are two different cases of usage of higher-order functions: when a robot launches other robots, and when several "picture behaviors" are placed behind another picture, to produce a more complex behavior. The latter usage was extensively developed and used with 6-8 year-old children in the European project Playground (Playground Project, 2001).

The former usage was successfully attempted in the field work supporting this thesis (*vd*. section 5.2), by combining robots programmed by the children themselves, and used in progression (*i.e.*, children programmed and used one robot at a time, I did not attempt to suggest to them the programming of several robots at once, without individual uses for each).

For instance, children programmed robots to put their name on the roof of a new house, as mentioned in section 3.4.7. They had previously programmed and used robots to build new houses. A welcomed suggestion of mine was "*if you teach the house-builder to make houses sending your name-writer, you'll manage to fill the cities with houses bearing your name, to tell them apart.*"

Since the name-writing robot was stored in the notebook, this usage of a robot from within a robot was achieved by simply teaching the house-builder to get the robot for new houses from its location in the notebook, rather than using a new, untaught robot from the toolbox[223] (Figure 204).

Figure 204 – Launching a new house with a different robot

[223] This occurred in a programming session at the early stages of the field work; it violates the preferred programming style developed subsequently.

3.4.10. Parallel/Concurrent execution

Computer programming concept	Preschool education connection
Several programs, a.k.a. "processes", can be executed seemingly at the same time. The programming style and background for this concurrent/parallel model was presented in section 3.2.1.	A problem or goal doesn't have to be analyzed as a sequence of steps. Rather, just like several children can cooperate, working concurrently towards a common goal, different parts of a problem can be considered independently.

Activity guidelines

Activities for children can consider the existence of multiple parallel events or actions in view. For instance, while a child conducts his/her activities, robots programmed by other children can be visible running; or various automated objects can be used at the same time, rather than planning for only one to work at a time. And, although ToonTalk doesn't currently support it[224], in a multi-user programming environment the actions of other programmers can cause changes to occur as one child navigates through his/her programming environment.

Another option is to consider that the parallel events or actions are taking place away from view. For instance, ToonTalk houses can be customized as farm buildings, such as a stable, a pigsty, etc. Various children can tend the animals in different houses; collect fruits, harvest fields, etc.

In a multi-user environment, the changes would be concurrent; in the current single-user ToonTalk environment, changes take place sequentially. However, since the amount of consecutive time that preschool aged children dedicate to computer activities is fairly limited (more on this in section 5.2), every time a child plays in this "shared farm" it seems as if various events are taking place concurrently.

A particular event that occurred with a 3½ year-old girl, during my 2004 field work, helps shed some light on these statements. She played with several automated images, whose animations could be turned on and off, using the space bar and the "." key (Figure 205 – the top-right and bottom-left pictures are turned off, the others are in motion). One week later, she resumed her ToonTalk play, from the same point. After ToonTalk launched, displaying some of those images working as she left them, she immediately exclaimed in surprise: "*they're still working!?!*"

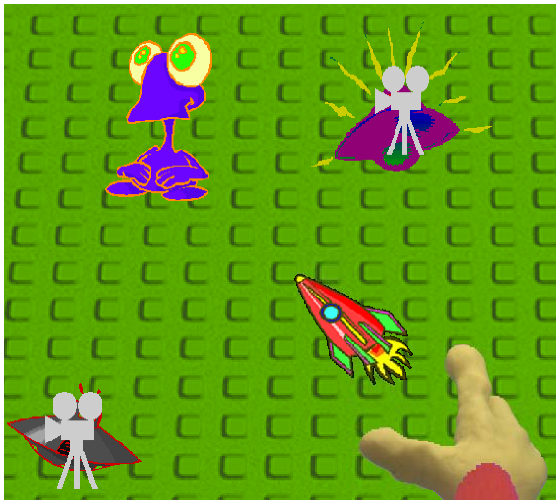


Figure 205 – Assorted animating and stopped pictures.

[224] ToonTalk users at several computers can communicate by sharing birds, and this can be used to produce objects that change according to remote operations. However, this method offers some amount of complexity and is not as general has having access to a truly multi-user programming environment, such as MOOSE Crossing (p. 172) or Pet Park (p. 182).

3.4.11. Parallel/Concurrent statements

Computer programming concept	Preschool education connection
A parallel statement[225] is composed of several other statements, which are launched simultaneously, when the parallel statement is launched. The parallel statement ends when all sub-statements also end (Hoare, 1971).	Parallel/concurrent execution can be initiated as in section 3.4.10, by starting one action at a time. But a child can also launch several parallel actions (or cause them to occur) simultaneously.

Activity guidelines

In ToonTalk, there are three kinds of situations where this technique can be involved in activities for children:

1. Teaching a robot to launch several trucks with other robots.

   As described in section 3.4.9, "Higher-order procedures", a child can teach a robot to use another robot. If, however, the robot is taught to use several robots, then the actions of all those robots will all be launched as a consequence of a single action. Considering the example about roof-naming, used in that section, the child could teach the robot with a box of pictures, instead of names, and use this technique to produce several houses at once, each with a different picture on the roof (an example with Euro coins is presented in Figure 206).

2. Copying a bird's nest to cause delivery of data to more than one location at once.

   A bird/nest pair (*vd*. section 3.4.12) can be used for communication of data. If the nest is copied, then the bird multiplies itself to deliver copies of data to different locations. This can be used to cause instructions or parcels, food, etc. to appear simultaneously at different locations (which, in the scope of the preschool activity, can imply immediate consequences).

   As an example, a Santa Claus role-play activity could include the delivery of "Christmas gifts" to several houses, by placing copies of the nest of Santa's bird in each house.

3. Playing in a setting with various automated elements.

   An educator can program several elements to respond to the same external change. For instance, sheep can be programmed to avoid a shepherd dog, and children can explore the consequences of that behavior on several sheep at the same time, by moving the dog.

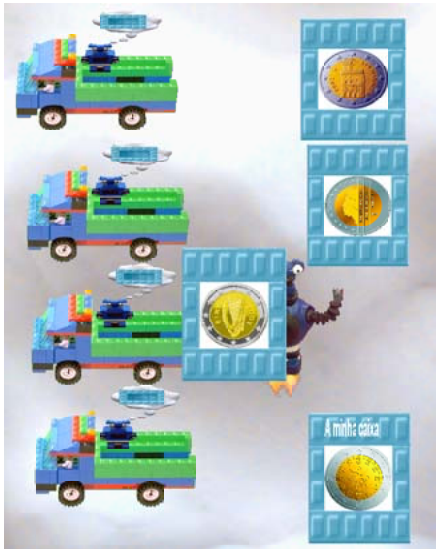


Figure 206 – Teaching a robot to create several houses, each with a different Euro coin.

[225] The concept of "statement" was presented in section 3.4.4.

3.4.12. Message passing / Communication channels

Computer programming concept	Preschool education connection
"(…) *parallel processes must be able to cooperate in the sense that they can activate one another and exchange information*" (Hansen, 1969, p. 158). "*Two parallel processes can cooperate by sending messages to each other*" (*id.*, p. 162). Messages sent to a process are queued, usually on the rule of "first-come, first-served". Messages can also be passed between one process and several others, in what is called a multicast, or 1-to-N communication. If a process receives messages from several processes in the same queue, it is called a concentration, or N-to-1 communication. Finally, if both situations are combined, it is called multipoint or N-to-N communication (Diot, 1994).	Message-sending can be used to exchange information, as technically described on the left. Exchanges of information are common in childhood activities, involving mathematics or social relationships. However, in a metaphorical world such as ToonTalk, this can also be used to perform a less obvious variant, namely the transportation of objects between locations. In ToonTalk, message communication is achieved by using birds, which act in a way similar to carrier pigeons (see rows about birds, in Table 18, on page 208). A bird can be used to carry an object to its nest, but nests can be joined together, so that several birds share the same nest (N-to-1 communication). Nests can also be copied, so that a bird delivers several copies of an object at the same time, one in each copy of the nest (1-to-N communication).

Activity guidelines

All activities that make use of birds to carry things involve message passing. At the simplest level, this can be something as simple as offering gifts: two children playing in the same ToonTalk session, each with his/her own bird and nest. By exchanging nests, the children can use the birds to send "gifts" to each other (such an activity was performed with children aged 4 and 5, during the field activities described in section 5.2). This can initially occur in plain sight (all birds and nests in view), but made more complex, by having a different house for each children.

This technique could be used in various ways: a postman that can use the birds to deliver letters and parcels; a baker that can use birds to deliver bread and cakes, as well as receive orders (a complex activity based on this is described in section 7.3.3).

If the nests are in robots' boxes, the arrival of an object carried by a bird can allow robots to perform as they were taught. This could be used to create reactions to the arrival of messages (such as playing sounds, placing images on the wall, etc.), but also to respond. If the receiving robots have a bird, and the sender has another bird, the robot can "reply" to the message (Figure 207). This has been used, for instance, to make two robots "toss a ball around", but also by a 3-year old to "play ball" with a robot that would send it back (these cases can be found in sections 7.3.4 and 7.3.5).

Figure 207 – Sending a ball for the robot to return it.

3.4.13. Speed independence

Computer programming concept	Preschool education connection
Nothing can be assumed regarding the speed of execution of concurrent processes. Identical processes do not ensure identical run speeds. "(…) *processes* (…) *are to be regarded as completely independent of each other. In particular, we disallow any assumption about the relative speeds of the different processes*" (Dijkstra, 1965, p. 74).	Part of the construction of procedural knowledge (Papert, 1980, p. 223) can include reflections on how the speed at which different procedures are executed may or may not impact the equivalence of the procedures themselves. Particularly, two procedures are identical, even if they are executed at different speeds (*e.g.*, opening a door: hardly will it ever be done at the same speed, even if by the same person).

Activity guidelines

In ToonTalk, a task being performed in different houses (or in the same house, by different robots) isn't necessarily run at the same pace. Activities that can make use of this notion include all where one has to analyze the sequence of results from robots. For instance, suppose two robots are working in a "store" (a customized house): one can respond to requests by providing products, another can respond by providing paper-wrapping for them. If both robots respond to the same bird or event, one cannot be sure of what will arrive first, the product or the wrapping.

In Figure 208, a group of boxes is delivered (using a bird) to two robots at the same time. However, as that figure shows, one of the robots finishes noticeably first. But doing this several times produces different "winners": the same procedure is executed at different speeds.

Another way of working with this is by taking advantage of time differences due to elements involved in the task: for instance, adding to a large number is slower than adding to a small number; joining two large pictures is slower than joining two small ones; adding a box to a small array is faster than adding it to a large array, etc. (although these differences may be too small to notice individually, they can add up and produce visible biases). A different approach to help children realize that a procedure can be executed at different speeds is to employ the different speeds of execution of a task by the same robot: slow when working in plain sight, faster when working while the programmer is standing up, fastest when working out of sight (behind a picture or in a different house).

A secondary line of action involves the impact of "irrelevant" actions: for instance, dropping and picking up again a box while doing a procedure wastes time, but doesn't change the outcome. However, it can change the speed. For instance, in order to teach a robot to write the name "MARIA", one might teach the robot to pick up a pad for each letter and join all the pads together (such a robot is presented in Annex II, p. 441). But one could also teach the robot to pick a single pad and type "MARIA" at once using the keyboard, and this second approach would be faster, although producing the same result.

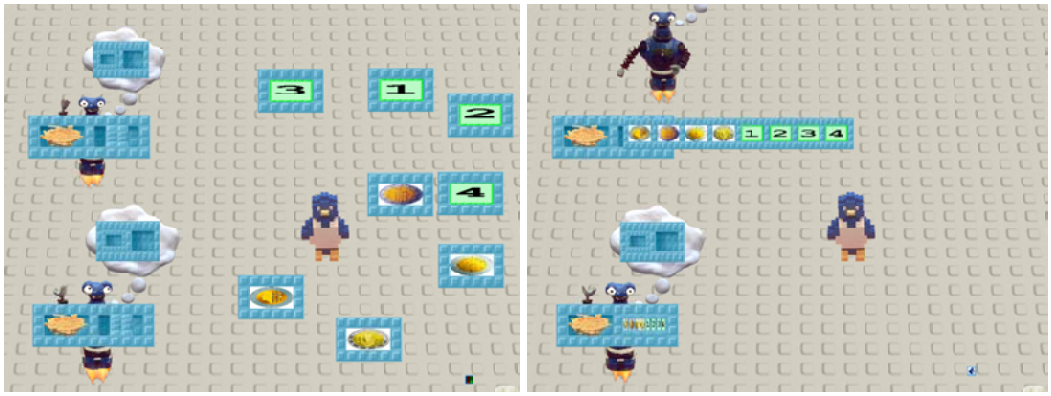


Figure 208 – Different execution speeds: the bottom robot finished first, this time.

3.4.14. Synchronous/asynchronous communication

Computer programming concept	Preschool education connection
Communication (*vd.* section 3.4.12), can be synchronous or asynchronous. The first variety includes all cases where, a process, say "A", only sends a message if the receiver, say "B", is ready to receive it – "*there is no automatic buffering: in general, an input or output command is delayed until the other process is ready with the corresponding output or input. Such delay is invisible to the delayed process*" (Hoare, 1978, p. 415). If a communication channel doesn't fulfill this requirement, then communication is said to be asynchronous. Synchronicity can be achieved either by forceful control of the processes by the system (*id.*, *ibid.*) or by requiring communication to be preceded by the exchange of "tokens" (Garg, 1988).	As everyday examples of this distinction, one can consider a child talking to another; communication is synchronous, because if the second child isn't listening, the first cannot speak. If, however, the first child sends a letter or an e-mail instead of talking (or phones and gets an answering machine), communication is asynchronous, because this first child can proceed with sending more letters/e-mails or phoning the answering machine several times, whether or not the second child is ready to receive the letters/e-mails or listens to the messages in the answering machine. In activities involving communication, the actions of sending and receiving can be analyzed, to decide if the sender can send something or not, without first checking to see if the receiver is ready. In such cases, the usage of tokens (physical or virtual) may be a helpful technique.

Activity guidelines

ToonTalk provides means for communication, in the form of birds. However, birds are asynchronous, because one can always give more than one message to a bird: it will simply pile them all up in its nest, whether or not those messages are being received by "someone". However, synchronous communication can be achieved using birds, by exchanging tokens.

For instance, considering the ball-playing robot of section 3.4.12 (Figure 207), it is not sending several balls: it is sending its only ball; in order to continue, it must receive it back. In this case, the ball functions as a token for enabling communication. Figure 209 shows an expansion of this to work with two robots: notice that the birds and nests are "crossed": the left-side robot has nest A and bird B, and the right-side robot has nest B and bird A.

Figure 209 – Synchronous communications with a token (ball)

Synchronicity may be required due to specificities of the activity. For instance, suppose a child is at a "supermarket" (a customized house) and sends products to another house, using birds. If those products include ice cream, yogurt, or frozen products, it is necessary for a robot at that house to be ready to store them in the refrigerator, lest they get ruined. In such a situation, there is a motive for caring about synchronicity between sending and receiving: it is imposed by the context.

3.4.15. Recursion

Computer programming concept	Preschool education connection

Recursion can be understood as a definition that is, at least partly, defined in terms of itself: "*In general* (…) *the routine for a recursive function uses itself as a subroutine*" (McCarthy, 1960, p. 193). A common example of recursion is the definition of the mathematical factorial function (n!) as:
 	
n! = {if n=0, result = 1; else, result = n × (n-1)! }
 	
Recursion can also be employed in speech, as in "*to clean a house, starting from the main door, you clean the rooms you can reach through the inner doors, and finally the entry hall itself; to clean those other rooms, you do the same: clean yet other rooms you can reach through room doors, before cleaning each room itself*". Expressions such as "*and so on*", or "*over and over again*" are commonly employed in such uses of recursion.
 	
In fact, recursion has been tied to the ability of the limited-element set of human languages to generate an infinite number of sentences, by being essential to grammar:
 	
« *It seems clear that we must regard linguistic competence as* (…) *a system constituted by rules that interact to determine the form and intrinsic meaning of a potentially infinite number of sentences.* (…) *Such a system – a generative grammar –* (…) *defines a language* (…) *as "**a recursively generated system**, where the laws of generation are fixed and invariant, but the scope and the specific manner in which they are applied remain entirely unspecified."* »
 	
(Chomsky, 1968, my emphasis)
	The use of recursion is associated with the ability to recognize, use, and define patterns and models: "(…) c*ontemporary thought views learning as a person's ability to construct new knowledge based upon what they already know or believe to be true* (…); *in short, the ability to perform model-based reasoning based upon reflection (**actually 'recursion'**), and metacognition"* (Reilly, 2002, my emphasis).
	
	The use of recursion in preschool and kindergarten is initiated with the description and application of simple patterns: "*In the lower grades, students can describe patterns like 2, 4, 6, 8, ... by focusing on how a term is obtained from the previous number—in this example, by adding 2. This is the beginning of recursive thinking*" (NCTM, 2000, ch. 3, p. 38).
	
	But for these age groups, it is also important to employ recursion in non-numerical ways. A well-known example is this imaginary program, Logo-like (Muller, 1998, p. 341):
	
	`TO GET.THROUGH.LIFE`
	`  GET.THROUGH.TODAY`
	`  GET.THROUGH.LIFE`
	`END`
	
	Disregarding the use of text, an analysis of actions using recursion, such as this, is at a complexity level usable at the preschool level.
	
	In ToonTalk, a robot can receive a copy of itself in its box (or get a copy of itself from the main notebook, or use ToonTalk's magic wand to copy itself). A recursive operation is done by sending one or more such copies in a truck (a recursive spawning, in this case).

Activity guidelines	
During my field activities, recursion was not a key topic, and therefore I cannot provide elaborate field-based examples, just the following simple field-based proposals. In these cases, recursion was employed, albeit not pursued, to create new houses. When teaching a robot how to build houses, it must put another robot in a truck (Figure 204 and Figure 206). Recursion is employed if that robot is a copy of the current robot. This means that the new robot is also a house-builder. The resulting program behavior depends on what the actual robots do, besides building houses. For instance: ▪ if the robots only build houses at each turn, this method is faster at "filling up" the city, because at each turn there are more robots building houses at the same time; ▪ if each robot builds a house and blows up a bomb, the result will be a "moving house".	

3.4.16. Guards and alternative commands

Computer programming concept	Preschool education connection
A command may have generic guards, determining its execution: the command is executed "*only if and when the execution of its guard does not fail*" (Hoare, 1978, p. 422). But a guarded command may do more than simply run or block: it can include alternatives for when a guard fails. "*An alternative command specifies execution of exactly one of its constituent guarded commands.* (…) *if all guards fail, the alternative command fails. Otherwise, an arbitrary*[226] *one with successfully executable guard is selected and executed*" (*id.*, *ibid.*).	The definition of alternative commands provides a setting similar to that of conditional expressions, which were mentioned in section 3.4.5. However, guards provide a way to reason about a problem from a different perspective. Using guards, the focus is on what alternative actions can or must be considered, and the conditions regulating each. Rather than analyzing a condition and considering what to do for each possible value, one must consider what possible courses of action are required, and under what conditions each should be executed.

Activity guidelines

In ToonTalk, guards may be seen as the robot's thought bubble constraints (*vd.* section 3.4.8), and alternatives can be seen has the members of a robot team. Thus, activities using only simple guards match those mentioned in sections 3.4.7, "Parameter passing", and 3.4.8, "Type checking".

Defining alternatives in ToonTalk by creating a robot team involves:

- deciding which actions need to be performed (a robot for each);
- deciding under what conditions they need to be performed;
- deciding the priority (*i.e.*, the desired sequence) for checking those conditions.

At the simplest level, such an activity is identical to a `case` expression (*vd.* section 3.4.5). For instance, a robot may know how to put "DAVID" on the roof of a house, when given a "D"; and another may know how to do the same thing with "JULIAN", when given a "J". By dropping one on another, the resulting team is able to respond to both "D" and "J" – the result of teamwork.

A further level involves having a generalized robot[227] that doesn't care about the letter: it simply puts it on the house roof. By dropping this robot on the previous team, one can now provide "D" to get "DAVID", "J" to get "JULIAN", and any other letter[228] will end up on the roof (this is equivalent to the use of the keyword `else`, mentioned in section 3.4.5).

Finally, the highest level of complexity is achievable if one has more than one generalization in play. This can occur when there are several holes in a box, and robots care about some holes, but not about others. But even the previous example can provide a case of this. If one wants the robots to try and put pictures, letters, and numbers on the roof, but nothing else, one needs three generalized robots: one that cares about any letter pad (erased letter pad in its thought bubble); one that cares about any number (erased number pad); and one that cares about any image (erased image). In this case, their sequence is not important, as long as they are behind the "DAVID" and "JULIAN" robots, but in other cases, the ordering could provide interesting cases to analyze and reason on why different orderings produce different results.

[226] Here, Tony Hoare is referring to a generic concurrent-programming system. But he does acknowledge that arbitrariness is not absolutely crucial to the use of alternatives: "*An implementation should take advantage of its freedom of selection to ensure efficient execution and good response*" (Hoare, 1978, p. 422). ToonTalk's option is to prioritize all alternatives (order of robots in a team).

[227] By erasing the letter pad in its thought bubble ("any letter") or vacuuming it ("anything").

[228] If the generalization was done by vacuuming the letter pad from the robot's thought bubble, the robot will in fact attempt to put any object on the roof, although only letters, numbers, and pictures work.

3.4.17. Input guards

Computer programming concept	Preschool education connection
This particular kind of guard provides a way to support synchronized communication between processes. "*A guarded command with an input guard is selected for execution only if and when the source named in the input command is ready to execute the corresponding output command*" (Hoare, 1978, p. 415). In ToonTalk, input guards can be seen as nests in the box given to a robot. The robot simply waits until something arrives on a nest, and only then will it compare the constraints with the contents (see if the guard holds) and proceed with execution.	Using an input guard implies that children need to define two actions (robots), and link them together (using birds). To do so, they must be able to keep in mind the actual actions of both robots, and devise a plan for flow of information: "*this one will give this to the bird, which will take it to that one, which will do X.*" In other words, children must go beyond thinking about communication, and start thinking about the necessity of coordinating communications. This can also be used for including conservationism (existence and operation of objects that are no longer in sight) in activities: one of the robots can be working in a house (or in another corner of the current house), out of sight.

Activity guidelines

The simplest level occurs when a robot is simply waiting for something to arrive in its nest, and the sender is the child herself (*i.e.*, the child, not another robot, gives the missing object to the bird). This level is represented in examples such as those mentioned in the section on communication, 3.4.12. Other simples examples, not requiring the robot to communicate back, are:

- a robot that vacuums the contents of its box; if one places a nest in that box, from then on, the robot only works when something reaches the nest;
- an "interior decorator" robot (also used with a nest in its box), which places on the room wall any picture it gets.

At a more complex level, a robot sends an object to another, which then acts upon receival. An example of this is provided in section 3.4.14: a robot sending a ball to another robot, which then replies by sending the ball back. This example, once working, could be changed by moving one robot away from the visible area, or even to another house. And yet, the two robots would still be able to "play ball". But there is no need for a robot to respond: for instance, a robot working in a warehouse could be dispatching Christmas decorations for robots in other houses; those robots, upon receiving them, could put them on the roofs. By traveling around in the helicopters, the child could see which houses had robots working in them that had already received the lights, and which houses either had no robot there to put the lights on the roof or had not received them.

Conversely, a robot may be at work, dispatching feed to rabbits, being bred in another house (decorated as a rabbit hutch). That feed would be accumulating, even if the nest where it is arriving is placed behind the picture of a rabbit. But a robot could be taught how to vacuum that feed, as if the rabbit had "eaten it" (see Figure 210). Better still, that robot could do something after vacuuming the feed, such as playing a munching sound, or making the rabbit fatter!

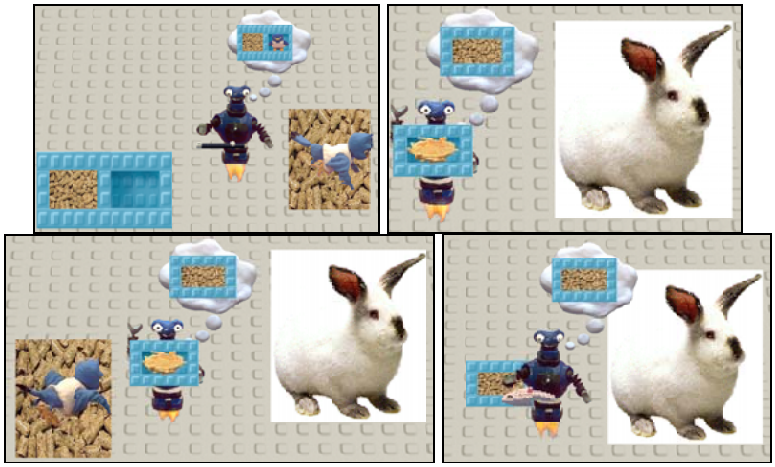


Figure 210 – Feeding a rabbit

3.4.18. Clients and servers

Computer programming concept	Preschool education connection
An important concurrent programming style involves the use of client/server relationships between processes. "*Some server processes render a service to some client processes. A client can request that a service be performed by sending a message to one of these servers. A server repeatedly receives a request for service from a client, performs that service, and (if necessary) returns a completion message to that client*" (Andrews & Schneider, 1983, p. 26). In ToonTalk, client and server processes can communicate using birds. These can be fixed at the time of programming (*i.e.*, a server always responds to the same clients) or passed along from the client to the server at program runtime, for dynamic client/server interaction (Figure 211).  **Figure 211 – Bird carrying a request: data + reply bird**	When modeling real-world interactions with a computer program, one finds multiple opportunities for employing the client/server model (shopping and parcel deliveries, for instance). A child programming a client process to make requests does not require knowledge of **how** they will be answered. She just needs to worry about **what are** the answers that may arrive (and in fact, whether they do at all or not). This requires the child to reason over the expected behavior of non-local, non-present entities (since the server process usually operates out of sight). Also, there is ample space for activities related to memory use. For instance, what if an answer does not arrive? The child must remember that a request was done earlier, and that an answer is missing. What if it does arrive? She needs to focus and associate the answer with the request made earlier, and be able to check if the answer matches what was intended in the request (besides memory, this is also an opportunity for using record-keeping).

Activity guidelines

A crucial aspect of such activities is how to identify the client making a request, and the server to whom the request is being made. In preschool settings, this can be made with text (capitalized names), but the most versatile solutions would be the use of pictures and/or sounds. For instance, Figure 212 displays several "requests", in the form of ToonTalk boxes. In the rightmost hole of each, a picture identifies the "requester", represented by his/her "hat" (nurse, nurse, baker, and postman). These could also be sounds for the child to play, achieving the same effect.

Another aspect is how to identify the request. Again, the use of pictures and sounds provides more flexibility, rather than depending solely on text or numbers (yet, these can also be used). For instance, in the same figure, the top request comes from the nurse, and it consists of a parcel (first hole) and a postage stamp (second hole). The text in the third hole simply restates it in written form.

Numbers could be used to represent how many boxes and stamps are being requested, so that the stamp picture could stand for as many stamps as desired.

A child can execute "server" actions, responding to requests, or program a robot to respond, or to produce the answer. The answers can be delivered to the requesters manually, but the requests can also include a reply bird, as shown in Figure 211 (there can be an extra hole in the request box, for this reply bird, but it can also be behind one of the pictures). Using such "reply birds", server robots or children can answer requests from clients not known beforehand.

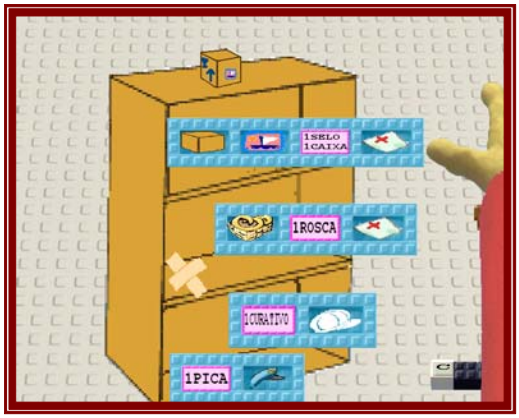


Figure 212 – Sorting requests

4. Introduction to preschool education and computers

4.1. Early childhood education philosophy(ies)

4.1.1. Brief history of early childhood education

The current educational practice with young children (less than 6 years old) is strongly based on educational and pedagogical research, particularly that focusing on children's cognitive, physical, and social development, as is further described in the coming sections. Section 4.1.2 is devoted to presenting the major basic theories that currently impact the practice and research in the education of young children, and section 4.1.3 presents some of the major pedagogic models based on those theories, which provide the guidelines for the actual practice in the field.

This current status of the practice is however a relatively novel situation regarding young children's education. Raising and educating children has been a task of humankind from our very onset, and although there are no written records prior to the civilizations of antiquity, much can be gleamed by anthropological research among modern-day practices of indigenous peoples throughout the world (*e.g.*, Briggs, 1998, details the high level of complexity and involvement in the emotional and ethical education of Inuit children aged between 2 and 4, through what might be considered a Socratic questioning method[229]). And formal education for older children (aged 6, 7 and up) at schools and/or with specialized tutors has existed since antiquity, in developed societies such as those in early China (Monroe, 1907; Cartier, n.d.), Egypt (Brunner, 1981), Greece & Rome (Monroe, 1907; Pietri, 1981; Lascarides & Hinitz, 2000; Mota & Cruz, 2001; Mota, 2003), India (Monroe, 1907; Misra, n.d.), Japan (Shoji, n.d.), and Mesopotamia (Lucas, n.d.). But although some of these civilizations recognized young children (with less than 6 or 7 years of age) as an age-group with remarkably distinct characteristics from those slightly older[230], before the mid-XVIII century there were no formal educational institutions devoted to them.

> *« In antiquity the family was the center of the child's early education. Education proper* (…) *did not begin until the child was seven and was sent to school. Until then, he was "brought up" at home by women, primarily by his mother, and in well-to-do families also by his "nanny." »*
>
> *(Lascarides & Hinitz, 2000, p. 3, referring to education in Ancient Greece)*

Although schools and educational philosophy underwent numerous changes and developments since antiquity, for the most part the education of younger children remained true to this description.

Departure from this *status quo* came gradually, as a result of changes in the goals and views of civilization on the education of older children. It evolved in focus from being a simple matter of providing children with fundamental skills for the practice of a profession or adequate social conduct, until modern times, when it became an endeavor focused on the development of the individual.

> *«* (…) *the conscience of the child as distinctive and different from an adult was not generally a common concept until the XVII century.* (…) *The child would quickly pass from the period of childhood to that of adulthood. The high rates of infant mortality, along with the short life expectancy of adults, were undoubtedly to blame for this prompt integration of the child in the world of adults.* (...) *»*
>
> *(Santomé, 1991, p.10, translated from the Portuguese version)*

[229] A description of the Socratic Method can be found in Windelband, 1919.

[230] *E.g.*, the Spartan culture acknowledged development stages and considered years from birth to 7 as those prior to formal education (Lascarides & Hinitz, 2000, p. 5); and Hippocrates "*divided man's life into seven stages*" (*id.*, p. 8), the first of which was "*infant (birth to seven years)*" (*id.*, *ibid.*).

This state of affairs began to change in the XVII century, due to the social changes brought upon by the Industrial Revolution. Child labor was widespread, but the operation of mills and other industries by adults was not compatible with the presence of children requiring the care and attention of their mothers. Also during this epoch, the new philosophical ideas of the Enlightenment started to take ground, although their impact on general educational practice was slow and mostly felt only much later, well into the XIX century (Abbagnano & Visalberghi, 1959, vol. III, p. 492).

> *« It is from the XVIII century onward that slowly, alongside the first signs of the Industrial Revolution and the increase in mortal age, that it will be sought to separate children from adults, something that will be evaluated for scientific and ethical reasons* ***a posteriori****. Philosophers start to support the idea of an innocent childhood, whose corruption had to be avoided, and due to that requiring special education and protection. »*
>
> *(Santomé, 1991, p.11, translated from the Portuguese version)*

This change in the role of children in society and the in role of their education eventually led to widespread schooling, rather than just for a fringe of the population; school gradually ceased to be a professional academy, and started evolving into a service to the benefit of all citizens.

> *«* [In the XVIII century] *there was no public measure in England, either national or local, regarding elementary education. It was the care of some parish schools, sometimes; in other cases women without any specific training would welcome in their homes, for a fee, some of the neighborhood children (Dame Schools); in yet other instances, finally, middle schools would organize small preparatory schools. Beyond this, for the poor, some pious societies emerge, with philanthropic and religious edification goals, whose schools and Sunday courses function as charity, with scant dissemination and effectiveness. In France, the situation is certainly not better. »*
>
> *(Abbagnano & Visalberghi, 1959, vol. III, p. 489)*

Young children would also be taken care of in institutions such as those mentioned above. Sometimes, it would also happen for children aged three, four, or five to attend classes with older children. For example, during the XVII and XVIII centuries, in the Puritan schools of the English colonies in North America, "*very young children were sometimes enrolled in grammar schools, for they were seen as able of high levels of intellectual achievement. They often learned to read by the age of three or four, and started to be taught Latin by the age of five or six*[231]" (Spodek & Saracho, 1993, p. 43).

It is from this context that the case of the "Knitting School" emerges. It was founded in 1769 by the French priest Johann Friedrich Oberlin[232], in Ban-de-la-Roche (Alsace, France), commonly referenced (*e.g.*, Gomes, 1977, p. 14; Spodek & Brown, 1993, pp. 15-16) as the first institution with an educational program specifically designed for young children. As young as 2, children would gather round a "teacher" that would instruct them while she knitted. Unlike other schools at the time, which focused on older children, Oberlin's school included games, handicrafts, and the learning of History and Nature, which was done verbally, but used images for support.

Figure 213 – J. F. Oberlin 1740-1826
From: http://www.sdv.fr/judaisme/histoire/document/oberlin/pict/8ober.jpg

Similar schools opened in five nearby villages before Oberlin's death in 1826, but this methodology was never adopted elsewhere. "*The French*

[231] Retroversion into English of the Portuguese translation.
[232] "*Johann Friedrich was his recorded name at birth. The French refer to him as Jean Frédéric and anglicized, his name often appears as John Frederick*" (Oberlin, 2004). Note: Alsace had only become part of France in 1648.

Revolution and the anticlerical attitude of the emerging nation were probably some of the reasons for this lack of influence" (Spodek & Brown, 1993, p. 16).

The schools of Oberlin nevertheless mark a turning point in the history of early childhood education. Until then, the education of young children was essentially done within the family and other informal environments. Oberlin's schools represent the beginning of a different historical period in early childhood education, characterized by the attendance of extrafamilial preschools.

Overviewing the history of early childhood education, one can divide it in several distinct periods that overlap in actual field practice, but nevertheless provide a picture of its evolution:

1. Early history, prior to the first extrafamilial preschools.
2. Enrolling of young children in schools and/or other extrafamilial institutions.
3. Early preschools and kindergartens, created with philosophical and social concerns[233].
4. Creation of widespread public preschools, and their evolution until modern day, strongly influenced by developmental psychology and other scientific research.

The first two items of this overview, on which I have so far elaborated, constitute the background of modern culture and view on young children. But the two latter items provide the link between the first efforts and modern-day education of young children. Oberlin's schools were the first link in this chain, but one without widespread influence, as mentioned above.

Figure 214 – Robert Owen 1771-1858
From: http://www.cottontimes.co.uk/cottonpix/Owen%20Rbt%201845.jpg

It was by the town of New Lanark, in Scotland, that appeared the first example of a formal institution for early childhood education which reached wide impact and adoption. It was created by the Welsh entrepreneur Robert Owen: a school for the children of the workers of the spinning mill he managed at that town (Spodek & Saracho, 1993, pp. 44-45; Gordon, 1994; Beatty, 1998, pp. 6-7; Donnachie, 2003). Owen was a social reformist, who explicitly wanted to improve social conditions for workers and achieve the re-ordering of human society. Following his views, presented in a series of essays with the telling title "A New View of Society" (Owen, 1812), he created the Institute for the Formation of Character, where children younger than 10 would be educated during the day, and older children and adults would attend night classes.

> *« (…) Any general character, from the best to the worst, from the most ignorant to the most enlightened, may be given to any community, even to the world at large, by the application of proper means (…) every day will make it more and more evident that the character of man is, without a single exception, always formed for him; (…) an education for the untaught and illtaught becomes of the first importance to the welfare of society. (…) children (…) will afford all the opportunity that can be desired to create, cultivate, and establish, those habits and sentiments which tend to the welfare of the individual and of the community. »*
>
> *(Owen, 1812)*

The ground floor of the Institute was devoted to the infant school, which opened in 1816 (Gordon, 1994, p. 5), with separate rooms for children aged 1 to 3 and 4 to 6 (Donnachie, 2003). The education of such young children was at the core of Owen's principles, as can be seen from the quote that follows.

[233] Including Oberlin's innovative Knitting School.

« It must be evident to those who have been in the practice of observing children with attention, that much of good or evil is taught to or acquired by a child at a very early period of its life; that much of temper or disposition is correctly or incorrectly formed before he attains his second year and that many durable impressions are made at the termination of the first twelve or even six months of his existence. The children, therefore, of the uninstructed and ill-instructed, suffer material injury in the formation of their characters during these and the subsequent years of childhood and of youth.

It was to prevent, or as much as possible to counteract, these primary evils, to which the poor and working classes are exposed when infants, that the area became part of the New Institution.

Into this playground the children are to be received as soon as they can freely walk alone; to be superintended by persons instructed to take charge of them. »

(id., ibid.)

The infant school of Owen was based on the principle of play, rather than formal lessons. Another notion that permeated the learning in all age groups, was that "*everything was made relevant for the children, that they should understand what they were learning and why, and that they should enjoy what they were doing*" (Donnachie, 2003). His concerns with the preservation and/or improvement of humankind's nature are in line with the philosophers of the Enlightenment, such as Jean-Jacques Rousseau. His emphasis in the educational use of observation, experience, and direct contact, were most likely inspired in the educational practice and writings of the Swiss educator Johann Heinrich Pestalozzi[234], whom Owen visited in his school in Switzerland (Gordon, 1994, p. 7). But "*probably the answer to the origins of Owen's educational thought is* (…) *that he owed very little to others, arriving at largely similar conclusions with other pioneers by a different road based on his own experience and peculiar philosophy of character*" (*id.*, *ibid.*).

Owen's model gained quite a large degree of success and dissemination. For example, between 1815 and 1825 his Institution in New Lanark received "*20,000 odd visitors*" (Donnachie, 2003) from several parts of Europe. Schools inspired in his model were created in the UK, the USA, France, and other parts of the world. Ultimately, however, Owen's educational philosophy influencing the methods of the infant school was not based on any developmental theory; his antireligious views and the strong link between his schools and social reform, in the form of cooperative communities, following his belief that "*cooperative socialism would simply displace capitalism by example and by education*" (Gordon, 1994, p. 12), along with his political involvement in trade unionism led to government backlash in the UK. In the USA, the infant school movement was also short-lived, due to a combination of several factors, such as "*the attitudes of the public schools system towards the education of young children"* (Spodek & Brown, 1993, pp. 16-17) and *"the radical activities of Bronson Alcott's infant school*[235]" (*id.*, *ibid.*). But although the infant schools of Owenish inspiration didn't directly evolve into modern-day preschools in the English-speaking world, "*Owen's message that training and education must be viewed as intimately connected, is echoed in many educational systems today*" (Gordon, 1994, p. 13). In France,

[234] Pestalozzi (1746-1827) developed an educational method and ideas that anticipated much of modern pedagogy, such as the fundamental value of knowledge based on experience, experimentation, and direct contact with objects; the need to ensure that each child finds meaningfulness what is learned; and the use of self-expression as a way to achieve a deeper intuition of the world. But although he gave great importance to the education of young children, he believed that such education should most adequately be done at home, in the family, rather than in extrafamilial institutions (Abbagnano & Visalberghi, 1959, pp. 589-603).

[235] Alcott was an American educator who attracted controversy towards his teaching after publishing a book called "Conversations with Children on the Gospels", with chapters such as "Conjugal Relations" and "Gestation". The book "*attracted unfavorable attention from Calvinist clergymen and some journalists*. (...) *The public did not forget the controversy* (...)" (Crouch, 1991).

however, several private initiatives were created after the British model of Owen and his followers, and were called *salle d'asile*. These flourished and were gradually taken over by the state. In 1848, a French Law recognized them already as "establishments for public instruction", and called them *écoles maternelles*, although this name would only become commonplace in 1881 (Gomes, 1977, pp. 15-16). Today, France's preschools are still known as "maternal schools" (Ambassade de France au Brésil, 2002).

The second major historical root of modern preschools appeared roughly 20 years after the opening of Owen's Infant School in New Lanark. In 1837, the German educationalist Friedrich Wilhelm August Fröbel created in the city of Blankenburg (Thuringia, Germany) an educational institution for young children, his *Kleinkinderpflegeanstalt* ("institute for infant care"), a name that, at the time, was commonly used in Germany for such institutions. But given that his views on the education of young children were quite different, three years later he renamed it as *Kindergarten* ("children garden"), a name that attempted to summarize those ideas: as plants in a garden, in harmony with nature and under the care of experienced gardeners, so should children be raised (Gomes, 1977, p. 17). His methods and views had a strong and widespread influence in the Western world.

Figure 215 – Friedrich Fröbel 1782-1852
From: http://www.froebelverein-keilhau.de/assets/images/db_images/db_froebel24.jpg

> *« We grant space and time to young plants and animals because we know that, in accordance with the laws that live in them, they will develop properly and grow well; young animals and plants are given rest, and arbitrary interference with their growth is avoided, because it is known that the opposite practice would disturb their pure unfolding and sound development »*
>
> *(Fröbel, 1826)*

Fröbel had studied the pedagogy of Pestalozzi. Like Robert Owen, he visited Pestalozzi's own school in Switzerland but his contact with Pestalozzi was much longer than a mere visit (1805 and 1808-1810). During his early career, between 1816 and 1835, Fröbel founded and run several schools and an orphanage, before his interests turned to the education of early childhood, influenced by the philosophical ideas of Jan Ámos Komenský[236] (*id.*, *ibid.*).

Underlying the pedagogy of Fröbel is the conviction of a profound "*unity of reality*": a profound convergence between physical reality, the human self, and divinity (Abbagnano & Visalberghi, 1959, vol. III, pp. 609-610). But the modern-day relevance of Fröbel lies in the methods he devised and sought use in connection with this conviction, rather than in his philosophical ideas themselves. For Fröbel did not just use the teachings of Pestalozzi, but had several crucial pedagogic intuitions, which together with his philosophical thinking provided the base for his then-revolutionary methods.

[236] Czech philosopher (1572-1670) better known by the Latin name of Comenius, whose didactic works put forward the notion of a "pansophic" system of education: "*universal by its very nature* (…) *intended for all men, irrespective of social or economic position, religion, race or nationality. It must be extended to all peoples, however 'underdeveloped', as we say today, they may be* (Piaget, 1957, p. 10). This included the notion of education regardless of age. Comenius recognized four different stages in education (infancy – from birth to age 6, childhood, adolescence, and youth), but "*grasps the fact that the same forms of knowledge are necessary at each of the different levels, because they correspond to permanent needs; and that the difference between these levels lies mainly in the way in which the forms of knowledge are re-outlined or restated* (*id.*, p. 4). Even young children should thus learn in all knowledge areas. Details of his ideas in this regard were presented in his 1633 work, *Informatorium Školy Mateřski*, "The School of Infancy" (Komenský, 1633).

The first crucial idea was that in order to fulfill his view of unity with reality, children should be **assisted in their development, rather than molded or formed**; and that such development implied the educator's concern with three areas (Santomé, 1991, p. 17): **creation** (by promoting activities), **perception** (by promoting sensibility), and **reflection** (by promoting cognition).

A second crucial insight was on how to avoid the danger of imposing a model on the child: this could be achieved by **recognizing the value of child-play in the development of a connection between the child and the world**. In his words, "*in this stage of life, playing is the purest and most spiritual of the products of man; and at the same time, the model and image of all human life, of the intimate, secret, natural life of man and all things. It thus generates joy, freedom, satisfaction, inner and outer rest, peace with the Universe. In it reside the sources of all good, and from it they pour*" (Fröbel, as cited by Abbagnano & Visalberghi, 1959, vol. III, p. 612, translated from the Portuguese text).

The third crucial insight was that the development of activities could be enhanced by **giving the child access to specially-crafted educational materials**, that is, he was the first to realize the need for items with the single purpose of being didactic materials, and took the task of design and test them. These were called "gifts" (Figure 216), for they were not mere haphazard concoctions, but rather strongly connected with the metaphysical principles underlying his philosophy (Santomé, 1991, p. 17).

Figure 216 – Fröbel's Gifts (modern versions)
Composed from images at: http://www.ozpod.com/gallery/set.html

In summary, the central impact of Fröbel in current-day pedagogical practice is this relevance given to helping each child develop by means of spurring her personal interests, and doing so with a strong emphasis on the manipulation of objects and on the development of activities, rather than by imposing knowledge and facts. His widespread influence is both due to his own efforts in training other adults to use his methods, and to the efforts of enthusiasts such as Bertha von Marenholtz-Bülow, who through conferences, travels, correspondence, and books helps spread Fröbel's methods and ideas throughout the Western world (Gomes, 1977, p. 18; Smith, M., 1997).

Many educators followed on Fröbel's stead, with varying impact and relevance. Of those, one that is mentioned most often is Margaret McMillan, from Scotland. Her work and thought, prevalent in the foundation of the "nursery school" movement, exerted a strong influence in the development of early childhood education, particularly with its central notion of nurturing the child, not just foster her instruction (Vasconcelos, 1993, p. 6). In what regards this thesis, however, only some of the ideas and contributions of Margaret McMillan are relevant, although being but a small part of her action and thought. Those are: the **usage of educational materials** (wooden building blocks she called "bricks") **without predetermined goals**, allowing children to fully "*employ their creative energy*" (*id.*, p. 7), something which was similar to Fröbel's philosophy, but not similar at all to his recommendations for use of the "gifts", which included specific activities; and the recommendation that **educators should be prepared to expect and anticipate a nice learning moment**, and continuously experiment with the preschool curriculum **and place before the child progressive levels of effort** (*id.*, p. 8). This was an anticipation of the current concept of "scaffolding", proposed later by Vygotsky (*vd.* section 4.1.2, p. 260).

Figure 217 – Margaret McMillan, 1860-1931
From:http://www.electricscotland.com/history/women/images/MargMcM.jpg

But Margaret McMillan also marks the final of the 3rd phase of the evolution of preschool evolution, as presented in p. 243. The early 20th century is the beginning of novel educational endeavors, inspired by scientific studies of pedagogy.

Figure 218 – Maria Montessori 1870-1952
From: http://www.awo-coburg.de/kindergarten/kindergarten_coburg/bilder/k63.jpg

The foremost educator in this early period of scientific influence was Maria Montessori, from Italy. She was a neuropathologist, who during her practice with mentally handicapped children at a psychiatric clinic acquired an interest for their education. She applied methods, concepts, and materials inspired in the work of other physicians, such as Pereira[237], Itard[238], and Séguin[239] (Abbagnano & Visalberghi, 1959, vol. IV, p. 851; Röhrs, 1994; Vasconcelos *et al.*, 1995). This interest developed into a care for the education of children in general, realizing that what had proved so fruitful for mentally-impaired children aged 8 to 10 might provide "*precious occasions for free sensory-intellectual organization to a normal children aged 4 or 5*" (Abbagnano & Visalberghi, 1959, vol. IV, p. 851). She had the opportunity to put these concepts in practice in 1907, when "*she became involved in the modernization of a Roman slum quarter* (…) *by assuming responsibility for the education of the children. Her answer* (…) *was the establishment of a Children's House (Casa Dei Bambini), in which the children were to learn about the world and develop the ability to plan their own lives*" (Röhrs, 1994).

As one would expect, of course, the scientific influence was not the single base of Montessori's pedagogy. It also embraces notions that already existed, such as Fröbel's vision of the education of young children as a support to their personal development, and his notions of self-activity as an essential part of one's learning (Spodek & Saracho, 1993, p. 51).

> *« Pedagogy must be guided as in the past, by the ideas which some philosophers and philanthropists formed of it, by some individual inspired by piety, sympathy or charity. Pedagogy must follow the guidance of psychology, by the psychology which, applied to education, should at once be given a distinct name: Psycho-pedagogy. »*
>
> *(Montessori, 1955)*
>
> *« Although Maria Montessori based her work on scientific principles, she nevertheless considered childhood to be a continuation of the act of creation. This combination of approaches is the truly fascinating aspect of her work: on the one hand she practised precise experiment and observation in the spirit of science, yet at the same time she regarded faith, hope and*

[237] Jacobo Rodríguez Pereira (1715-1780), was a Spanish expert in the oral education of the deaf, working in Paris, who recognized the crucial role of embracing sensory input within the learning process (Plann, 1997, pp. 73-77; Röhrs, 1994).

[238] Jean Itard (1775-1838), was a French doctor that specialized on diseases of the ear, as Chief Physician at the National Institution for Deaf-Mutes in Paris (Plucker, 2003). He is better known for his educational work with "*the child known as 'The Wild Boy of Aveyron'* (...) *and he is recognized today as one of the founding fathers of special education*" (*id.*, *ibid.*). This boy was a child seemingly aged eleven or twelve, found naked and living alone in the woods of southern France, modernly believed to have been mentally retarded or autistic; Itard thought otherwise and developed an educational strategy to educate him, relying heavily on sensory-training and stimulation (*id.*, *ibid.*). His efforts had limited success, but he was "*the first physician to declare that an enriched environment could compensate for developmental delays caused by heredity or previous deprivation*" (*id.*, *ibid.*).

[239] Edouard Séguin (1812-1880) was a student of Jean Itard (*vd.* footnote 238, above). He *"claimed that he could change idiots by educating them and, more fundamentally, that idiots could benefit from the effort* (…)*. Once idiots showed that they could learn, they became worth understanding, and they became worth changing*" (Noll & Trent Jr., 2004, p. 2). His approach used muscular exercises to induce a change in behavior, a method he described as physiological education (*id.*, *ibid.*).

> *trust to be the most effective means of teaching children independence and self-confidence. »*
>
> *(Röhrs, 1994)*

The essence of her contribution, however, departs from Fröbel's views. The most crucial aspect is the emphasis given in her educational methods to sensory aspects, not just cognitive or social aspects. Traditionally, in philosophical views of education, the child's increased awareness and interpretation of the senses, as well as the development of the child's interactions with the physical world, were but a consequence of the cognitive and personal development. Montessori's scientific orientation involved experimenting, observing, and recording events and conditions that were external to the child, and as mentioned above, her previous experience involved the use of sensory-based techniques in the education of the mentally-impaired.

Therefore, the role of educational materials in her methods is not to convey specific notions of knowledge, but rather to serve as foci for the child's interests and energy, by which means the child reaches her own conclusions, directed by the teacher, rather than instructed (Vasconcelos *et al.*, 1995). This view of the role of educational materials is somewhat shared with that of Margaret McMillan, already mentioned on p. 246. But while McMillan's wooden blocks were seen as a means for self-expression, **Montessori's materials were meant for exploration and discovery**. She experimented and tested various items, such as button frames, solids of various shapes, wooden letters and numbers, sets of wool balls with various graduations of color for sorting, etc. (Abbagnano & Visalberghi, 1959, vol. IV, p. 852).

In summary, under the views of Montessori education the child develops herself, by contact with individual and social experiences, and learns from their consequences. The teacher's role is seen as one of **directing** the child's activities, and educational materials are to be carefully studied and planned in order to promote a rich exploration and discovery of concepts which, in the end, are effectively developed by the own child. These ideas already contained an embryo of the theories that other scientists and philosophers would develop, and eventually be known as constructivism (*vd.* section 4.1.2).

> *« Scientific observation then has established that* (…) *education is a natural process spontaneously carried out by the human individual, and is acquired not by listening to words but by experiment upon the environment. The task of the teacher becomes that of preparing a series of motives of cultural activity, spread over a specially prepared environment, and then refraining from obtrusive interference. Human teachers can only help the great work that is being done, as servants help the master. Doing so, they will be witnesses to the unfolding of the human soul and to the rising of a New Man who will not be the victim of events, but will have the clarity of vision to direct and shape the future of human society. »*
>
> *(Montessori, 1946)*

Montessori and other science-based educators of her time also mark the beginning of "modern" early-childhood education. Alongside the development of educational methods and theories, the period summarized in this section, from Oberlin to Montessori, was also the scene of an ever-increasing spread of the education of young children outside the home environment. Preschool education eventually ceased to be a privilege of some well-to-do families, or a requirement of a profession, gradually acquiring its current status: a service provided to the population in general. This reach-out process took place at different times across the world, throughout the 20th century; and in some regions or countries much must be done in order to render preschool education available to most children. But throughout the developed world, preschool became part of the regular schooling of most children (Spodek & Saracho, 1993, p. 57; Beatty, 1998).

This is not to say that the development of new pedagogy has ceased or diminished since Montessori. On the contrary, since then there has been an immense increase in the number of people involved in research, philosophy, and thought regarding the theme of early childhood education. But not all "modern" developments possess the same historical relevance, or indeed adequate background relevance regarding this thesis. For this reason, the next section presents a selection of theories of education representing the major and more relevant driving ideas behind current educational practice.

4.1.2. Main current theories

The development of children is the core concern of various modern theories, but these are not evenly balanced regarding the various development areas: cognitive, linguistic, physical, emotional, and social (Spodek & Saracho, 1993, pp. 79-80). Within the scope of this thesis, the main area of interest is cognitive development, and with a lesser degree social development[240]. The following table presents the main theories under this light (after Spodek & Saracho, 1993, pp. 66-78; Smith, 1999; and Wegerif, 2002), indicating their most influential theorists.

Behaviorist	Constructivist	Ecological Systems	Maturational	Participatory	Psychodynamic
Watson, Thorndike, Skinner	Piaget. Vygotsky, Bruner, Papert	Bronfenbrenner	Stanley Hall, Arnold Gesell	Lave, Wenger	Freud, Erikson

Table 21 – Main cognitive development theories and theorists

Modern pedagogy for early childhood education is also still somewhat based on the pioneers mentioned in the previous section, with the actual degree of influence varying among national educational systems and particular educational approaches of each school and teacher. But the major driving forces behind current practice are these more recent approaches based on the scientific research of child development, listed in Table 21, above.

However, the contributions to educational practices in early childhood education are not evenly distributed among those theories. Mostly, their contributions have only marginally impacted common everyday practice, albeit some had significant impact in the past. Currently, the major influence comes from the set of theories known as **constructivism**, whose description occupies the main part of this section. I therefore first provide just a brief overview of those other modern theories possessing some relevance regarding this thesis, and their impact on the practice of early childhood education. Then I'll conclude this section with the description of constructivism.

In overview, a central difference between these several theories lies in how they approach and balance two major influences on the child's cognitive development: heredity and environment, *i.e.* **internal influences and external influences**. Two early theories (behaviorist and maturational) focused on just one of these influences, and had a large impact on the practice of the first decades of the 20th century. The remaining theories adopted less extreme approaches to these perspectives, their influence proving more lasting.

Under the **maturational** theory, whose major theorists were the American thinkers and researchers G. Stanley Hall and Arnold Gesell, the internal influences were seen as predominant: genes constituted the major influence on children's development, and the environment played but a lesser part. It was established by G. Stanley Hall, a pioneer researcher that combined psychology and education, developing a child-centered approach to education (Stanley Hall, 1901). Stanley Hall's ideas had themselves been inspired by early research on genetic evolution, including the flawed and fraud-tainted idea (popular at the time and remaining so for many years) that the development of embryos recapitulated the evolution of species. According to Hall, the human mind would similarly evolve over various stages, and any attempt to change this course of events would be counter-productive (Spodek & Saracho, 1993, pp. 66-67; Grezlik, 1999).

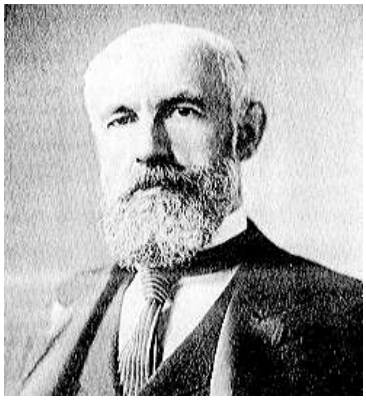


Figure 219 – G. Stanley Hall 1844-1924
From: http://www.muskingum.edu/~psych/psycweb/history/hall.jpg

[240] Social development is partly in the scope of this thesis due to concerns of contextual integration presented in section 6.2 and chapter 1 regarding teacher intervention.

Gesell built on Hall's ideas by conducting research in a more systematic manner, with careful and duly study of children. His research resulted in the creation of the Gesell Development Schedules, a set of stages grouping physical, language, personal and social developments and traits, aimed at children between four weeks and six years of age. Under the basic philosophy of Stanley Hall and Gesell, the role of education is to support or avoid obstructing children's natural development process, and children should be assessed to evaluate their level of readiness. The teacher should avoid presenting a child with a specific theme before the child is ready to approach it (Spodek & Saracho, 1993, pp. 67-68). "*There is an inner timetable which determines the child's rate of development. Trying to teach activities ahead of that timetable will at best result in only minor, temporary growth*" (Gesell Institute, 2004).

Figure 220 – Arnold L. Gesell 1880-1961
From: http://www.freudianslip.co.uk /images/arnold-gesell-portrait.jpg

The maturational theory was highly influential during the first decades of the 20th century, but eventually gave way to constructivist approaches to education in most educational institutions. Its dismissal of attempts to better a child's development was criticized for causing self-fulfilling prophecies, since it may cause parents or teachers to give up on a child by considering his/her future to be predetermined by genetics and therefore being little point in intervening (Meisels, 1988). Current early childhood education programs assume that while some children do indeed achieve the learning expectations without requiring specific intervention, other children require the specific support and intervention of educators: many children have access routinely to formal and informal experiences that provide success in learning, but others may not have access to such experiences or suffer from developmental deficiencies or delays limiting their abilities to relate to them (Spodek & Saracho, 1993, p. 68).The contributions of the maturational theory that are still relevant in common modern-day practice are its emphasis on an educational process centered on the needs of the child, rather than on the needs of teaching, and the concept of progressive child development – progressive not just in its amount, but also in its nature and qualities.

Another early theory was **behaviorism**, whose basic tenets have already been mentioned in section 2.4.3. It mirrors the aforementioned maturational theory, in that it sees the environment as the main source of influence on human development, disregarding internal influences. The ideas of behaviorism developed from classical psychological research on conditioning, namely classical conditioning principles derived from the well-known Pavlov's experiments[241].

From those principles, American theorists John Watson (Figure 221, on the right) and Edward Thorndike (Figure 37, p. 66) coined the term *behaviorism* and made the first efforts to apply the principles of conditioning to human learning.

Figure 221 – John B. Watson 1878-1958
From: http://www.psychologie.uni -heidelberg.de/ae/allg/lehre/wct/e/ E11/Bilder_1-1/Watson.jpg

They focused on controlling the environment to influence the learning and development of each individual, and researched various

[241] "*The work that made Pavlov a household name in psychology actually began as a study in digestion. He was looking at the digestive process in dogs, especially the interaction between salivation and the action of the stomach. He realized they were closely linked by reflexes in the autonomic nervous system. Without salivation, the stomach didn't get the message to start digesting. Pavlov wanted to see if external stimuli could affect this process, so he rang a metronome at the same time he gave the experimental dogs food. After a while, the dogs – which before only salivated when they saw and ate their food – would begin to salivate when the metronome sounded, even if no food were present. In 1903 Pavlov published his results calling this a 'conditioned reflex,' different from an innate reflex, such as yanking a hand back from a flame, in that it had to be learned. Pavlov called this learning process* (...) *'conditioning.' He also found that the conditioned reflex will be repressed if the stimulus proves 'wrong' too often. If the metronome sounds repeatedly and no food appears, eventually the dog stops salivating at the sound*" (PBS, 1998).

perspectives on the learning impact of applying stimuli and providing responses to the subject's behavior (Spodek & Saracho, 1993, p. 69).

The central figure of behaviorism was however another American, Burrhus F. Skinner (Figure 39, p. 68), who developed the concept of Operant Conditioning (Kearsley, 2004). As described in section 2.4.3 (p. 68), Skinner discovered that the subject of conditioning wasn't merely a passive receptor of stimuli and target of responses. In his research, he found out that rats would press a bar more often not in response to a stimulus, but in anticipation of the consequence of pressing the bar (more food, for instance). This realization meant that the subject of conditioning did in fact behaved in a way that "*operated on the environment and was controlled by its effects*" (Vargas, n.d.), thus Skinner named this Operant Behavior. The conditioning that accounted for this and arranged the "*contingencies of reinforcement responsible for producing this new kind of behavior*" (*id.*, *ibid.*) was thus named Operant Conditioning. Under Skinner's views on the application of these concepts to humans, since each person's actual internal mental processes cannot be witnessed, they must be ignored, any discussion of them being seen as merely speculative. As a consequence, teachers should only strive to prepare adequate responses to the learner's behavior, providing rewards for positive behavior ('positive reinforcement'), and not build models of student's methods for concept acquisition. An alternative method would be to remove from the environment some pre-existing negative stimulus ('positive reinforcement'). He did not believe in providing unpleasant results as an effective way to better learning (Spodek & Saracho, 1993, p. 70).

With an impact record similar to maturational theory, the main influence of behaviorism on early childhood educational practice also occurred in the early decades of the 20th century, until constructivism eventually assumed a preponderant status in driving the practice. The current influence of behaviorism is at the level of the attention that is given to the keen observation of children's behavior in the context of each child's educational environment, regarding the stimuli and responses that are present. But in general, early childhood teachers' intervention is no longer based on the planning and preparation of stimuli and responses (*id.*, p. 71).

These two early theories and constructivism, which is described later, provided encompassing approaches to early childhood education in the light of global cognitive development. In contrast, the remaining theories from Table 21 have contributed with perspectives on different aspects of child development, rather than over its entirety. In this light, they all remain influential sources of the current practice in early childhood education.

The earlier of these remaining theories is the **psychodynamic** theory. Rather than focusing on cognitive development, it deals with factors affecting the development of personality. Its relevance for this thesis is therefore at the level of its contributions regarding the role and intervention of the early childhood teacher.

Figure 222 – Sigmund Freud
1856-1939
From: http://www.wien.gv.at/ma53/rkfoto/2004/592g.jpg

The psychodynamic theory of child development is based on the concepts of psychoanalysis, seen as the observation that "*individuals are often unaware of many of the factors that determine their emotions and behavior,*" and that these "*unconscious factors may create unhappiness, sometimes in the form of recognizable symptoms and at other times as troubling personality traits, difficulties in work or in love relationships, or disturbances in mood and self-esteem*" (APA, n.d.).

The concepts of psychoanalysis were originally developed by the Austrian neurologist Sigmund Freud, who worked mainly with adult patients, but devoted particular attention to their memories of childhood events. He strongly emphasized the role of early childhood in the development of personality. His insights dealt the with the concept of different development stages, which he called Psychosexual Stages of Development, during which children focus their attention on different sources of stimulation and pleasure, their personality developing in the process. Freud's

ideas on child development exerted a strong influence on childhood psychology and childhood education (Spodek & Saracho, 1993, pp. 71-72).

Freud's original focus on the individual self as the source of development influences was expanded by other thinkers and researchers, to encompass cultural influences. Among these, the German researcher Erik Erikson was highly influential in the practice of early childhood education: his focus of research included the development of children themselves, rather just the influence of childhood development on adult psychological traits. He replaced Freud's psychosexual stages of development with different set of stages, focusing on several interrelated "crises" that individuals face while interacting with the environment and society, their resolution impacting the psychological development of the child, adolescent, and adult (Erikson, 1959, p. 128; Spodek & Saracho, 1993, p. 72).

Figure 223 – Erik Erikson 1902-1994
From: http://rcswww.urz.tu-dresden.de/~dornhoef/erik.jpg

The ideas and research of Freud and Erikson remain influential regarding the importance of childhood experiences in the development of the personality of the individual. In the scope of this thesis, the most important consequence of their thoughts is that both support an active, intervening role for early childhood education professionals, although the roles proposed by both differ. Freud's view holds that teachers should create a healthy environment where children can express their feelings, but Erikson's views are more relevant to this thesis: he held that teachers should coach children so that they develop skills that strengthen their egos, helping them avoid the negative sides of each psychosocial stage, a relation which Erikson referred to as the criteria of relative psychosocial health *vs.* ill-health, as presented in Table 22 (on the next page) within the double-bordered cells (Erikson, 1959, pp. 128-130).

In Table 22, adolescence is presented as the axis of personality, although in it Erikson devotes much more attention to the childhood stages than to the evolution of adult life or even the evolution of adolescence itself. The age group of preschoolers is identified with its 2nd and 3rd stages, which Erikson situated respectively between the 2nd and 3rd years of age, and between the 4th and 5th years of age (*id.*, p. 155).

Erikson sees the second stage as a phase when the most delicate balance is between personal autonomy and feelings of shame (which he considers a consequence of internal doubts); and the third stage as a crucial phase of balance between personal initiatives (fantasy and play) and guilt. A negative balance in either stage would impact adolescence (under the criterions in the fifth row and fifth column), and from there, the entire adult life of the individual (*id.*, *ibid.*).

I Infancy	**Trust vs. Mistrust**				Unipolarity vs. Premature Self-differentiation			
II Early Childhood		**Autonomy vs. Shame, Doubt**			Bipolarity vs. Autism			
III Play Age			**Initiative vs. Guilt**		Play Identification vs. (Oedipal) Fantasy Identities			
IV School Age				**Industry vs. Inferiority**	Work Identification vs. Identity Foreclosure			
V Adolescence	Time Perspective vs. Time Diffusion	Self-certainty vs. Identity Consciousness	Role Experimentation vs. Negative Identity	Anticipation of Achievement vs. Work Paralysis	**Identity vs. Identity Diffusion**	Sexual Identity vs. Bisexual Diffusion	Leadership Polarization vs. Authority Diffusion	Ideological Polarization vs. Diffusion of Ideals
VI Young Adult					Solidarity vs. Social Isolation	**Intimacy vs. Isolation**		
VII Adulthood							**Generativity vs. Self-absorption**	
VIII Mature Age								**Integrity vs. Disgust, Despair**

Table 22 – Erikson's psychosocial stages

From: Erikson, 1959, p. 129

A more recent theory impacting the practice in early childhood education is the **ecological systems** theory, originally proposed by the Russian-born researcher Urie Bronfenbrenner (Spodek & Saracho, 1993, p. 78).

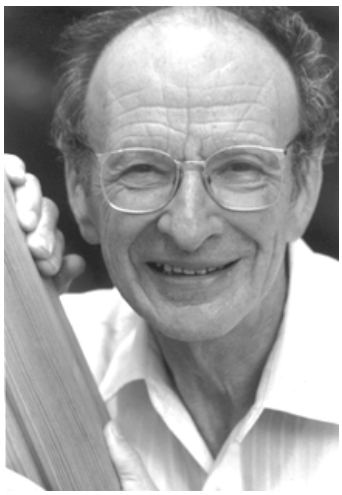


Figure 224 – Urie Bronfenbrenner 1917-
From: http://www.news.cornell.edu/Chronicle/96/1.18.96/urie.gif

This theory focuses on the environmental influences on child development, but significantly expanded the concept of "environment," as I will soon explain. It sees development as "*a lasting change in the way in which a person perceives and deals with his environment*" (Bronfenbrenner, 1979, p. 5). A key element in this statement is the emphasis it places on the **personal perception** of the environment, rather than on any possible "objective" features of it – a concept based in the early work of Polish theorist Kurt Lewin, as expressed in the quote below. I have already analyzed these views, in the context of the relationship between abstractness and concreteness, in section 3.3.3.

> *"Essentially* (...) *the phenomenological concept of environment that is at the base of this theory derives its structure and logic principles from the ideas of Kurt Lewin* (...). *Lewin takes the stand that the environment that is most relevant to the scientific understanding of behavior and development is not reality as it exists in the so-called objective world, but how it appears in the person's mind; in other words, he focuses on the way in which the environment is perceived by human beings that interact within it and with it."*
>
> *(Bronfenbrenner, 1979, p. 19)*

Figure 225 – Kurt Lewin 1890-1947
From: http://home.ubalt.edu/TMITCH/LEWIN.JPG

But from this early notion Bronfenbrenner expanded the aspects and reach of what constitutes relevant reality in this regard. His notion of "environment" goes beyond the current and historical immediacy of a child and his/her behavior, which under the ecological systems theory are simply the environmental **microsystem**. His notion of environment includes the relationships between several such microsystems where the child is actively participating (*e.g.*, child's home, school, friend's homes), and these form the environmental **mesosystem**. Further, and most significantly, he expanded the notion of environment into settings where the child is not an active participant and isn't, has not been, and possibly won't even ever present, but where nonetheless "*events take place that affect, or are affected, by what is happening*" (Bronfenbrenner, 1979, p. 21) in the places where one can find the developing child – and these settings form the environmental **exosystem**. Finally, he also takes in account the relevance of global influences on all the previous environmental systems, such as cultures and subcultures, and their relationships in society, what he calls "*consistencies in the form and content of lower-order systems* (...)*, that exist* ***or might exist****, at the level of subculture or culture as a whole, along with any belief system of ideology underlying those consistencies*" (*id.*, *ibid.*, my emphasis) – and he refers to these consistencies as the environmental **macrosystem**.

The main impact of these views on educational practice in early childhood has not been at the curriculum level, but rather at the level of social policy (Spodek & Saracho, 1993, p. 78). This was consequential from Bronfenbrenner's novel perspective on what constitutes an environmental impact on development, by including exosystem and macrosystem settings: "(...) *an ecological approach to the study of human development requires a reorientation of the conventional view of the proper relation between science and public policy.* (...) *what is required is not merely a complementary relation between these two domains but their functional integration*" (*id.*, p. 9).

The most recent of the theories mentioned in Table 21 deals with **participatory** learning, which takes on learning under the perspective of social participation in a group. The main contribution for this perspective on human learning came from the work of American social anthropologist Jean Lave and Swiss researcher Etienne Wenger. They have developed the concept of **communities of practice** and its role in the process of learning (Smith, 2003):

Figure 226 – Jean Lave and Etienne Wenger
From:
http://www.vanderbilt.edu/lsi/images/JeanLave.jpg
http://www.ewenger.com/bio/Etiennes_informal_JPEG_photo.jpg

> *« Being alive as human beings means that we are constantly engaged in the pursuit of enterprises of all kinds, from ensuring our physical survival to seeking the most lofty pleasures. As we define these enterprises and engage in their pursuit together, we interact with each other and with the world, and we tune our relations with each other and with the world accordingly. In other words, we learn. Over time, this collective learning results in practices that reflect both the pursuit of our enterprises and the attendant social relations. These practices are thus the property of a kind of community created over time by the sustained pursuit of a shared enterprise. It makes sense, therefore, to call these kinds of communities "communities of practice." »*
>
> *(Wenger, 1998)*

Lave and Wenger do not approach learning as the acquisition of knowledge, but rather as a process through which people gradually involve themselves with communities that possess some internal structure. As people engage in social relations, and get formally or informally more involved in communities of practice, they start by being on the **periphery** of such communities. As learning develops, people are able to participate more full in the life of such communities, and gradually move to the **center** of the communities, going from newcomers to old-timers, from apprenticeship to mastery (Smith, 2003).

> *« Asserting that it is learning that gives rise to communities of practice is saying that learning is a source of social structure. But the kind of structure that this refers to is not an object in itself, which can be separated from the process that gives rise to it. Rather it is an emergent structure* (...)*. Indeed, practice is* (...) *produced by its members through the negotiation of meaning. The negotiation of meaning is an open process, with the constant potential for including new elements. It is also a recovery process, with the constant potential for continuing, rediscovering or reproducing the old in the new. »*
>
> *(Wenger, 1998)*

The ideas of Lave and Wenger have received a significant level of attention from education professionals, associations, and policy-makers, mostly at the level of teacher training and professional development (*e.g.*, Buysse *et al.*, 2003; New Zealand Ministry of Education, 2005). However, their relevance to the subject matter of this thesis derives from successful efforts in the application of these concepts with school children. These efforts provided important notions, namely that schools "*must prioritize instruction that builds on children's interests in a collaborative way*" (Smith, 2003), of "***intimate connection between knowledge and activity***" (*id.*, *ibid.*), and of the need for learning activities that are "*planned by children as well as adults, and where parents and teachers not only foster children's learning but also learn from their own involvement with children*" (Rogoff *et al.*, 2001, as cited by Smith, 2003). These notions are quite similar to several aspects of Seymour Papert's constructionism (presented in section 4.2.2), and are also linked to the integration of activities in preschools, discussed in section 7.2.

The theories previously presented in this section fell under two different categories: the first two presented (maturational and behaviorist) were, for several decades, major influences on the whole early childhood educational practice. The others (psychodynamic, ecological systems, and participatory) have contributed to current practice in specific issues, but have not been fundamental influences in the overall practice.

Currently, as mentioned in the opening paragraphs of this section, the set of theories collectively known as **constructivism** provide the most important influence on the educational practice at the preschool level.

> *« The constructivist theories of Piaget (1971,* [1974]*), Bruner (1966, 1973), and Vygotsky (1978) permeate early childhood education. »*
>
> *(Skeele & Stefankiewicz, 2002)*

The basic idea of constructivist theories is that **human beings learn by developing personal interpretations from both external events and of their personal knowledge, and not by accumulating specific, person-independent information**. Consequently, "knowledge" is seen not as specific concepts with autonomous existence, but rather as a set of personal constructions, resulting from the multiplicity of different interactions a person experiences through life and that same person's interpretation of those experiences and previously-developed knowledge.

Constructivism thus provides a balanced approach between the influences on child development mentioned on p. 250 as internal influences and external influences (heredity and environment). Development is seen as the consequence of external influences, as in behaviorist theory, but such **developmental consequences are not objective properties of the external influences themselves; rather, they result from internal, personal interpretations and constructions of each person** in face of the external influences. As in the maturational theory, in constructivism this internal basis for development is seen as dependent on biological features of the human brain, but unlike maturationism those features are not seen as predetermining human development. Instead, constructivism sees the biological features as providers of open-ended functionality: **the brain is seen as a system that rules interactions, determining the processes but not the outcomes, the processes themselves subject to change, creation, and destruction**.

Although modern constructionist educational practice derives various influences from the many researchers that embraced these notions, the central contributions were made by Swiss biologist and educational researcher Jean Piaget (Figure 227), who established the fundamental notions above, and explored the mental processes involved in them (Papalia *et al.*, 1999). According to Piaget, the personal construction of knowledge takes place as a process of adaptation to the environment, which he called cognitive **equilibration** (Piaget, 1975), as I explain below.

Figure 227 – Jean Piaget 1896-1980
From: http://i.timeinc.net/time/time100/images/main_piaget.jpg

Piaget considered that all the events and circumstances of human life are seen by each individual in the light of his/her own existing conceptions and knowledge; the resulting views are then incorporated into mental constructs (personal memories and knowledge, for instance). In other words, the personal experiences are interpreted differently by each individual, and these interpretations are – using Piaget's terminology – **assimilated** into the pre-existing notions of the individual. Each assimilation, to a greater or lesser degree, unbalances personal concepts, by contradicting them or rendering them insufficient. To correct this unbalance, aiming towards equilibrium, the human mind then needs to change its internal constructs, and Piaget called **accommodation** to this mirror process.

The full process of balance between assimilation and accommodation thus constitutes the Piagetian concept of **equilibration** I mentioned earlier. This full process must be understood not as one of progression between static states of equilibrium, but as **a continual interactive process**: humans are continuously adapting new information and experiences, and accommodating to them, but also behaving in accordance to this process, therefore continuously impacting the information and experiences that generate future assimilations and accommodations (Fosnot, 1996, p. 30).

> *« (...) all behavior is adaptive and (...) adaptation is always some form of equilibrium (stable or unstable) between assimilation and accommodation (...) »*
>
> *(Piaget, 1962)*

These concepts were the consequence of a fundamental change brought by Piaget to the field of child-development research: his concern with and focus on **children's internal mental processes**, rather than the observable external influences and behaviors.

During his long research career, Piaget researched many finer aspects of children's mental processes, while developing the global processes described above. In doing so, he established a popular set of stages describing the cognitive development of children, presented in Table 23. However, these stages have in recent years fell under criticism, in light of new research. Foremost among this criticism are, according to Papalia *et al.* (1999): the stages' focus on "average" children, not accounting for individual differences; some underestimation of the abilities of babies and young children; the strict demarcation provided by the stages, rather than a progressive, continual evolution; and its universal, single progression towards formal thinking.

> *"Studies initiated in the late 1960s suggest that children's cognitive processes are closely linked with specific contents (what they're thinking about), as well as the context of a problem and the kind of information and reasoning that is seen as important in a given culture (...)."*
>
> *(Papalia et al., 1999, p. 32)*

This last bit of criticism, in particular, is quite relevant to this thesis. Mainly, regarding the recent views on the concepts of "concrete" and "abstract", and the valuation of other thinking styles besides formal thinking (*vd.* Turkle & Papert, 1990), as I discussed in section 3.3.3. The impact of context is also relevant in light of the discussion I provide in sections 7.1 & 7.2.

Age	Stage	Features
Birth – 1½ or 2 years	Sensory-Motor	Children develop schemes based on sensory information and bodily movements.
2 – 7 years	Preoperational	Development of language and other symbolic representations. Intuitive thought is neither systematic nor sustained.
7 – 11 years	Concrete Operations	Use of logic processes, but applying only one form of classification at a time; logic thought requires physical objects or concrete events.
11 years onward	Formal Operations	Logic reasoning, formulation and testing of hypotheses, abstract thinking.

Table 23 – Piaget's Stages of Cognitive Development

From: Spodek & Saracho, 1993, p. 75

This criticism notwithstanding, Piaget's stages of Table 23 are still relevant if seen as providing a global overview of the evolution of specific thinking abilities, rather than a strict evolutionary map. Piaget himself may have seen them as more flexible than some of the above criticism assumes (Papert, 1999b). Also, they're useful as a background for understanding finer Piagetian concepts.

Among those finer concepts, the most obviously relevant for this thesis are the limitations to thought that Piaget associated with the Preoperational stage, since the age group in which he identified them (2 – 7 years) is almost entirely associated with preschool children (2½ to 6 years). These limitations to thought are summarized in Table 24.

Limitation	Description	Example
Centration	The child focuses on a single aspect of the situation, disregarding any other.	John cries when his father gives him a biscuit broken in half. Since each half is smaller than the full biscuit, John thinks he's getting less.
Irreversibility	The child is unable to realize that an operation or action can be reversed.	John doesn't realize that both halves of the biscuit can be joined to make a full biscuit.
Static thinking	The child is unable to realize the meaning of state transformations	In a conservation[242] task, John fails to realize that the shape transformation of a liquid (from a glass to another) doesn't change the quantity.
Transductive reasoning	The child doesn't employ either deduction or induction; going instead from one particular aspect to another, seeing a cause where there is none.	"I had ill thoughts about my brother. My brother got ill. So, I made him get ill." Or: "I misbehaved, so mum and dad divorced."
Egocentrism	The child assumes that everyone thinks like he/she does.	Mary picks up a game and tells her mum, "This is *your* favorite." She's assuming that her mother likes the game as much as she does.
Animistic thinking	The child sees life in inanimate objects.	Mary thinks the clouds are alive because they're moving.
Inability to distinguish appearances from reality	The child mistakes appearances for reality.	John thinks that a sponge made to look like a rock is indeed a rock.

Table 24 – Piaget's limitations of preoperational thought

From: Papalia et al., 1999, p. 313

In section 6.1, as a result of the field work supporting this thesis, I present several hurdles that preschool children face when trying to program computers. Those hurdles are presented in light of existing research on human difficulties in computer programming. However, a different avenue of research can be envisaged: trying to establish associations[243] between Piaget's limitations, presented in Table 24, and my hurdles of section 6.1.

Among the above limitations, one is relevant to the overall panorama of this thesis, and that is the aspect of **egocentrism**, as I will now explain. As is mentioned in Table 24, one should note that in Piagetian terminology, this does not refer to the psychological condition of obsession with oneself, but to the lack of ability to engage in what he calls **intellectual co-operation**, which requires one to recognize the existence of different viewpoints and ways of thinking:

> *«* (...) *the only valid meaning of egocentrism: the lack of decentering, of the ability to shift mental perspective, in social relationships as well as in*

[242] A particular variety of Piagetian cognitive tasks, where one feature (such as length or volume) is kept while other features change, and children's perceptions of the transformation are analyzed.

[243] Some of these associations may seem almost obvious, but specific research is needed to establish associations that go beyond speculation. For instance, my second hurdle is "**Children may expect human-like interpretation of their intentions and autonomy in its execution by the computer.**" This may seem like an instance of animistic thinking, but it seems feasible to suggest that egocentrism is also playing a non-negligible role. The actual roles of such limitations can only be elucidated by conducting further research.

others. Moreover, I think that it is precisely co-operation with others (on the cognitive plane) that teaches us to speak "according" to others and not simply from our own point of view. »

(Piaget, 1962)

That Piaget identified this limitation as common in children's reasoning in the preschool age is relevant to the subject matter of this thesis in light of a particular requirement of computer programming: in order to understand how to teach a computer, the child programmer must realize how the computer will understand his/her commands. Such a requirement thus directly confronts Piaget's egocentric limitation of thought, and this idea is central to the educational philosophy of Seymour Papert, described in section 4.2.2 (it has also been mentioned earlier in section 2.4.5).

Any modern description of constructivism cannot be complete without reference to two other major research contributions: those from Russian psychologist Lev Semjonowitsch Vygotsky and American researcher Jerome Bruner.

The contributions of Vygotsky, in particular, have been gaining widespread acceptance and are becoming a major influence on educational practice (Spodek & Saracho, 1993, p. 76). Foremost reasons for this are the attention he devoted to the intertwined relationship of the development of thought and language (Vygotsky, 1934), and above all, the way his perspective complements Piaget's thinking. Piaget acknowledged the importance of social factors in child development, but focused on the internal processes ruling the child's development of thought. In contrast, Vygotsky's approach to the development of thought focused on the influence of social factors, a constant concern in educational settings.

Figure 228 – Lev Semjonowitsch Vygotsky 1896-1934
From: http://www.marxists.org/archive/vygotsky/images/1934.jpg

The development of Vygotsky's research and thought took place soon after the initial phases of Piaget's research. However, his premature death, the fact that his work was published in the 1930s but only in Russian, and that it fell out of favor of the political leadership of the Soviet Union until the 1960s, when the first English language translations were published, all mean that his contributions to the field of child development must be seen both in light of the posterior development of Piaget's constructivist ideas and in the context of the time of their original writing – a sometimes contradicting effort, since his educational thought contains both ideas that can be seen as behaviorist and as constructivist, even though this latter view is predominant (Rheta DeVries, 2000). Piaget himself states:

« It is not without sadness that an author discovers, twenty-five years after its publication, the work of a colleague who has died in the meantime, when that work contains so many points of immediate interest to him which should have been discussed personally and in detail. (...) *concerning Vygotsky's sympathetic and yet critical position with respect to my work, I was never able to read his writings or to meet him in person, and in reading his book today, I regret this profoundly, for we could have come to an understanding on a number of points.* (...) *on certain points I find myself more in agreement with Vygotsky than I would have been in 1934, while on other points I believe I now have better arguments for answering him. »*

(Piaget, 1962)

A central concept to Vygotsky's thinking is that of **mediation**. By considering the behaviorist relationship between stimuli and responses, Vygotsky realized that this relationship could be direct (when a response associated with a stimulus without interference) or mediated. In this latter case,

the response cannot be interpreted solely in face of stimuli, since other interferences (the mediation) can be also a small or major part of it. For instance, when one's hand approaches a flame, a direct response might be drawing it back upon feeling a burning pain, and this would constitute a direct relationship between a stimulus and a response. But one might also draw back the hand upon the simple sensation of heat, by interference of the memory of burning, or even without any sensation of heat at all, if another person provides a warning about the possibility of burning. Both these cases represent instances of mediated relationships between a stimulus and a response (Oliveira, 1997, pp. 26-27).

This concept of mediation provided a basis to Vygotsky's thinking, in that the human relationship with the world evolves from being essentially direct, to being essentially mediated. In his psychological research, he also studied the importance of symbolic elements in this mediation, both in terms of physical symbolic items and abstract mental processes, the latter developing into ever more complex mediation processes that are internal to the individual.

Thus, following a different path, Vygotsky did converge with Piaget's fundamental views of a relationship between the individual and the world as a continuous process of construction of "*internal signs, that is, mental representations to replace real-world objects*" (*id.*, p. 35), which gradually become the main mediations in that individual's relationship with the world. In this regard, Vygotsky particularly focused on the impact of language. Under his conception, when learning to speak, a child's development of meaning for a word is not developed in isolation, but rather as a result of the child's participation in her social group and culture[244]. It is this relationship with fellow humans that Vygotsky sees as fundamental in the cognitive development of each human being.

> *« As an individual only exists as a social being, as a member of some social group within whose context he follows the road of his historical development, the composition of his personality and the structure of his behaviour turn out to be a quantity which is dependent on social evolution and whose main aspects are determined by the latter. »*
>
> *(Vygotsky, 1930)*

A corollary of these views was the concept of learning seen as processes internal to the human mind that enable the individual both to participate in the cultural development of knowledge and in the **internalization** of its current cultural state – a concept shared by Bruner, as is mentioned further ahead in this section. The research and thought of Vygotsky on this matter involved the realization that children cannot equally internalize all concepts or notions to which they're exposed while participating in society: in some cases, the social intervention has no impact; in other cases, the child can partly employ the concepts, when provided with support by another person; and in yet other cases, the child can employ the concepts and notions in an autonomous manner.

Vygotsky concluded that somehow a child must be in a state of readiness in order to benefit from the social contacts he or she experiences – a notion with some echoes of the maturational theory, presented on p. 250. But this concept of readiness, under Vygotsky's research and thought, is fundamentally different from maturational theory. Instead of seeing each child as the maturationists did (developing autonomously, following a preset trajectory), Vygotsky considered social interactions to be absolutely crucial to the child's development – the intervention of other people is the major source of development, more important than any internal preset conditions.

His thought and research on education and cognitive development can be summed in the following manner: at any given time, a child is at an undetermined **actual development level**. This level represents the psychological functions of that particular child which are found to be fairly well developed and consolidated. Knowledge and concepts that can be fitted within the mental

[244] This concept was further explored by Alexander Luria and Jerome Bruner, as is mentioned further ahead in this section; but it is also a central idea to participatory theory, which was described in p. 256.

framework provided by these functions can therefore be internalized and employed by the child fairly quickly, autonomously. At the same time, the child has initiated the development of several other psychological processes, still undergoing adequate development. Knowledge and concepts that require these processes cannot be autonomously internalized by the child, but this early level of development may be sufficient if the child is provided with the assistance of an adult or a slightly more advanced peer. Vygotsky called this set of almost-consolidated concepts as the **zone of proximal development**, since it comprises the functions nearest to those already internalized[245]. Finally, knowledge and concepts requiring psychological functions whose development has not yet been started in the child cannot be internalized, no matter the level of support provided.

This line of reasoning has profound consequences regarding the role of education and educators. One is that since the zone of proximal development is the result of the child's life course, it is quite different and individualized, because each child will have a different set of almost-ready mental constructs – depending on each child's personal history and culture.

But another consequence it that since the child's immediate cognitive development is occurring in the child's zone of proximal development, the most effective role for education and the educator is that of intervention in this zone, aiming at advancements that would not take place spontaneously. Thus, according to Vygotsky, "*the only good education* (...) *is that which acts in anticipation of development*" (Oliveira, 1997, p. 62); that is, **education leads to development, instead of accompanying it** (Spodek & Saracho, 1993, p. 77). This aforementioned intervention of educators in the zone of proximal development is known, under Vygotsky's terminology, as **scaffolding**: providing children with the cognitive support they need to complete the development of already-initiated mental processes (*id.*, *ibid.*).

Both Piaget and Vygotsky provided significant advances to scientific knowledge and thought on the cognitive development of children, but their contributions to the application of their ideas on educational practice were quite limited.

Several researchers dedicated their attention to the implications of constructivist theory on educational practice; possibly the most influential of these was the American psychologist Jerome Bruner, as I mention further ahead. But Bruner also provided significant contributions to the constructivist foundations of Piaget and Vygotsky, having been in close contact with the ideas of both[246]. Among these contributions, two are particularly relevant in an overall perspective of constructivism, for they build from the concepts of Piaget and Vygotsky. I am referring to his concern with the developmental role of **mentalism** (Morejón & Sierra, 1998; Bruner, 2002; Lock, 2003), and more recently with his interest on the processes behind the **cultural** aspects or learning (Bruner, 1996; Smith, 2002).

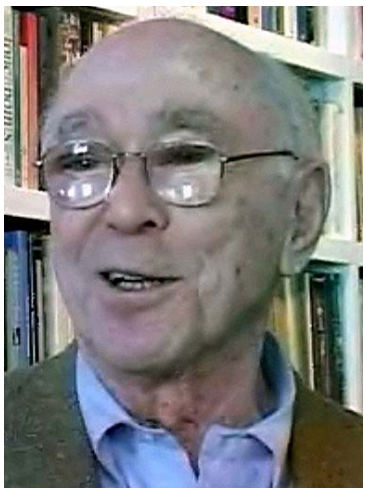


Figure 229 – Jerome Bruner
1915-
From:
http://luria.ucsd.edu/Luria.mov

> *«* (...) *Bruner managed* (...) *to demonstrate how unobservable, mental processes could be empirically investigated. The essential point* (...) *was that 'learning' could occur without any accompanying observable behaviour. That it was possible to do something 'mentally' rather than by pure trial-and-error behaviour. »*
>
> *(Smith, 2002)*

[245] The combination of the actual development level with the zone of proximal development would be the theoretical maximum level of psychological achievement within the child's reach at a given moment, and was therefore named by Vygotsky as the child's **potential development level**.

[246] Bruner corresponded with Piaget and – extensively – with one of the main collaborators and followers of Vygotsky, Alexander Luria (Bruner, 2002).

In other words, Bruner devoted attention to the particular learning that occurs as a result of personal introspection and reflection, but also without a conscious effort from the part of the learner. This theme that was not entirely absent from the work of Piaget and Vygotsky, but definitely secondary to their concerns about external experiences and interactions. For instance, he gave particular attention to the mental aspects of learning by categorization: from the contact with external information, humans can categorize it and structure it, drawing inferences and personal meaning. In a common example, someone who has never seen a parrot is likely to hypothesize that it is a bird, based on common features of known birds. (Bruner, 1957; Bruner, 1960; Smith, 2002). He famously referred to these mental processes as "*going beyond the information given*" (Bruner, 1957).

His research efforts under this overall concern for the mental component of learning were many and varied, including topics such as the cognitive strategies employed by learners (Morejón & Sierra, 1998), the often-cited[247] internal representations of knowledge (Spodek & Saracho, 1993, p. 77; Lock, 2003) and the role of structure in learning (Bruner, 1960; Smith, 2002).

> *«* (...) *learning that is produced by* (...) *transfer of principles, is dependent upon mastery of the structure of the subject matter* (...)*. That is to say, in order for a person to be able to recognize the applicability or inapplicability of an idea to a new situation and to broaden his learning thereby, he must have clearly in mind the general nature of the phenomenon with which he is dealing. »*
>
> *(Bruner, 1960, p. 18)*

From these concerns with the mental process of learning, Bruner moved on to researching their implications to educational practice[248], as can already be gleamed from the above citation. In fact, in that same text, Bruner explicitly refers to it by saying:

> *« It is simple enough to proclaim, of course, that school curricula and methods of teaching should be geared to the teaching of fundamental ideas* (...)*. But as one makes such a statement a host of problems arise, many of which can be solved with the aid of considerable more research. »*
>
> *(id., ibid.)*

His research on educational practice covered many topics, among which the aforementioned importance of **structure** in the subject matter; a novel perspective regarding **readiness in learning**, by which he recommended avoiding the postponing of subjects for fear of them being too difficult: "*We begin with the hypothesis that any subject can be taught effectively in some intellectually honest form to any child at any stage of development*" (*id.*, p. 33), and a proposed conciliation of these two concepts, by proposing a **spiral curriculum** – one where the same basic ideas are revisited often, repeatedly, continuously building upon them until their formality is fully grasped. These concepts are quite relevant for the present thesis, since I offer strategies for using, at the preschool level, what would commonly be seen as advanced concepts in computer programming, if properly adapted. *E.g.*, concurrency (section 3.2.1), constraint programming (section 3.2.2), and the various computer science and computer-programming concepts presented in the "cookbook" of section 3.4, which as stated there is not presented as a set of strict recipes of formulas, but rather as a dip allowing preschool students to taste several concepts of computer science and programming.

[247] Part of Bruner's thought on knowledge representations was his proposed model of systems of representation: **active** representation, by means of acts and manipulations; **iconic** representation, by means of images used in referral to specific information; and **symbolic** representations, by using language or other symbols to represent information and meaning (Spodek & Saracho, 1993, p. 77).

[248] The issue of applying theoretical principles in educational practice is presented in section 4.1.3, and is also discussed in the scope of the educational use of computer programming in sections 2.4.5 and 4.2.2.

Further, Bruner explored the importance of the combined role of **intuition and analytical thinking**, and the need to find **personal motives for learning**[249], rather than external motives such as grades or later competitive advantage in life (Smith, 2002).

As I mentioned above, on p. 262, more recently Bruner has devoted his research efforts to the cultural aspects of learning, a concern which had been originally proposed by Vygotsky. Bruner has explored the negotiation of meaning in culture, the processes by which people not just convey meaning but negotiate it and develop it, in a way that one can see meaning as residing in the culture itself, rather than inside any particular individual's mind:

> *«* [It is an] *evolutionary fact that mind could not exist save for culture. For the evolution of the hominid mind is linked to the development of a way of life where "reality" is represented by a symbolism shared by members of a cultural community in which a technical-social way of life is both organized and construed in terms of that symbolism. This symbolic mode is not only shared by a community, but conserved, elaborated, and passed on to succeeding generations who, by virtue of this transmission, continue to maintain the culture's identity and way of life. Culture in this sense is* ***superorganic****. But it shapes the minds of individuals as well. Its individual expression inheres in* ***meaning making****, assigning meanings to things in different settings on particular occasions.* (...) *Culture, then, though itself man-made, both forms and makes possible the workings of a distinctively human mind. On this view, learning and thinking are always* ***situated*** *in a cultural setting and always dependent upon the utilization of cultural resources. Even individual variation in the nature and use of mind can be attributed to the varied opportunities that different cultural settings provide, though these are not the only source of variation in mental functioning. »*
>
> *(Bruner, 1996, pp. 3-4)*

This concern is addressed in section 6.2 and chapter 1, which deal with the involvement of preschool teachers in the use of computer-programming with children aged between 3 and 5. While not considering a full cultural approach to the use of programming, those parts of the thesis provide a link to the embedding of programming in an important subculture of children's lives: school itself.

> *«* (...) *school is an entry into the culture and not just a preparation for it* (...) *»*
>
> *(id., p. 39)*

[249] A concept one also finds in Seymour Papert's educational ideas, when he calls for personally meaningful, *e.g.*, "ego-syntonic" learning projects (*vd.* sections 2.4.5 and 4.2.2).

4.1.3. Pedagogic models

The previous section presented an overview of the main educational theories impacting the current practice in early childhood education. But little information was then provided on how that impact takes place. Cognitive development theories inspire and support the practice, but the latter's success implies that many other complex requirements must be met, beyond cognitive development. And the implementation in practice of the theoretical concepts of cognitive development is in itself a complex research problem. This much was acknowledged and researched by Bruner, as mentioned at the end of the previous section, albeit not specifically for the preschool level.

The view on what issues are most relevant in putting theory to practice is in itself a contentious problem (not the least because it must be all-encompassing), and one that every professional educator faces. Educational systems thus typically develop "pedagogic models", a.k.a. "curriculum models." These are "*comprehensive educational systems*" (Formosinho, 2001, p. 109) that aim to be a supportive framework for educators, providing a link between educational theory and educational practice.

Being encompassing is in itself a mammoth task, one that no single educational theory has explored. Professional educators do not teach average children but individuals; they cannot focus solely on cognitive development, on social development, or psychological traits. Fundamentally, even if a specific educational issue has not been extensively researched, a professional educator cannot simply avoid facing it or postpone it until there is solid research ground: the children, the family, the issues are all varied and present themselves daily.

As a consequence, curriculum models combine scientific knowledge, heuristic knowledge, and philosophic principles, providing guidelines for assessment, teacher intervention, and both general and specific everyday details (*id.*, *ibid.*).

> *« Whereas in the scope of theory the greater educational purposes and their consequent goals are defined (at the level of the teaching-learning process), in the scope of practice one details guidelines for the various dimensions of the educational context. The physical environment, in what concerns space and materials; time organization within a daily routine and its everyday flow; the various levels of socio-educational interaction; work and play that are developed in this context* (...)*. »*
>
> *(Formosinho, 2001, p. 109)*

It is interesting to note that this was the case of the various historical educational models presented in section 4.1.1. They were created prior to the development of significant scientific research on the development of children, and regardless of their philosophical underpinnings, they all had to maintain a constant concern for practical issues, which were mostly absent during the discussion of theory in section 4.1.2 (with the notable exception of Jerome Bruner, as mentioned).

Besides Bruner, several education thinkers, philosophers and practitioners delved on the specific issues of educational practice. Amongst these, one particular researcher and thinker has devoted considerable effort in bridging the abreast fields of educational theory and practice building from an approach based on computer programming. I am referring to Seymour Papert, whose thought is presented[250] in section 4.2.2: "Seymour Papert and constructionism" (p. 283).

However, Papert hasn't specifically considered the age group of very young children, or the preschool education context. The present section thus provides some relevant background at this level, presenting some early thinkers that devoted their attention to the issues of educational practice, their insights still relevant in today's practice in early childhood education, and their thinking also echoed in Papert's views. Then, to complete the presentation of the modern

[250] Some highlights and a summary of Papert's educational thought were already presented in section 2.4.5.

educational process in preschools, I conclude this section by presenting some modern pedagogic models, providing a broader view of the field.

Figure 230 – John Dewey
1859 - 1952
From: http://www.marxists.org/glossary/people/d/pics/dewey-john.jpg

It's interesting to note that the central ideas are recurrent in the pedagogic thought of these philosophers and educators, from the earliest, the American philosopher John Dewey (Figure 230, on the right), to the most recent of the three, French teacher Célestin Freinet (Figure 232, on p. 266) – obviously, with some distinctions, as I'll discuss below. The misinterpretation of their educational proposals has also been common, in disregard of the actual educational practice they all performed to render context to their thinking (*e.g.*, for comments regarding common criticism to John Dewey's philosophy, *vd.* Westbrook, 1993, pp. 10-11).

The first such recurrent theme is the relevance given to **children's education by means of creative activities**, which should be **integrated in the overall context of the society** in which children live. Already in 1897 John Dewey had said (my emphasis):

> *« I believe that the only way to make the child conscious of his social heritage is to enable him to perform those fundamental types of activity which makes civilization what it is. I believe, therefore, in the so-called expressive or constructive activities as the center of correlation. I believe that this gives the standard for the place of cooking, sewing, manual training, etc., in the school. I believe that they are not special studies which are to be introduced over and above a lot of others in the way of relaxation or relief, or as additional accomplishments. I believe rather that they represent, as types, fundamental forms of social activity; and that* ***it is possible and desirable that the child's introduction into the more formal subjects of the curriculum be through the medium of these activities****. »*
>
> *(John Dewey, 1897, Article III)*

Figure 231 – William Heard Kilpatrick
1871 - 1965
From: http://www.tc.columbia.edu/centers/coce/photoGallery/images/Kilpatrick2.jpg

Dewey and the other thinkers presented here did not see such activities as an alternative curriculum for education. Rather, as I have emphasized in the above quote, **creative activities are a medium for children to explore and comprehend formal subjects**.

Soon after, the American philosopher William Kilpatrick (Figure 231, on the right), one of Dewey's students, proposed a pedagogic method that shared these views, and is now quite popular: the **project method**. Simply put, this method proposes that children set up to do specific, focused projects, during which various subjects can be approached, explored, and studied (Beyer, 1997). But rather than a mere pedagogic trick, Kipatrick saw this as a crucial strategy to provide purpose in learning, eliciting interest and effort, as I'll soon mention.

Figure 232 – Célestin Freinet
1896 - 1966
From: http://www.freinet.com.br/images/Freinet.jpg

Later, during the 1920s and 1930s, Célestin Freinet also embraced similar proposals. He would take his classes of primary school children out of the school "*in search of life in the rich*

environment of the nearby countryside and among the rural crafts still practised [in the region where his school was located]" (Legrand, 1993, p. 3). From these trips, back in the school, he conducted what would nowadays be seen as regular follow-up activities: discuss impressions, write them down, correct the texts, and improve them.

Besides reading and writing, similar approaches would be taken for arithmetic, science, history, geography, and art. The source of inspiration could be a trip, as I mentioned, but also the recreation of typical activities from the children's society. Freinet called this approach as making subjects "come to life" (*id.*, p. 6). For instance, "*measuring of length, volume, weight, problems arising in the feeding or rabbits and poultry, the purchase of seeds, the sale of field and garden crops*" (*id.*, *ibid.*) – not as fictitious situations, which might be useful, but as actual problems arising from activities in the school garden of farmyard. This is strikingly similar to the much early educational experiment of John Dewey's laboratory school, where the "*youngest children in the school, who were 4 and 5 years old, engaged in activities familiar to them from their homes and neighbourhoods: cooking, sewing and carpentry. The 6-year-olds built a farm out of blocks, planted wheat and cotton, and processed and transported their crop to market. The 7-year-olds studied prehistoric life in caves of their own devising while their 8-year-old neighbours focused their attention on the work of the sea-faring Phoenicians, on Robinson Crusoe and adventurers, like Marco Polo, Magellan and Columbus*" (Westbrook, 1993, p. 6).

Another recurrent theme is that of the **necessity of changes in the institutional framework as a prerequisite for application of these methods**. Among these three thinkers, Freinet moved the farthest along this effort. He proposed specific adaptations to the physical environment of schools, either for building them from scratch or adapting "*existing premises*" (Legrand, 1993, p. 7). He detailed the specific organization of the educational space in separate indoor workshops, plus a school garden and school farmyard. Being mostly a practitioner, he also detailed many other aspects of his educational practice, in order for other teachers to be able to start employing his ideas as fast as possible. In order to reach the educators at large and also the widest possible public, he organized a teacher's co-operative, the *Coopérative de l'enseignement laïc* (Secular Education Co-operative), for "*production of teaching materials and the publication of works on education*" (*id.*, p. 2).

To Freinet's credit, his political efforts have outlived him: currently, several dozen national institutions exist throughout the world, devoted to the promotion of Freinet's pedagogy (FIMEM, n.d.). The efforts of John Dewey haven't met a similar fate. In 1904, Dewey lost a power struggle for the control of his laboratory school and was never to have another. During his academic career, he realized that changes to education could not be achieved solely from within school, but rather in a broad political approach of society. He collaborated with educational reform efforts in several countries, besides his home USA. However, his efforts were mostly ineffective:

> *« Dewey's* [proposed] *'way out of educational confusion'* (...) *posed too great a threat to traditional methods and subject-matter. At the same time, its social implications were too radical for advocates of scientific efficiency, and not radical enough for some proponents of social reconstruction.* (...) *Thus,* (...) *'his intellectual stature, his international reputation and his many honours notwithstanding, Dewey did not have enough of a true following in the world of educational practice to make his impact felt'* (...) *»*
>
> *(Westbrook, 1993, p. 10)*

Similarly, Kilpatrick's impact had no major lasting consequence at the institutional level; but his project method, which is only an instrumental part of his philosophy, did eventually acquire significant popularity worldwide (Knoll, 1997), as can be deduced from the chart in Figure 233. Currently, his contributions are not felt at this institutional level, but "*Kilpatrick's personal love for teaching and the depth and popularity of his ideas concerning educational matters continue to be worth pondering*" (Beyer, 1997, p. 9).

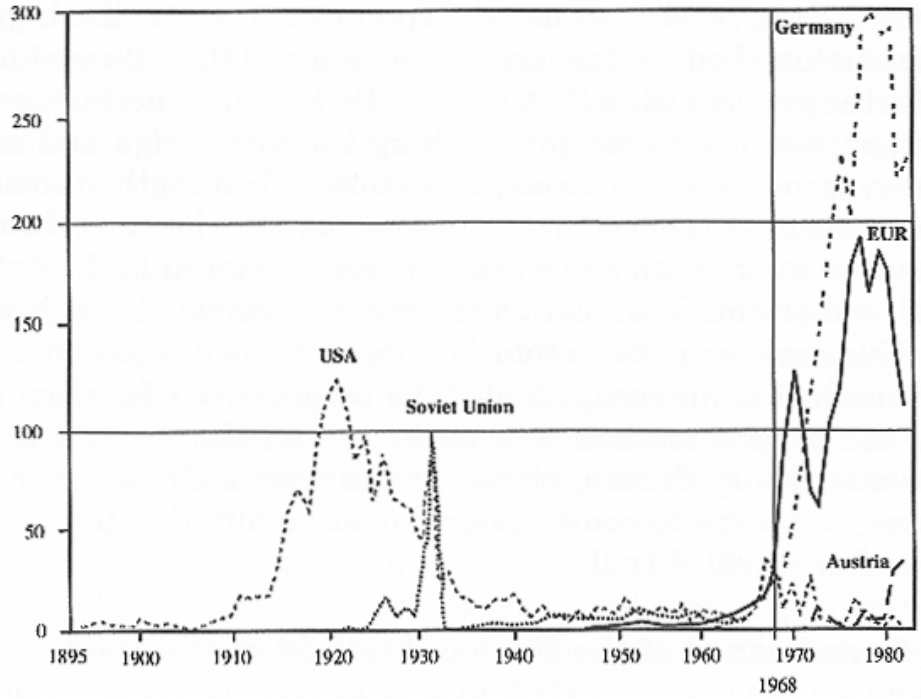


Figure 233 - Number of annual publications on the project method in selected countries and regions[251], 1895-1982

From: http://scholar.lib.vt.edu/ejournals/JITE/v34n3/images/Image5.gif (Knoll, 1997)

Lastly, three other concerns are quite similar between the three approaches of these educators. These are the preoccupation with the usage of **democratic principles in education**, which these thinkers linked to the importance of **developing the learners' motivation, by basing education on each individual's interests**.

> *« Individuals, Dewey argued, achieved self-realization by utilizing their peculiar talents to contribute to the well-being of their community, and hence the critical task of education in a democratic society was to help children develop the character, the habits and virtues, that would enable them to achieve self-realization.* (...) *Dewey argued that in order for a school to foster social spirit and develop democratic character in children, it had to be organized as a co-operative community. In order to educate for democracy the school had to become 'an institution in which the child is, for the time, to live — to be a member of a community life in which he feels that he participates, and to which he contributes' (Dewey, 1895, p. 224). »*
>
> *(Westbrook, 1993, p. 5)*

As the above quote indicates, Dewey saw the self-realization of children as achievable by allowing them to participate in the development of the educational process. This must be seen in the light of traditional schooling, where children would be ordered to open a book on a given page and all children were to read the same text. But Dewey also meant it in a more profound way than simply providing individual attention to each student, as can be seen by his "co-operative community" concept, mentioned at the end of the above citation. This can be mistaken for an entirely children-centric perspective, but that is not entirely so; Dewey believed that the teacher had an essential role in promoting and developing this participation:

> *« I believe that the child should be stimulated and controlled in his work through the life of the community. I believe that under existing conditions far too much of the stimulus and control proceeds from the teacher, because of*

[251] "EUR" represents Netherlands, Great Britain, Denmark, Sweden, Belgium, Switzerland, and Norway.

> *neglect of the idea of the school as a form of social life. I believe that the teacher's place and work in the school is to be interpreted from this same basis.* ***The teacher*** *is not in the school to impose certain ideas or to form certain habits in the child, but* ***is there as a member of the community to select the influences which shall affect the child and to assist him in properly responding to these influences.*** *I believe that the* ***discipline of the school should proceed from the life of the school as a whole*** *and not directly from the teacher. I believe that the* ***teacher's business is simply to determin****e on the basis of larger experience and riper wisdom,* ***how the discipline of life shall come to the child****. »*
>
> *(Dewey, 1897)*

One should notice that, even though in the above text Dewey employs words such as "simply" to address the role of the teacher, this role as described by him is nonetheless nothing short of essential. But his description of change in teacher's roles from a quasi-military starting position may be misleading, potentially allowing an interpretation of his thought as one of lack of self-discipline and permissiveness. But discipline is a constant care of Dewey; it's simply that it is a discipline of a democratic nature, where all must contribute and every contribution is valued.

Kilpatrick followed much in Dewey's stead, believing that by adopting democratic principles, a community – even the school community – "*encourages its members to become participants in civic discussions that require concerted, collaborative actions in the name of social justice and structural change.* (...) *children are people who are, and ought to be, actively engaged in attempts to understand and become more skilful in the world in which they live*" (Beyer, 1997, p. 11). The reasoning behind his project method requires each project to be agreed upon between teacher and student: the crucial point is that "*there be some dominating purpose—which of course may not be observable—in which students wholeheartedly participate*" (*id.*, p. 8). This way, according to Kilpatrick, there is an unification between student's interests and their actions, in the course of project development: having an inner "want" or desire leads to the creation of an "aim" or "goal", which people effort themselves to achieve. There should therefore be an educational effort to create interest, leading to effort, leading again to interest, a "*life process*" Kilpatrick calls the "*true unit of study*", "*the organism-in-active-interaction-with-the-environment*" (Kilpatrick, 1951, p. 14).

Freinet's educational ideas were not specifically addressing the teaching of democracy, but in effect lead to the same principles: "*The lack of genuine communication effectively condemns working-class pupils to intellectual banishment from a universe of language that is totally alien to them. This results in under-achievement and dropping out, and in particular accounts for the anti-democratic manner in which schools operate. Here too Freinet was a pioneer, to the extent that his ideas on language teaching called for genuine communication, i.e. for personal expression and a sympathetic ear*" (Legrand, 1993, p. 11). This attention to children was also present in Freinet's attention to the personal drives and motivations as essential to learning:

> *«* (...) *for him the essence of his techniques is 'experimental trial and error'. Schools, of course, are there for learning, but the learning process is something that cannot be imposed from outside* (...)*. The essential input must come from the learner himself or herself. The urge to know is aroused by the obstacles encountered, by gaps in the evidence, by failure to understand and by searching for what will make understanding possible. To be effective, this search must be spontaneous, actuated by the internal need of the sector, and there will inevitably be mistakes along the way. It is by feeling their way, by trying first one approach then another, that the child and the adult achieve real learning »*
>
> *(Legrand, 1993, pp. 9-10)*

All the themes presented so far, which are common to the three thinkers presented in this section, fall within a similarly shared global educational purpose: that of **enabling children with mind skills that enable them to more easily adapt to unforeseen future requirements**. Dewey, for instance, expressed it in the following manner:

> *« With the advent of democracy and modern industrial conditions, it is impossible to foretell definitely just what civilization will be twenty years from now. Hence **it is impossible to prepare the child for any precise set of conditions. To prepare him for the future life means to give him command of himself**; it means so to train him that he will have the full and ready use of all his capacities; that his eye and ear and hand may be tools ready to command, that his judgement may be capable of grasping the conditions under which it has to work, and the executive forces be trained to act economically and efficiently. It is impossible to read this sort of adjustment save as constant regard is had to the individual's own powers, tastes, and interests* (...). *»*
>
> *(Dewey, 1897, my emphasis)*

Or, in the context of Freinet practice: "(…) *what is essential in the New Education*[252] *is the desire, not of novelty, but of progress, of constant adaptation to a world in accelerated evolution* (…)" (Gal, 1966, p. 61).

Kilpatrick, in this regard, was particularly concerned with ability for teachers and students to be able to participate in a democratic society, judging different points of view. Students should not just be able to perform as they were taught, and teachers should be able to reason over their own actions and constantly refine a set of personal philosophical principles to help them drive their practice, in constant improvement of personal realization of the consequences of their actions, "*beliefs and assumptions, ideals and convictions*" (Beyer, 1997, p. 10).

> *« Democracy wishes all the people to be both able and willing to judge wisely for themselves and for the common good as to the policies to be approved; it will accordingly seek a type of education to build responsible, thinking, public-spirited citizenship in all its people. »*
>
> *(Kilpatrick, 1951, p. 5)*

I will now ensue with the descriptions of three actual pedagogic models, all based on the overall goal of supporting child development, under the constructivist theory, albeit from different approaches.

[252] Freinet's pedagogic approach is usually categorized as part of the French "New Education" tradition (Gal, 1966, p. 37), along with Maria Montessori, who was mentioned in p. 247. This citation was translated from the Portuguese version.

High/Scope

This model is strongly based on the child development principles of Jean Piaget (*vd.* p. 257). It evolved from a 1962 preschool project by American psychologist David Weikart (experienced in the education of children with special educational needs), at the Ypsilanti school district, in Michigan, USA (Formosinho, 1998, p. 56). This preschool project, known as the *Ypsilanti Perry Pre-School Project* (*id.*, *ibid.*), eventually led to the development of what became known as the High/Scope curriculum, in 1970 (*id.*, p. 59; Epstein, 2003; High/Scope Educational Research Foundation, 2005). From these origins, the High/Scope model evolved into its current form.

Figure 234 – David P. Weikart
1931 - 2003
From: http://lifeinlegacy.com/2003/1220/WeikartDavid.jpg

From its theoretical background in Piaget theory, the High/Scope model applies it in practice through the activity-based learning style, in the Dewey and Kilpatrick tradition (*vd.* p. 266). Specifically, it focuses on providing each child with "key experiences" for his/her cognitive development, as is described below.

Space and materials

The entire educational environment (*e.g.*, the physical organization of the preschool room) must be "*carefully chosen and arranged to promote active learning*" (Epstein, 2003). The children should have access to several different spaces and materials, and be able to find, use, and return them independently (*id.*, *ibid.*). This is achieved by having distinct activity areas within a room, to allow diversity of curricular approaches, for instance: home area, constructions area, library area, etc. These do not follow a strict model, but are instead supposed to be organized several times in the course of the day-to-day educational events. For instance, a trip to the local post office may result in children wanting to create a post office area (Formosinho, 1998, pp. 68-69).

A specific feature of this model is a set of 58 "key experiences". These are "*social, intellectual, and physical experiences* (...) *organized into ten categories that comprise social development (initiative and social relations), visual and performing arts (creative representation, movement, and music), reading (language and literacy), and math and science (number, classification, seriation*[253]*, space, and time)*" (Epstein, 2003). These experiences are a constant reference for the High/Scope teachers in setting up the environment and planning, and are also the basis of a Child Observation Record, which is the main assessment tool of the model (Formosinho, 1998, p. 60; Epstein, 2003).

Time management

The daily structure of the High/Scope model is based on a "*plan-do-review*" sequence (Epstein, 2003). The activities themselves are not pre-structured (dependent on the contributions of adults), but they are also not totally random or totally improvised (dependent on children's bursts of energy). The High/Scope model aims to combine both situations, through the plan-do-review sequence. So, the adult must in his/her activity plan beforehand, right from the organization of the educational environment itself, from the anticipation of likely events and the thinking through of how time must be managed, proceeding from this planning to its execution with the children and to its review. The time management aim of the High/Scope model is that children are aware of an underlying structure that provides confidence by being known and predictable. It is to be flexible when needed, but not random, so that a child knows what to expect (Formosinho, 1998, pp. 70-72).

[253] "Seriation" is the ability to arrange things in a logical progression. For instance, putting the smaller objects in front of the larger ones, creating a row with the objects in order from the oldest to the newst, etc.

Children also follow the "plan-do-review" sequence, by making plans, carrying them out, and finally reflecting upon the results:

> *« (...) the plan-do-review sequence supports actions initiated by the pre-schooler. Adults encourage children to plan – to express their intentions – by asking, for example, 'What do you think you will do with all those blocks?' Children then carry out their intentions, or plans; and this* ***do*** *stage may last a few minutes or more than an hour. Dramatic play involving several children, 'cooking' to 'feed' fellow playmates, or creating shell structures and patterns are the types of things that occur during the* ***do*** *stage. After each work period, adults encourage children to* ***review*** *their experiences. The children may talk about what they did or express themselves in drawing or in 'writing'. In so doing, children begin to exercise memory skills and develop insight about their experiences. »*
>
> *(Weikart, 2000)*

Adult intervention

In the High/Scope model, the adult intervention closely follows from Piaget's views, particularly as seen in opposition to maturational and behaviorism theories (*vd.* section 4.1.2). Under this view, each child is developing from his/her own world experiences, assimilating them, finding contradictions and seeking balance. So it follows that the role of adults is one of creating situations to challenge the child's current thought, creating conflicts in his/her understanding (Formosinho, 1998, p. 73). The child's personal commitment to achieving solutions for such conflicts is the driving force of development (as seen above, the environment itself, planned by the adult, is also meant to allow children to experience different learning situations).

Adults must also, therefore, provide encouragement, independence and creativity in children's approaches to the various situations that arise, assisting children in the resolution of cognitive conflicts (Epstein, 2003).

Reggio Emilia

This model takes its name after the city of Reggio Emilia, in Northern Italy (Figure 235), where it developed. Its origins, documented by journalist and teacher Loris Malaguzzi, lie in the joint creation by several citizens[254] of a young children school, six days after the end in Europe of the Second World War, in the Spring of 1945, in the small village of Villa Cella, near the city of Reggio Emilia (Malaguzzi, 1998). Several similar schools were created, and for several years the various teachers strove to make the citizen's efforts worthwhile. Eventually, in 1963, these previous efforts led to the creation of municipality-supported schools.

Figure 235 – Reggio Emilia location and Loris Malaguzzi (1920-1994)
Photo from: Spaggiari & Rinaldi, 2000, p. 212

> *« We were now involved in a serious endeavor.* (...) *We received the first expert group of teachers from the parent-run schools. Responsibilities were clear in our minds; many eyes, not all friendly, were watching us. We had to make as few errors as possible; we had to find our cultural identity quickly, make ourselves known, and win trust and respect. I remember that, after a few months, the need to make ourselves known became so strong that we planned a most successful activity. Once a week we would transport the school to town. Literally, we would pack ourselves, the children, and our tools into a truck and we would teach school and show exhibits in the open air, in the square, in public parks, or under the colonnade of the municipal theater. The children were happy. The people saw; they were surprised and they asked questions. »*
>
> *(Malaguzzi, 1998)*

Since these beginnings, the Reggio Emilia model has been developing, from the efforts of the various teachers involved, within this strong background linkage with the local society. Several philosophical and scientific influences have been integrated into the model, such as those of Dewey and Freinet (presented earlier in this section), Piaget, Vygotsky, Erikson and Bronfenbrenner (presented in section 4.1.2), Montessori (presented in section 4.1.1), and others, upon debate and exchange of ideas with Italian, French, and Swiss early childhood educational professionals (Lino, 1998, p. 97; Malaguzzi, 1998).

Regarding Piaget's influence, the teachers behind this model back his fundamental view of children actively developing their concepts of the world, and using a teaching style that provides each child with learning opportunities. But they did not support some of Piaget's views, such as his separation of cognitive, affective, and moral development; the little attention he devoted to

[254] "*I found women intent upon salvaging and washing pieces of brick. The people had gotten together and had decided that the money to begin the construction would come from the sale of an abandoned war tank, a few trucks, and some horses left behind by the retreating Germans. 'The rest will come,' they said to me. 'I am a teacher,' I said. 'Good,' they said, 'If that is true, come work with us.' It all seemed unbelievable: the idea, the school, the inventory consisting of a tank, a few trucks, and horses. They explained everything to me: 'We will build the school on our own, working at night and on Sundays. The land has been donated by a farmer; the bricks and beams will be salvaged from bombed houses; the sand will come from the river; the work will be volunteered by all of us.' 'And the money to run the school?' A moment of embarrassment and then they said, 'We will find it.' Women, men, young people - all farmers and workers, all special people who had survived a hundred war horrors – they were all dead serious. Within 8 months, the school and our friendship had set down roots*" (Malaguzzi, 1998).

memorization and social interactions, and the little importance he gave to the role of adults, among other criticism. Social and cultural relationships are central to the Reggio Emilia model (*id.*, p. 98).

Space and materials

The educational space is meant to reflect the ideas, values, attitudes and cultural background of all the people that use it. This organization is used so as to allow children to communicate, cooperate, and share activities and ideas, building their conceptions of the world, and is to be thought-out by all participants in the educational process: teachers, supporting personnel, parents, and children.

Figure 236 – Lunchroom, Diana school
From: Spaggiari & Rinaldi, 2000, p. 44

The typical organization of a Reggio Emilia preschool comprises three activity rooms for children each with their own mini-workshop. The three activity rooms share a common central space (the *piazza*), a school workshop, a library with computers, an archive, a lunchroom, and several restrooms. All these common spaces (including the restrooms) are used for common activities involving the school's children and adults (Lino, 1998, p. 107).

Within the activity rooms themselves, one finds a division into different activity areas, an organization similar to the one previously described in the High/Scope model. The archive is used to store mail, news, and documentation on children's works and experiences, available whenever necessary. The library with computers is also similarly available for research by children. But the most distinctive space element of the Reggio Emilia model is the school workshop. This is not a technical workshop, but an artist's workshop, and has a permanent staff member, a specialized art educator (New, 2000), with specific training in the use and exploitation of artistic techniques.

Figure 237 – The "piazza", Diana school
From: Spaggiari & Rinaldi, 2000, p. 45

The central common space – the *piazza* – is used how one would expect, to welcome parents and visitors, and is a foremost place for children to display their work; but it is also used for shared activities. For instance, materials for making disguises, kaleidoscopes, etc. – are all kept in the *piazza*. Regular shared activities, such as holding a bi-weekly marketplace, stalls and all, also take place there (Lino, 1998, p. 108 & 111). Another element of crucial relevance to the model is the exterior playground space. The activities in the Reggio Emilia model typically involve many children in a common project, often requiring enough space for the development of large structures, *e.g.*, an enormous dinosaur (*id.*, p. 109).

Children use traditional educational materials, as one would expect, but the contact with various forms of personal artistic expression is central to the model's activities. Art is **not** the focus of the schools, one must realize. It is extensively used in order to empower children with many different forms of expression – expressive languages – allowing them to work upon a theme or

concept from different perspectives, and therefore to develop and increase their understanding of it. That is, children are encouraged to represent one same idea in various forms, such as felt pens, pencils, gouache painting, water painting, clay modeling, wood, cardboard and paper constructions, etc. (Lino, 1998, p. 102). It is at this level that the support of the art educator becomes crucial: he/she helps children understand the various techniques, but also to develop their critical appreciation of their work. The other teachers also benefit from the knowledgeable assistance of the art educator. The productions are debated together with all children, they're photographed, what children say is transcribed, and all the resulting information is shared with all children in the school, all adults working in the school, and with parents and families (*id.*, *ibid.*).

Time management

The Reggio Emilia model does not have a structural time pattern. The common element is that early in the morning (at about 9 AM), once all children have entered the preschool (they start to arrive by 8 AM), there is a planning meeting at each activity room, between the children of that room and both their teachers. At these meetings, children will elect to start specific projects, develop existing projects or conduct other activities (Lino, 1998, p. 112).

During the day, the children are organized into several groups, each group devoted to some specific project or activity, under the support of adults (mostly, the teachers and art educator; sometimes, parents are involved, and also other school professionals – the cook, for instance).

Throughout the day, children can experience various kinds of interactions, in several areas, and using a diversified range of materials. Several times in the day, before and after any work (in groups or alone), children and adults gather (sometimes all the children at the same time), to talk, share discoveries, difficulties, and problems, and plan activities (*id.*, pp. 112-113).

Adult intervention

At each preschool room, two teachers are involved, in close cooperation with the art educator based at the school workshop. As mentioned at the end of the "space and materials" section, above, the art educator cooperates with the children and the teachers, helping them to better use and understand the expressive possibilities given by each artistic form and medium.

The teacher's role is described as "listening first" – a listening pedagogy (Lino, 1998, p. 101). Behind this simple idea are many others: to listen to someone requires attention to what is being said. By listening, the speaker is legitimized; he/she is no longer anonymous. Being listened to also changes the speaker (*id.*, *ibid.*). From this listening, coupled with observations, documentation, and reflection with other adults, the teachers in each room "*serve as resources and guides to the children*" (Edwards, 2002), organizing the children's efforts under common focus points, *i.e.* projects that involve various children and adults (Lino, 1998, p. 122-126).

But that is not all: by collecting documentation, the teachers are also doing it not only for their own assessment, but they are also doing it as a documentation service to the children. The children's works, documented by their teachers, are a basis from which to establish a dialogue with parents and relatives. From this dialogue, further contact between parents and the school is established, under the organization of the teachers. This can involve common meetings of parents, but also labor sessions to improve the school, by building furniture or collecting material for the children to use, and workshops to introduce parents to the expressive techniques employed with children.

Portuguese Modern School Movement

The pedagogy of Célestin Freinet, presented at the beginning of this section (alongside Dewey's and Kilpatrick's), was supported and promoted by a teachers co-operative. This promotion effort eventually led to the worldwide creation of similar teachers associations, for national promotion of Freinet's "Modern School" pedagogy in their countries (for an up-to-date list, see FIMEM, n.d.). Being co-operative efforts, developing through the contributions of their members, these national associations possess a specific development history, albeit maintaing the core ideas originally proposed by Freinet. The evolution of Portuguese Modern School Movement, since its creation in 1966, has led it to integrate educational ideas of Vygotsky and Bruner (*vd.* section 4.1.2, pp. 260-262). It thus has moved from a central focus on each child's personal experimentation in various situations, towards an education more focused on social and cultural interactions, with peer and adult support (Niza, 1998, p. 139).

In this model, children are not grouped by age, so as to ensure that each grouping of children may possess as much cultural and personal variation as possible. In Freinet's stead, children are to freely express their views and the school should valor their life experiences, opinions and ideas. From children's interactions with ideas, materials, documentation, or other people, moments of doubt and questioning will inevitably arise, and these are to be used as the source for children's research projects. Such projects, found at the heart of Kilpatrick's pedagogy (as described earlier in this section), are also at the heart of the pedagogy of the Portuguese Modern School Movement (*id.*, p. 146).

Space and materials

Each school room is divided into workshops, under Freinet's original division (Legrand, 1993): library & documentation area; writing & reproduction workshop; sciences & experiments lab; carpentry & construction space; plastic arts workshop; toys, games & make-believe space; a shared multifunctional space for collective enterprises. In schools that do not possess a lunchroom, or where it is not accessible to children, the school room also has an area for food culture & food education. Unlike the High/Scope model (p. 271), whose internal room spaces are flexible, constructed and dismantled in the course of educational activities, the room division in the Modern School model is predetermined and permanent (Niza, 1998, pp. 146-147). However, in contrast to this rigidity of the model, as it is described by Sérgio Niza, who is one of its main Portuguese proponents, the actual field practice of preschool teachers is in effect flexible (*e.g.*, Vasconcelos, 1997, pp. 94-99; Teixeira *et al.*, in press).

In these workshops, children employ the materials one would expect in preschools, with some exceptions. In the Modern School model, the educational spaces aim to be as close as possible to the original, inspirational social spaces. In all workshops (except in the toys, games & make-believe space), authentic materials are used – real wood in the carpentry space, a real computer plus printer (or a typewriter) in the writing & reproduction workshop, etc. A consequence of this view is that miniatures are seen as infantilizing and avoided; each area "*should reproduce* (...) *a study or workshop, in everything resembling the organizational environments of adult society*" (*id.*, p. 148).

The walls of the preschool room are continuously employed as a place for displaying the products of children's efforts. One of the walls is also used for posting up registration tables, in support of children's planning, management, and assessment of the educational activities in which they took part.

Time management

Under the Portuguese Modern School Movement model, the time organization is quite detailed. For each day, the morning is devoted to children's efforts, supported by their teacher, on the tasks or activities they selected, and the afternoon is devoted to group-wide sessions of information or cultural activity, led by guests, children, or the teacher.

A regular day would thus follow this pattern: 1) Welcoming; 2) Council planning; 3) Activities and projects; 4) Break; 5) Presentations of learning; 6) Lunch; 7) Recreational activities (songs, traditional games, etc.); 8) Collective cultural activity; 9) Overall assessment in council. Weekly, there is a half-day for field trips, usually Wednesdays (Niza, 1998, pp. 150-151).

In this organization, one can see a global pattern similar to High/Scope's plan-do-review sequence, but here it is organized at global level of the day's time, not within each specific task. There are specific guidelines for each part of the day, detailing which documents should be used, what children should record in registration tables, and so on (again, as previously stated above for space and materials, the actual field practice is flexible).

The 8th section of the day, "Collective cultural activity" also has a week-long schedule. On Monday, it is the "tale time", when the teacher and the children read a tale, and discuss it; on Tuesday, parents or other people come to talk about things that happen in their lives; on Wednesday, the recall and discussion of the morning's field trip; on Thursday, the time is open to the children's initiatives (compose mail, complete a journal, theatre play, etc.).

Figure 238 – MSM room journal
Photo by Margarida Teixeira, school year 2004-2005, Parada de Cunhos preschool, Vila Real, Portugal

On Friday, there is a special council meeting (also during the 8th time section), during which the teacher reads the columns of the room's journal (*vd.* Figure 238):

> *« Briefly, but solemnly, negative judgments are discussed (giving all implicated a chance to speak): positions are clarified, while avoiding the violence of a trial. People involved in positive judgments are applauded. One becomes aware of significant accomplishments, and suggestions are oriented, regarding commitments to be taken and actions to be scheduled on the following Monday or in another short-time occasion. »*
>
> *(Niza, 1998, p. 153)*

Adult intervention

The teacher's actions and planning regarding the children's learning should be supported on the methods that each scientific or cultural field developed throughout history (*id.*, p. 139). But rather than focus on isolated individual accomplishments, teachers should continuously develop the exchange of information and realizations among children.

> *«* (...) *teachers supporting this preschool education system come forward as promoters of the participated organization; stimulators of cooperation; civic and moral animators of democratic training; active auditors to provoke freedom of expression and a critical attitude. They maintain and stimulate the autonomy and responsibility of each learner* (...) *»*
>
> *(Niza, 1998, p. 155)*

Other adults, such as relatives, neighbors and local organizations are seen by the preschool as sources of knowledge and training. This view must be shared by those adults, and it is also the teacher's and the school's task to make them aware of their roles, by inviting them to participate in

specific sessions. These can be a gathering at the school, a personal visit to the school, a field trip, requests for documents to be employed by the children as part of their research, etc. The families and the community are also expected to participate in the school's organization, and to cooperate in this mediator role of the school (*id.*, pp. 155-156).

4.2. Computers in preschool

4.2.1. Non-programming use

> *« What we as early childhood educators are presently doing most often with computers is what research and NAEYC guidelines say we should be doing least often. »*
>
> *(Clements, 1994, p. 33, as cited by Haugland, 2000)*

In the above quote, Douglas Clements is referring to "most often" uses of computer in early childhood education; but actual field surveys on common practices tend to focus on the formal[255] levels of education, for children aged 6 or older (*e.g.*, Office of Technology Assessment, 1995).

> *« In most of the countries researched on, the studies have been limited to ICT[256] indicators used in the primary and secondary level. »*
>
> *(UNESCO, 2003)*

At the level of preschool education, field surveys documenting common uses of computers are rare. For instance, a literature review of the field, conducted in 2002, reported, regarding the UK (although a similar remark could have been made about other countries[257]):

> *"Although observations in pre-school playrooms suggest that a computer is commonly available for use during free play* (...) *a search for survey evidence about the provision of computers or programmable toys in pre-school settings in Scotland (or elsewhere in the UK) revealed no statistical evidence about current usage or playroom availability"*
>
> *(Plowman & Stephen, 2003, expanded version)*

Since then, however, some survey efforts on the use of ICT in preschools took place, albeit mostly focusing on the availability of hardware and software, rather than on educational practices (Kinderet[258], 2002b; Specht *et al.*, 2002; Leung, 2003; Lynch & Warner, 2004). Recently, in November 2004, another review study reported "*a current surge in research and writing on the use of ICT in early childhood education*" (Bolstad, 2004), the results of this surge being grouped in the following categories: "*(1) 'effects' research; (2) investigations of children's behaviour and interactions around computers; (3) research into children's experiences of ICT in early childhood education settings and at home; (4) research about practitioners' professional learning in, or through, ICT; and (5) case studies or exemplars of innovative use of ICT in early childhood education settings*" (*id.*, *ibid.*). It is telling that these categories don't include surveys on common uses of ICT – or even just computers – in preschool education settings. Assorted case studies, obviously, don't provide a picture on how common or representative is each case. Fortunately, two welcome contributions to the field have taken place: a nationwide survey on actual use of computers, conducted by the New Zealand Council For Educational Research, with the cooperation of "*242 early childhood managers or co-ordinators, and 402 early childhood teachers*" (*id.*, p. 62); and a pre-intervention survey on 117 preschool settings in six countries[259], as part of the research program KidSmart on the improvement of ICT use at this educational level (Siraj-Blatchford & Siraj-Blatchford, 2004).

---

[255] *I.e.*, "elementary", "primary", "basic", the actual nomenclature varying accross national educational systems.
[256] "Information and Communication Technology", which includes computers, but also several other types of equipment, such as digital cameras, cellphones, etc.
[257] Recently, the researchers involved in a project to establish the educational impact of teacher training on preschoolers' learning mentioned how that project provided "*hitherto unavailable data and inferences about preschoolers' cognitive, social and technological capabilities within a technology-enhanced environment*" (Swaminathan *et al.*, 2005).
[258] This survey study encompasses settings in Bulgaria, Portugal, Spain, Sweden, and the UK.
[259] France, Germany, Italy, Portugal, Spain and UK.

Thus, one is left with a patchy picture of the use of ICT in the practice, discerned from:

- the few localized surveys mentioned above (Kinderet, 2002b; Specht *et al.*, 2002; Leung, 2003; Lynch & Warner, 2004; Siraj-Blatchford & Siraj-Blatchford, 2004; Bolstad, 2004);
- general evaluations and situated research ("effects" research), provided by various professionals and researchers (*e.g.*, Clements, 1994, 1999a; Haugland, 2000; Klein *et al.*, 2000; Plowman & Stephen, 2003; Clements & Sarama, 2003; Paz, 2004);
- the numerous case studies, examples, and proposals for preschool computer activities found in the World Wide Web and traditional publications (*e.g.*, Davis & Shade, 1994; Pierce, 1994; Druin, 1996, pp. 77-81; Segers & Verhoeven, 2002; Gimbert & Cristol, 2003; Kent NGfL, n.d.; KidSmart, n.d.);
- the practice recommendations provided by public bodies and professional associations (*e.g.*, NAEYC, 1996; NETS, 1999; Learning and Teaching Scotland, 2003a);
- research studies regarding the subject matter included in professional training courses for early childhood teachers (Kinderet, 2002a; Coutinho, 2005; Swaminathan *et al.*, 2005);
- my personal contact with preschool teachers and computer-activity teachers for preschools (Morgado *et al.*, 2003b; Cruz *et al.*, in press).

I find this patchy overall picture to be well summarized by the following statements (originally made to describe the state of the practice in Sweden, in 2004/2005):

> *«* (...) *it is more common for the children to have learned how to handle the functions of a computer at home than in pre-school. In pre-school, the children use computers in other activities, that is, to do things, to play and to let time pass. It is extremely unusual for teachers to use ICT as a pedagogical challenge.* (...) *The ICT are not integrated in other pre-school activities unless the teacher has participated in special projects arranged by for example ITiS*[260]*. Pre-school teachers who have participated in competence development program* (...) *are the ones that, to some extent, use the computers in a pedagogical sense. More common is for these teachers to use ICT for documentation of various themes and/or to document children's learning process as well as for the children's own enjoyment. In those cases, a digital camera is used to take photos to be printed out through the computer or stored as digital documentation. Teachers who have not participated in competence development programs use ICT mostly to relieve the pressure and for the children to have something to do – to devote the children's time. These teachers are more or less just circling around helping the children with various functions and technical support – if they can! »*
>
> *(Kinderet, 2002a, p. 40)*

In other words, the typical computer use in practice, in preschool contexts, is far from achieving the recommendations published by various public bodies, such as those mentioned above, which were based on case studies and specific research on the potential benefits and challenges of computer use in early childhood education.

I therefore based my understanding of current non-programming uses of computers in preschool settings on these main reference sources: recommendations from public bodies (NAEYC, 1996; Ministério da Educação, 1997; NETS, 1999; NCTM, 2000; Learning and Teaching Scotland, 2003a; New Zealand Ministry of Education, 2005), analyses of the lectured contents of ICT courses in early childhood education curricula (Kinderet, 2002a; Coutinho, 2005) and research articles and

[260] "IT i skolan", a Swedish national action program for ICT use in Schools, which ran from 1999 to 2002 (ITiS, 2003).

reviews (Clements, 1999a; Haugland, 2000; Clements & Sarama, 2002, 2003; Plowman & Stephen, 2003; Paz, 2004; Bostad, 2004; Plowman & Stephen, 2005). A central research-based concept in all these sources is that "*simply providing ICT equipment to schools or teachers will not necessarily make a difference; what makes the difference is the way in which this equipment and other resources are used*" (Bostad, 2004, p. 4). That is, technology should be "*integrated into the regular learning environment and used as one of many options to support children's learning*" (NAEYC, 1996, p. 2).

Table 25 summarizes the evolution and development of the field and provides a framework for situating the various educational practices on the use of computers (and ICT in general).

	Physical and technical arrangements	**Role of children and adults**	**Scaffolding of children's learning**
A low level of quality ("isolation")	*Only one computer is available for children to use, at the teacher's discretion. Only a few software programs are available, the software is unconnected with the current classroom themes and topics. The child operating the computer has his or her back to the other children and is not involved in their activities.*	*Children seldom use the computer, nor do teachers encourage its use. Teachers often take a controlling and instructing role, partly to ensure that all children have equal opportunities to use the computer.*	*Teachers stop engaging themselves once children are self-sufficient and have learned basic ICT skills.*
A good level of quality ("integration")	*The computer is relocated into a more central position among other classroom activities. Computers and other ICT equipment (such as digital cameras) are available for children to use. A range of software programs is available, including pedagogical programs, creativity/multimedia programs, and games.*	*Sitting together in front of a computer, children help each other, negotiate turn-taking, collaborate, and tutor each other. Children communicate, discuss strategies, solve problems, and have fun together while they use games and educational programs. Children develop different strategies while learning to handle the computer and/or different programs. They ask friends, experiment, guess, move the mouse aimlessly, use help functions, and explore by themselves or with friends. Teachers encourage children to send email, use the Internet for information, and write or illustrate, or lay down soundtracks and narration for their own stories on the computer.*	*The computer is still not an integrated part of other activities in the preschool. Its uses can be described as learning by doing various activities on the computer, compared to learning through the computer.*
A high level of quality ("immersion")	*Children use computers and ICT equipment throughout the day as a multifunctional tool that is integrated with other activities and themes. Children learn through the computer and from each other while using a variety of programs or creating their own.*	*Children explore new topics, are creative in their search for information, ask questions, and express their reflections and feeling. Practitioners and children use computers to document children's activities, make labels and signs as needed, and send messages. Parents can access information while in the setting.*	*Teachers interact with and guide the children. They create possibilities in which ICT can be used to support children in developing new experiences and to expand their world.*

Table 25 - Levels of quality of ICT use in an early childhood education setting

From: Bostad, 2004, p. 41

Even though this table does mention the use of ICT, not just computer, the examples it provides are focused on the use of computers within the preschool classroom. But children are immersed in human society, not just the classroom; and as described in section 4.1.3, modern preschool education models put considerable importance into children's relationship with the overall society. Under this perspective, the recent explosion in the use of the computer as a communication medium holds great potential for preschool education. It can improve children's and teachers' contact with places, organizations, and people located far from the preschool room, and even allow for better coordination of activities between various teachers[261] and various preschools.

In section 7.2, I provide a hands-on technical perspective on the matter of immersion of the computer in the daily practice of preschools, in the form of a four-step guide, including the use of computer programming in this manner.

[261] The use of computers in this fashion, with teachers, has been frequently associated with the notion of "communities of practice" of Jean Lave and Etienne Wenger (*vd.* section 4.1.2, p. 256).

4.2.2. Seymour Papert and constructionism

So far in this chapter, I have provided an historical background on the development of preschool and kindergarten education (4.1.1), and the theoretical background behind current practice at these levels (4.1.2). I have also presented several influential examples of how reflection and practice have been employed to create and develop actual everyday pedagogic environments in preschool education (4.1.3). Finally, in the previous section (4.2.1), I addressed the current situation on the use of computers (but not computer programming) in preschools and kindergartens.

I will now at last address the use of computer programming proper, in preschool and kindergarten. The next section (4.2.3) presents the background on the use and research about computer programming involving children aged 3, 4 or 5 years old. And the following chapters (1, 1, and 1) present my field work on this matter and results deriving from it.

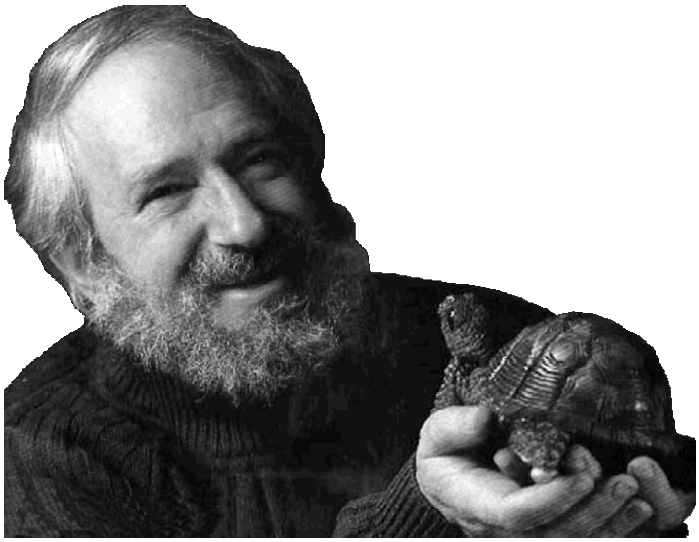


Figure 239 – Seymour Papert
From: Papert, 1999, p. iv (edited)

The present section is therefore at an apex in this thesis. The presentation of the computational and educational backgrounds is almost complete, and the presentation of the field work is about to begin. For this reason, this section builds on the previous ones by being concerned with the theoretical support for the use of computer programming in education, a theme already approached in section 2.4.5, albeit then merely in the scope of an overview to the field of computer-use in education (not just computer programming). For contextual reasons, that section only provided an historical overview and the major highlights of this theme, which this section provides in more detail. Its central intellectual underpinnings are inextricably connected to the work and thought of the South African mathematician, computer scientist and epistemologist Seymour Papert (Figure 239) – hence the section's title.

As described in section 2.4.5, Papert worked with Jean Piaget in Geneva, Switzerland, who immensely inspired him[262], centering his interests on children's cognitive processes (and while still in Geneva he started to get interested in the impact of computer cultures in psychologist's methods for thinking about thinking[263]). His computer science work at MIT's Artificial Intelligence Laboratory further involved him with researchers and developers in various fields: creating thinking machines, understanding human cognition, developing programming languages, and educational computing. A crucial development within all this personal background and efforts was the development of the Logo programming language[264] (as described in section 2.4.5, pp. 92-93), and from then on a lifelong contact and embracement of the issues involving computer programming by children and its cognitive potential. This lifework has produced several of the most crucial insights on the role of computer programming in cognition and education.

What is the distinctive nature of Papert's educational contributions? From a summarized point of view, Papert sees himself as a follower of constructivist theory (described in section 4.1.2), which is the prevalent influence behind most modern pedagogic models of preschool and kindergarten. His educational thought has therefore plenty in common with previous thinkers,

[262] *Vd.* footnote 53, on p. 92. Piaget's educational thought is presented in section 4.1.2, p. 257.
[263] Papert, 1980, p. 23.
[264] The Logo language is described at length in section 3.3.4, pp. 139-146.

particularly those mentioned in section 4.1.3 (Dewey, Kilpatrick, and Freinet)[265]. Papert's contributions to general educational theory (which I'll describe shortly), are therefore not absolutely new, but rather novel developments, building on earlier ideas.

To say this is not to diminish Papert's contributions in any way, quite the contrary; it's just that in my view, his absolutely visionary contribution was the idea that computer programming, if wisely used, had the potential to radically change the thinking style of humankind, in a way comparable to the invention of writing:

> *"*[computer programming is] *a **means** that can, **in principle**, be used by educators to **support the development** of new ways of thinking and learning."*
>
> *(Papert, 1980, p. xiv, emphasis from the original)*

Before pursuing this discussion, I may add that proposing that a medium of expression can significantly impact the development of ways of thinking and learning is, in itself, not new: as I described in section 4.1.2, Vygotsky and his colleagues (and many researchers since) have explored the role of language on the development of the human mind and cognition – to which recent neurology research on the mental development of seriously-neglected children provides support (Perry & Pollard, 1997; Perry, 2001, 2002); also from neurology research one finds evidence of the impact of reading and writing on the development of mental processes, as I described on p. 85.

The basis for an assertion as strong as the development of new ways of thinking and learning came from Papert's realization that **computer programming could allow one to establish different, richer relationships with knowledge**, by **fostering awareness of one's own learning processes** (Papert, 1980, p. 177).

This claim comes from the way in which programming provides people with a lens and language to view novel concepts and communicate them. **When using programming, a person is not just creating something, but defining a process for creating**. Beyond communication with others by means of opinions, goals, events, accomplishments, and other static notions, the individual experienced in computer programming is thus empowered with a means to consider the very processes for doing, analyze them, discuss them, and describe them.

> *« Although the work at the computer is usually private it increases the children's desire for interaction. These children want to get together with others engaged in similar activities because they have a lot to talk about. And what they have to say is not limited to talking about their product: LOGO is designed to make it easy to tell about the process of making them.*
>
> *By building LOGO in such a way that structured thinking becomes powerful thinking, we convey a cognitive style, one aspect of which is to facilitate talking about the process of thinking. »*
>
> *(Papert, 1980, p. 180)*

It's perhaps difficult to appreciate the significance and actual content of these statements, without having experienced, first-hand, the mental processes behind computer programming. For instance, after learning how to read, one can no longer look at words without reading them; the urban environment and cultures speak to the reader with a torrent of information unlike any experience that a non-reader can have in a similar location. Words cease to be abstract visual patterns, requiring conscientious decoding when necessary, to become part of a reader's worldview[266]. A similar argument has been presented earlier in this thesis, regarding the worldview-

[265] As described in section 4.1.3, Kilpatrick, for instance, mentioned that children should be "*actively engaged in attempts to understand and become more skilful in the world in which they live*" and "*wholeheartedly*" participating in the "life process" of learning (Beyer, 1997, p. 7, 11, & 14).

[266] Literate adults can experience a partly preliterate worldview by traveling to a place with a different alphabet.

shaping effect of metaphors and culture (*vd.* section 3.3.3, particularly pp. 130-131). It is to such an extent that Papert places the potential of computer programming.

Obviously, people can communicate about what they do, but they do not necessarily have the vocabulary to be proficient at it. In fact, one doesn't have to look beyond everyday events to realize how often people struggle with the simple interpretation of procedures, be they the instructions for assembling a traveling bed, following driving instructions to reach a location, and many other common situations. And I am just referring to cases of following prespecificed procedures, not about specifying them or analyzing them.

> *«* (…) *our* [popular] *culture is relatively poor in models of systematic procedures. Until recently there was not even a name in popular language for programming, let alone for the ideas needed to do so successfully. There is no word for "nested loops"* (…)*. Indeed, there are no words for the powerful ideas computerists refer to as "bug" and "debugging." »*
>
> *(Papert, 1980, p. 22)*

Papert was not alone in realizing this association between programming and the existence of a vocabulary of ideas to interpret processes and doing: as I mentioned at the beginning of this section, he was already interested in such methods while still in Geneva, working with Piaget. For instance, in 1970 his then MIT colleague Marvin Minsky said:

> *« Before computation, the community of ideas about the nature of thought was too feeble to support an effective theory of learning and development.* (…) *Now we have a flood of such ideas, well defined and implemented, for thinking about thinking; only a fraction are represented in traditional psychology:*
>
> *Symbol table* — *Pure procedure*
> *Time-sharing* — *Function-call*
> *Functional argument* — *Memory protection*
> *Dispatch table* — *Error message*
> (…)
> *Data type* — *Hash coding*
> *Microprogram* — *Format matching*
> *Interpreter* — *Garbage collection*
> *List structure* — *Look-ahead*
> *Diagnostic program* — *Executive program*
>
> *These are just a few ideas from general systems programming and debugging; we have said nothing about the many more specifically relevant concepts in languages or in artificial intelligence or in computer hardware or other advanced areas.* ***All these serve today as tools of a curious and intricate craft, programming****. »*
>
> *(Minsky, 1970, my emphasis)*

This technical terminology itself is not the important part of this discussion. Some of it may end up finding its way into the language of a wider group than just professional software developers, but probably most of it will not. The point goes beyond the actual names that happen to be used at a given time: what matters is that their very nature as named entities reflects the reality that **such concepts have an autonomous, identifiable existence in the minds of programmers**.

I have personally experienced the benefits of contacting – and being able to interpret – a teacher with a rich perception on the fine nature of processes and the ability to explain them. Around the time of my 19th birthday, I started learning to play a traditional musical instrument, the Portuguese guitar – which is quite unique in its playing technique. I clearly remember practicing in the annex of a garage where a group of experienced players would be rehearsing, while I was practicing repetitive exercises for 2 or 3 hours in a row. After some sessions, I started to practice the reproduction of specific musical pieces, repeating them in detail, note by note. For several years, I immensely benefited from the close contact with my original teacher, not just through the help, advice, and insights he provided during his sessions with me, but also by being amidst his musical work with other experienced practitioners, both players and singers: I was in contact with their appreciations of the minute details and problems posed by each piece, by each particular approach or interpretation. My musical appreciation of the Fado de Coimbra genre expanded accordingly. However, my own technical skills as a player evolved slowly. I would struggle over a particular section or exercise because it wasn't "good" or "too muddled", or suffering from some other similarly fuzzy problem. I would try again, and again, and my teacher would tirelessly and duly point inadequate approaches, patiently draw my attention to errors or deficiencies, encourage my achievements and making sure I realized them.

Eventually, after about 3 years of hard work, I had become an acceptable player, able to perform for an audience, shamelessly, and rehearsing regularly with my own group of traditional players and singers. Wanting to improve my guitar-playing skills, sometime around my 4th year of experience as a Portuguese guitar player, I enrolled in private sessions with one young master player, who by coincidence shared with me a background as a college student of Electric Engineering. But the crucial difference in his approach to teaching was the way in which he devoted attention to the explanation and analysis of the minute details of the process of playing. He made me look into the precise shape of the fingernails used for plucking the strings, and on the precise movement and placement of the plucking fingers. He explained the various manners in how these interrelated factors impacted the process of plucking strings and their vibration, and therefore, the resulting purity of the sound. For my left hand, the one running the guitar scale, he showed me how the selection of any given finger for pressing a string at a specific point through a piece might allow me to easily press the next note, or otherwise be an hindrance, considering the movements I would have to make before and afterward; and overall, how I should consider the minute aspects of the **process** of playing Portuguese guitar.

I had to make a serious effort to become aware of finger and arm motions which had already become second nature, in order to realize exactly what was happening, and change my habits. But the results were – from my point of view – phenomenal: after a mere 6 weeks, I was managing to extract a sound from my guitar of a purity beyond anything I had previously accomplished, and my entire playing style was much more fluid and free. Not only that, my hearing had also changed. Now I had a much keener understanding of the relationship between what I was doing and the various resulting components of the sound I produced. A sound was no longer just "bad", "good", "edgy", or any of the other imprecise ways I had to interpret it; now I was aware of exactly how it was also the result of the specific way in which I was acting. I don't know many of this teacher's students, so I am unaware if other student's views were in any way similar to my own, regarding this awareness of the focus his teaching style had on the process of playing. But it was a powerful personal experience.

With this example, I am also making a connection to one of the examples Papert usually employs in his written works to convey these ideas: a process-based analysis of cascade juggling (Papert, 1980, p. 105-112). With that example, Papert explained his ideas on this matter of processes, by developing a Logo-like programming procedure for understanding cascade juggling. But he also used it to exemplify another of his ideas: once you are developing a process for doing something – or for interpreting something, that process invariably has errors. Or, in programming terms, bugs.

As I mentioned earlier, while programming a computer, one's reasoning is centered on achieving an adequate description of one's intentions. Acquiring awareness and proficiency regarding the very notion of processes and their description cannot be separated from an essential activity: identifying and correcting errors – **debugging**.

> *« A child* (…) *lives in a world in which everything is only partially understood: well enough, perhaps, but never completely. »*
>
> *(Papert, 1980, p. 116-117)*
>
> *«* (…) *when you learn to program a computer you almost never get it right the first time. Learning to be a master programmer is learning to become highly skilled at isolating and correcting "bugs"* (…)*. The question to ask about the program is not whether it is right or wrong, but if it fixable. If this way of looking at intellectual products were generalized to how the larger culture thinks about knowledge and its acquisition, we all might be less intimidated by our fears of "being wrong." »*
>
> *(Papert, 1980, p. 23)*

Papert acknowledged that this was not something that would instantly be recognized by a would-be programmer; in fact, accomplishing this change in perspective, from being "right" or "wrong" into being constantly on a continuum of improvement, requires considerable personal effort, and also the social and cultural support to make that effort happen.

> *« Children often develop a "resistance" to debugging (...). I have seen this in many children's first session in a LOGO environment. The child plans to make the Turtle draw a certain figure (...). It doesn't work. Instead of being debugged, it is erased. Sometimes the whole project is abandoned. Sometimes the child tries again and again and again with admirable persistence but always starting from scratch in an apparent attempt to do the thing "correctly" in one shot. The child might fail or might succeed in making the computer draw the picture. But this child has not yet succeeded in acquiring the strategy of debugging. »*
>
> *(id., pp. 113-114)*

Papert relates this resistance to the ethics of school and culture overall: "*what we see as a good program with a small bug, the child sees as 'wrong', 'bad', 'a mistake.' School teaches that errors are bad; the last thing one wants is to pore over them, dwell on them, or think about them* " (*id.*, p. 114). He explains that while a child might be perfectly happy to take advantage of the computer's ability to erase all traces of a mistake, the educational philosophy of computer programming demands a different attitude. Errors are beneficial because by their analysis, one gains a better understanding of the entire process and can correct the problem, instead of trying to achieve something out of tireless repetition.

This points to the need of careful consideration regarding the way in which programming environments are used, in order for actual debugging to occur. Children's attempts to reproduce typical non-programming drilling behavior while programming is a reminder to the importance of social and cultural structures in the adequate development of children's cognition. As I mention in the next section, for instance, regarding Radia Perlman's Button Box element of the TORTIS system, one of her concerns was whether it prevented children from debugging, since there was no way to edit a procedure, rather than erase it and write it again. A similar concern can be made regarding ToonTalk, the system I employed for my field research (*vd.* section 3.3.5). While ToonTalk does have an editing mechanism, called time-travel, this mechanism is quite limited, in that it simply allows a robot to reproduce its programmed actions up to a point, and then let the

child take over and remake the rest of the procedure – even if the error was simply that the robot was picking up the wrong element, all the rest remaining the same.

These concerns are sensible and pose a challenge to the educator using computer programming. As an answer, I propose two parallel ideas: modularity and the overall educational support and environment provided to the programmer. By modularity, I am referring to the circumstance of a program being, typically, a combination of many interacting parts. In ToonTalk, for instance, a program usually involves having various robots or picture behaviors, each producing its own effects. This modularity can serve as much of an entry-point into debugging as the ability to edit a single procedure. In fact, it can promote it, since a robot or picture behavior is visually an independent element, not just another row of textual code, and this can therefore promote the interpretation of problems as individual issues, instead of considering the code to be entirely wrong.

At the level of educational support available for the programmer, this is something that Papert had already advocated. Rather than have a child erase and redo small programs continuously, or accept it as a central strategy, an educator can provide both tolerance for errors, and be a source of inquiry and interest in the source of errors. Even if the quickest or more comfortable way to correct a ToonTalk robot or Button Box procedure is to remake them, the point is that they should be scrutinized to determine the actual problem, rather than simply erased without a second thought.

> *« Some people who observe the children's growing tolerance for their "errors" attribute the change of attitude to the LOGO teachers who are matter-of-fact and uncritical in the presence of programs the child sees as "wrong." I think that there is something more fundamental going on. In the LOGO environment, children learn that the teacher too is a learner, and that everyone learns from mistakes. »*
>
> *(id., p. 114)*

These two ideas, processes and debugging, are best seen together. It's hard to understand a large, complex system, whose features don't match one's expectations – be that system a technical construct or a mental construct. To correct a system, *i.e.*, to debug it, having the mental skills to interpret it in terms of its internal processes (either real or assumed) greatly simplifies a person's task. As I described in section 2.4.5, on p. 94, for some people this may mean the decomposition of a system into small, manageable bits; for others, it may best be done by "tinkering", experimenting with changes to the system. But in both cases, the complex entity being perceived as a system, as the result or ongoing evolution of a process – and programming contribute to the creation of personal models of this perception (Turkle & Papert, 1990).

Picking up on my personal guitar-playing example, I can provide a nice instance of this use of debugging. While developing one of my personal compositions, I was stuck on a particular phrase which invariably I only managed to "hiccup through", *i.e.*, play without the fluidity I intended. The ongoing work on understanding the process of playing the guitar scale led me to present this problem to my teacher, whose approach was a clear moment of debugging. By electing to play the notes prior to the phrase with a set arrangement of fingers, I was putting myself in a situation where I would have to make a huge jump across the scale with my index finger to press the appropriate locations. So, by placing the index finger on that specific location, one could back-trace across the phrase and determine which other selection of fingers could simplify the task of the index finger at that moment. From then on, it was just a matter of practicing, and without much effort the correction to the procedure was complete.

These ideas are not necessarily dependent on the existence of computers and computer programming. Thinking about procedures and debugging them can and has been done without resorting to any of these technologies. But when programming a computer, they become explicit ideas: rather than having a half-sketched idea of a procedure, one must actually think it over

entirely. Bugs must be found, interpreted, and explicitly corrected, not just bypassed or resolved without considering their nature.

> *« It is obviously not necessary to work with computers in order to acquire good strategies for learning. Surely "debugging" strategies were developed by successful learners long before computers existed. But thinking about learning by analogy with developing a program is a powerful and accessible way to get started on becoming more articulate about one's debugging strategies and more deliberate about improving them. »*
>
> *(Papert, 1980, p. 23)*

These are the basic ideas from which Papert developed his educational thought. Ideas of procedural knowledge and deliberate debugging in the end became just part of a wider intellectual panorama, for which there is currently a name: **constructionism** (Papert & Harel, 1991).

> *« Constructionism, my personal reconstruction of constructivism, has as its main feature the fact that it looks more closely than other educational '-isms' at the idea of mental construction. It attaches special importance to the role of constructions in the world as a support for those in the head, thereby becoming less of a purely mentalist doctrine. It also takes the idea of constructing in the head more seriously by recognizing more than one kind of construction (some of them as far removed from simple building as cultivating a garden), and by asking questions about the methods and materials used. How can one become an expert at constructing knowledge? What skills are required? And are these skills the same for different kinds of knowledge? The name **mathetics** gives such questions the recognition needed to be taken seriously. »*
>
> *(Papert, 1993, p. 143, emphasis in the original)*

A first remark on this presentation given by Papert himself of the field of constructionism is the absence in it of any mention of computers or computer programming. They remains crucial, as I'll explain shortly, but in constructionism the computational **means** "*that can, in principle, be used by educators to support the development of new ways of thinking and learning*" (Papert, 1980, p. xiv) have entirely taken their role as means, submitted to the goal: the development of new ways of thinking and learning. **Learning**, in particular, has become the central concern of Papert, as attested by the fact that he coined a name for the art of learning – **mathetics**[267].

The above abstract description allows one to easily overlook the significance of Papert's ideas. And Papert himself has argued[268] against attempts to start a debate on constructionism by providing a definition of it. Instead, he usually provides just an entry-point idea for the entire concept, and then provides ample materials from which the reader can build his or her own representations of the concept:

> *« Constructionism – the N word as opposed to the V word – shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. »*
>
> *(Papert & Harel, 1991)*

[267] Formed from the greek root *mathe* ($\mu\alpha\theta\eta$), "to learn" (Papert, 1980). One way to see it is to consider that mathetics is to learning what pedagogy is to teaching.

[268] "*If one eschews pipeline models of transmitting knowledge in talking among ourselves as well as in theorizing about classrooms, then one must expect that I will not be able to tell you my idea of constructionism. Doing so is bound to trivialize it. Instead, I must confine myself to engage you in experiences (including verbal ones) liable to encourage your own personal construction of something in some sense like it. Only in this way will there be something rich enough in your mind to be worth talking about*" (Papert & Harel, 1991).

> *« (…) the construction that takes place "in the head" often happens especially felicitously when it is supported by construction of a more public sort "in the world" – a sand castle or a cake, a Lego house or a corporation, a computer program, a poem, or a theory of the universe. Part of what I mean by "in the world" is that the product can be shown, discussed examined, probed, and admired. It is out there. »*
>
> *(Papert, 1993, p. 142)*

To follow up from this entry-point, I'm returning to my personal example of learning to play the Portuguese guitar. One cannot learn to play an instrument without actually creating several "products" with a real-world existence: sounds, notes, and hopefully, music.

In my history as a learner of guitar-playing, I've been lucky to benefit from those two teachers who, from different approaches, allowed me to develop crucial elements for my development in powerful ways. First, by allowing my technical skills to develop in a context of deep contact with the emotional significance of musical interpretation and the concrete notion of how much small details could impact that overall significance and conveyed emotions. Second, by allowing me to acquire a concrete notion of the process of playing and of the impact of my physical motions on the physical production of sound and music.

However, I can also easily imagine an alternative, fictional history, with all these elements absent. In this fictional story, I was shown specific drills to practice over and over, until the result was acceptable. When presenting the result, or drilling before a teacher, in this fictional story the teacher would correct the positioning of my hands during the drill, point out the wrongness in the movement of my fingers, telling me to slow down or to accelerate, or keep repeating the word "again" until I managed to improve or the teaching session was over.

A horror story? Perhaps. The truth is, I have often been in such situations. In fact, it is quite easy to find players (and even singers – ok, perhaps especially singers) who take it upon themselves to provide this kind of instructions. Or even stricter ones, such as "the way you are doing it is wrong. Do it **this** way." Perhaps I should have listened, after all these were as direct a suggestion as I could get, but I usually couldn't bring myself up to do it – even if I made a brief attempt at following such suggestions, the usual result was an overall discomfort that distasted my playing and irritated me. Given the amount of personal effort required to improve one's playing style, this distaste and irritation meant that I would invariably end up falling back to my usual style and take different routes for my development.

Oddly enough, I have also been in such situations out of my own accord. Possibly the person from whom I've most often have heard "again" over and over endlessly while practicing a piece was myself. Other situations where this would happen were while rehearsing with my colleagues. It's something most players can relate to: a group of players deciding to stick with a musical piece – sometimes to just a small musical phrase – until most of the group is fully satisfied or completely mesmerized.

There are several ideas from constructionism involved in this situation. But a baseline idea has to do with the meaning associated to the word "concrete" as I employed it four paragraphs ago. In traditional terms, all the experiences described in this example would have been considered "concrete": the guitar, being physical and the object of playing, is always present; all sound is produced by physical strings, and their motion is the direct consequence of my tactile and physical contact with the strings. The sounds reach my ears and those of anyone else present.

In the terminology and theoretical background of constructionism, this is not the approach to take to the meaning of "concrete"; rather, as I have described in section 3.3.3, pp. 128-132, a concept is abstract or concrete to a person not from any physical features of it, but from the nature of that person's mental relationship with that concept, regardless of whether such concept has any physical features or is a purely mental construction (Wilensky, 1991).

Seeing the above guitar-playing example under this light, a mechanical effort to induce a particular behavior from a finger can be entirely abstract when it is detached from any particular goals or context, other than "this is the right way". And it can also be entirely concrete when one recognizes it as the next step in one's path, as the necessary route to reach the intended goal. To put in more plain terms, in one case, highly abstract, I'm trying to reposition my playing hands, and fight the overall discomfort at the unusual playing situation, with the resulting sound getting worse and resisting the unavoidable feel of uncertainty while doing it. ("Someone told me it was the right way; is it really?") In another case, highly concrete, I have constructed my understanding of the purpose (or at least the high feasibility) of the novel approach. My hands may be uncomfortable, but I no longer am just listening to the sound getting worse: now I know what to look for in the sound. I know it may be sounding worse overall, but I can begin to notice those specific changes I was aiming at, I can drill my hands a bit more but the "worse" sound is something I can look and interpret in a rich way, almost as if I could tell the strokes of a painter instead of just the color patches. I can even imagine how a particular feature, sensation or texture might be further developed. I may end up discovering I was wrong after all, but it is no longer a slot-machine betting, all-or-nothing situation: I am not simply mindlessly repeating over and over the same motions, until they smooth out or become entirely my own. It is a much more interactive process, a process of discovery and change. Each repetition is not just another attempt, it becomes a novel experiment. And yet, for the unaware viewer, I would probably be seen as doing the exact same thing: drilling.

This creation of such personal relations to knowledge is central to constructivism. It is not simply "learning by doing" – although certainly doing is central to it. Papert originally referred to this in 1980 resorting to the concept of **syntonic learning**.

> *«* (…) *a fundamental mathetic principle: Make sense of what you want to learn.* (…) ***syntonic learning*** (…) *is borrowed from clinical psychology* (…)*. Sometimes the term is used with qualifiers that refer to the kind of syntonicity. For example,* (…) ***body syntonic*** (…) *is firmly related to children's sense and knowledge about their own bodies.* (…) ***ego syntonic*** (…) *is coherent with children's sense of themselves as people with intentions, goals, desires, likes, and dislikes. »*
>
> *(Papert, 1980, p. 63)*

It's easy to see how guitar-playing was body syntonic. My body had to be involved! But the ego-syntonicity was a major part. I wanted to learn how to play the guitar. That inner want was driving my efforts, my attention. And while my second teacher provided the awareness for the minute details of the technical process of playing the guitar, it was due to the efforts of my first teacher that my own efforts were not solely concerned with the technical development of guitar-playing. I learned to value each minute detail of an interpretation in terms of its emotional content and impact. For instance, a musical piece which bears sadness can be immensely transformed if one can bring some "emotional spice" to this plain feeling. That sadness can include various shades or flavors of hope; of disbelief; of acceptance or rejection of fate; of inner struggle, or submission. All these can turn the interpretation of a musical piece from a plain, "flat", reproduction of notes into a memorable moment. By being in an environment where such self-reflection was emphasized, I developed a better, more concrete understanding of the complex nature of human emotions. But I also learned to value the interpretations where those subtleties could be found. This meant that instead of simply accepting decent recordings as a reference for a given musical piece, I sought and absorbed as many different varieties and versions as I could, even if only available in low-quality tape recordings under layers of noise and hiss, eager to appreciate those musicians that could best put their emotions onto sound. In the process, not only did I end up learning a lot of history about the development of the musical genre itself, but I also started to be aware of a deluge of sociological information, due to the sources of inspiration for the musical themes: unexpected references to faraway regions, contrasts and evolution in poetical style. My first attempts at using a computer to

define a musical score and register and record music came about in connection with the wish to save my personal compositions. And I also got in contact with mechanical aspects connected to the behavior of metal strings under tension and various types of wood – which I didn't pursue further.

The above description, if successful, should render clear the learning power – or, in Papertian terms, mathetic power – of rendering learning concrete by emphasizing the multiple personal connections with the learning domains. Papert advocates that teaching strategies poll this power, rather than fight it. This is ideological and programmatic content of constructivism:

> *« Constructionism* (…) *does not call in question the value of instruction as such. That would be silly* (…)*. The constructionist attitude to teaching is not at all dismissive because it is minimalist – the goal is to teach in such a way as to produce the most learning for the least teaching. Of course, this cannot be achieved simply by reducing the quantity of teaching while leaving everything else unchanged. »*
>
> *(Papert, 1993, p. 139)*

Papert compared such powerful learning examples with research from various areas. As I explained in section 2.4.5, p. 94, one source was the works on "primitive" societies by French anthropologist Claude Lévi-Strauss, from where he picked up the word "*bricolage*" to describe a learning approach in contrast to the formal one (Turkle & Papert, 1990, pp. 129, 135-143): "*Bricoleurs construct theories by arranging and rearranging, by negotiating and renegotiating with a set of well-known materials*" (*id.*, p. 136). Another was the research of Piaget on similar cognitive strategies in young children:

> *« Essentially, Piaget made the same observation as Lévi-Strauss, except that where the anthropologist had looked at la pensée sauvage in distant societies, Piaget looked at la pensée sauvage close to home, in children. What they both saw was thinking that differed from "our" norms and yet had a degree of inner coherence that forbade dismissing it as simply erroneous. »*
>
> *(Papert, 1993, p. 151)*

Another source of insight for Papert were the research efforts and personal accounts on the thinking styles used by "sophisticated thinkers", such as professional scientists – including Nobel prize winners.

> *«* [Piaget and Lévi-Strauss] *failed to recognize that the concrete thinking they had discovered was not confined to the underdeveloped – neither to Lévi-Strauss's "undeveloped" societies nor to Piaget's not yet "developed" children. Children do it, people in the Pacific and African villages do it, and so do the most sophisticated people in Paris or Geneva.* (…) *features of what* [they] *identify as "concrete" are present at the core of important and sophisticated intellectual enterprises* (…) *»*
>
> *(id., ibid.)*

Thus one can better interpret the ideas underlying Papert's constructionism, neatly encompassed by this sentence:

> *« Rather than pushing children to think like adults, we might do better to remember that they are great learners and to try harder to be more like them. While formal thinking may be able to do much that is beyond the scope of concrete methods, the concrete processes have their own power. »*
>
> *(id., p. 155)*

One should bear in mind that Papert doesn't intend to provide any argument detrimental of the power of abstract thinking. Rather, his arguments are detrimental to the over reliance on abstract thinking, ignoring the learning power of concrete thinking.

> *« I am a mathematician and know firsthand the marvels of abstract reasoning. I know its pleasures as well as its power. I also know how stultifying it can be if it is used indiscriminately. Our intellectual culture has traditionally been so dominated by the identification of good thinking with abstract thinking that the achievement of balance requires constantly being on the lookout for insidious forms of abstractness that may not be recognized as such by those who use them. »*
>
> *(id., p. 146)*

As a very short example illustrating this last remark, I'll use another personal event: in my undergraduate studies, during a lecture on digital and analog communications, I was despairing to grasp what in my mind was the "actual" meaning of the lecturer's formulaic exposition. For the benefit of the reader, I may add that, at the time, I already had a very decent understanding of both the formulaic transformations involved and of the kind of effects an electric or radiofrequency signal underwent when subjected to power, phase, and frequency transformations. So I asked the lecturer more or less along these lines: "look, I have no problem following the mathematical transformations of this entire reasoning, from one equation to the next. But if this is describing changes in the signal, what are those changes? How is the signal itself being affected, between the original signal and the final one?" His reply was the least helpful: "I don't understand your question. Everything is there. Everything is there" (and he simply pointed at the formulaic transformations). Clearly, I now know I needed to understand which few transformations were merely formulaic, to proceed in the mathematical analysis of the process, and which were both formulaic transformations but also identifiable changes and impacts to the physical signal.

It must be underscored, however, that although already defended in more simplistic terms by Dewey and others, as I described in section 4.1.3, approaches based on concrete experiences have serious limitations, regarding the amount and complexity of ideas that it can incorporate. Ideas from mathematics and physics, for instance, need more than just an interest in strings, guitar wood, and tuning to develop beyond a starting point. And here, again a major contribution of Papert was his realization that **this limitation can be tremendously changed by using computers and computer programming**.

> *« None of this absolutely requires computers. What we see in experiences* (…) *is how the computer simply, but very significantly, enlarges the range of opportunities to engage as 'bricoleur' or 'bricoleuse' in activities with scientific and mathematical content. »*
>
> *(id., p. 145)*

Papert worked with various researchers, developing ways of using programming and other educational techniques in this fashion. One important technique, whose development started prior to the development of the concept of constructionism itself, was his notion of **microworlds**. He calls these "*incubators for knowledge*" (Papert, 1980, p. 120). They serve as a technique allowing a concrete approach to knowledge, overcoming two existing obstacles for such an approach. The first of those obstacles arises from the structure of knowledge: the problem posed in many fields by

prerequisites[269]. The other is found in each individual; as Papert calls it, "*finding a context for the construction of 'wrong' (or, rather, 'transitional') theories*" (*id.*, p. 132).

In the first case, a microworld structures knowledge, allowing a student to interact within a specific context, acquiring a concrete grasp of its inter-relationships directly. Giving Papert's own example, a student can use a microworld to explore Newtonian "perpetual motion", a concept with students, most likely, have never contacted.

> *« In the absence of direct and physical experiences of Newtonian motion, the schools are forced to give the student indirect and highly mathematical experiences of Newtonian objects. Their movement is learned by manipulating equations rather than by manipulating the objects themselves.* (…) *The simplest way in which our computer microworld might help is by putting students in a simulated world where they have direct access to Newtonian motion. This can be done when they are young. Instead of making students wait for equations, it can motivate and facilitate their acquisition of equational skills by providing an intuitively well understood context for their use. »*
>
> *(Papert, 1980, pp. 123-124)*

To overcome the second obstacle, a microworld – or more likely, a series of microworlds – can let a child employ some form of computer programming to specify behaviors and rules in that world. In this manner, assumptions can be made and tested, to determine their applicability. As they reveal deficiencies, they can be refined. For instance, resuming Papert's example based on Newtonian motion, he proposes using Logo's TurtleTalk (*vd.* p. 144) to reinterpret Newton's laws of motion based on Logo command's impact on the Logo turtle – replacing Newtonian forces with a combination of Logo's FORWARD and RIGHT commands.

> *« 1. Every Turtle remains in its state of rest until compelled by a TURTLE COMMAND to change that state.*
>
> *2. a. The input to the command FORWARD is equal to the Turtle's change in the POSITION part of its state.*
>
> *b. The input to the command RIGHT TURN is equal to the Turtle's change of the HEADING part of its state.*
>
> (…) *How can students who know Turtle geometry (and thus recognize its restatement in Turtle laws of motion) now look at Newton's laws? They are in a position to formulate in a qualitative and intuitive form the substance of Newton's first two laws by comparing them with something they already know. They know about states and state-change operators. »*
>
> *(Papert, 1980, p. 127)*

The point is not that such a translation into turtle commands of Newton's laws should be studied; that would be a complete subversion of Papert's ideas. Such a translation presents a new concept in a familiar-looking way, but it needs to be explored, and tested for validity, by using the

[269] Papert put it this way: "*Someone who wanted to learn about aerodynamics might lose interest upon seeing the set of prerequisites including mechanics and hydrodynamics that follow an exciting course description in a college catalogue. If one wants to learn Shakespeare, one finds no list of prerequisites.* (…) *The route into Shakespeare is no less complex, but its essential constitutive elements are part of our general culture: it is assumed that many people will be able to learn them informally*" (Papert, 1980, p. 123). Perhaps. I contrast it to this informal account by a Portuguese mathematician in search of a novel to read on a vacation: "(…) *I confessed to a friend my ignorance of North-American literature. He was in Philadelphia, completing a doctoral thesis contrasting Poe and Mailer; he seemed to me the right person to recommend a novel. He answered me that contemporary North-American literature was a world in itself. That if I wanted to get to know a little, even if just a little, I would have to start by Henry James and Edith Wharton, read them again, and again. After a few years and a lot of Faulkner, I might be able to proceed…*" (Crato, 2005).

actual commands involved (and 'explored' means that children should try to determine their internal validity and consistency by making different attempts on their own, instead of aping a test procedure provided by their teacher).

From this microworld, Papert proposes the design of more microworlds, "*in which the laws of motion move towards a closer approximation of the Newtonian situation. To do this, we create a class of microworlds that differs in the properties that constitute the state of the Turtle and in the operators that change these states*" (*id.*, *pp. 127-128*). For instance, defining turtles that have not only a position, but a velocity, and defining a "setvelocity" command to change it, and defining turtles with acceleration. Finally, to approach Newton's third law of motion, dealing with interactions, by having various turtles which can communicate, with commands for exchanging speeds. Such a series of microworlds could also be expanded to cover relativistic motions, as well as fantasy laws of motion, to explore their similarity and distinctions.

All these examples and elements of Papert's thought require a deep and intentional involvement of the student. While he constantly refers to the freedom to experiment, to follow and develop projects that are syntonic with the student, there is also the underlying concept that students must work very hard to learn. Papert calls this concept as being **demanding permissiveness**: children are expected to "*work hard on a project* (…) *but* (…) *given very wide latitude in choosing the project*" (Papert, 1993, p. 124). He considers these two factors absolutely essential:

> *« Many teachers strive to bring into their traditional classes something very like the teaching attitude* [of permissiveness] (…). *But the permissiveness is deceptive, even if the intentions are good, when the demand is for children to fit into the straightjacket of the traditional curriculum, especially in subjects like math and science where the elbow room for personal appropriation is so narrow. »*
>
> *(id., ibid.)*

That is, the culmination of Papert's educational ideas goes beyond the educational use of computers and computer programming, beyond the changes to teaching style, it strikes at the standard concept of a curriculum, and eventually Papert assumed a stand of a necessity to replace standard schooling, as mentioned in section 2.4.5. However, I won't pursue this part of his educational thought, since preschool and kindergarten education are themselves typically devoid of a curriculum (Ministério da Educação, 1997). Suffice to say that in a move similar to Dewey's creation of a Laboratory School (*vd.* section 4.1.3), Papert has recently been involved in the development of an educational institution to explore the development of his ideas in this regard, called "The Constructionist Learning Environment"; an inspiring and promising research report on the results of this institution has recently been published (Stager, 2005).

> *« The Constructionist Learning Environment's emphasis on building, crafting, making and doing gave life to the theory of constructionism. Although there had been several previous educational projects built upon Papert's theory, principally by Papert and his students, the CLL represented the first learning environment built entirely from the ground-up to support constructionism. The project at the Maine Youth Center also represents the most recent experiments in school reform led by Seymour Papert. The independence from any existing school structure, both physically and philosophically, also adds to the significance of the CLL. »*
>
> *(Stager, 2005)*

Throughout this section, I have presented Papert's educational thought, which although mainly concerned with the levels of formal education, shares many insights and views with modern preschool and kindergarten education. However, as discussed in section 4.2.1 and further presented in the following section, not only is the use of the computer at these educational levels extremely

limited, and research on the methods for using programming, particularly regarding Papert's inspiring ideas, has not been conducted in depth. This thesis attempts to be a step towards a change in this state of affairs. And why should I, a computer scientist, hope to provide a meaningful contribution to this? I believe I can now answer, by offering a quote from Marvin Minsky, one of Papert's colleagues at the Artificial Intelligence Lab of the MIT:

> *« The computer scientist thus has a responsibility to education.* (…) *He knows how to debug programs; he must tell the educators how to help the children to debug their own problem-solving processes. He knows how procedures depend on their data structures; he can tell educators how to prepare children for new ideas. He knows why it is bad to use double-purpose tricks that haunt one later in debugging and enlarging programs.* (…) *The computer scientists are the ones who must study such matters, because they are the proprietors of the concept of procedure, the secret that educators have so long been seeking. »*
>
> *(Minsky, 1970)*

4.2.3. Computer Programming in Preschool and Kindergarten

A review of the survey of child-usable programming languages and systems on section 3.3.4 allows one to identify several systems which seem to be at least potentially suited for simple tests with preschool-aged children. However, by surveying documentation on the research or use of computer programming with children aged 3, 4 or 5, I managed to find only the following major bodies of information:

- many studies and reports on the use of Logo and Logo-related systems;
- a few studies associated with other programming systems for children, including my own efforts detailed in this thesis (on chapters 1, 1, and 1, and on section 3.4);
- some "suitability" trials performed by companies or researchers involved in the process of developing new programming systems for children;
- some personal comments from teachers, parents, as well as childhood recollections.

This lack of information on possible attempts to use the many of the systems form section 3.3.4 is quite disappointing. Table 26, below, provides a list of all such systems:

Programming system	Creation year	Described on page
Pinball Construction Set	1983	190
AlgoBlock	1993	172
The Incredible Machine	1993	185
Thinkin' Things Collection 3 – Half Time	1995	165
Logiblocs	1996	173
AlgoCard	1997	173
Drape – DRAwing Programming Environment	1999	166
AutoHAN/Media Cubes	2001	176
Concurrent Comics	2003	168

Table 26 – Systems potentially usable by preschoolers, for which there is record of such use

Two different cases are found within these systems, however:-

- some are still just research prototypes or proof-of-concept systems, and for these I find it likely that no actual trials with preschool-aged children took place (I am referring to AlgoCard, Drape, AutoHAN/MediaCubes, and Concurrent Comics);
- of the remaining systems, particularly those which benefited from large commercial exposure (*e.g.*, The Incredible Machine, Logiblocs), it is likely that several undocumented cases occurred.

I have reached these assumptions from analyzing the cases presented in Table 27, on the following page. These are programming systems, several of them commercially released, for which there is only scarce (and often unreliable) documentation available, **regarding their use with preschoolers**. (Some of these systems are extensively documented regarding their use by older children, *e.g.*, Stagecast Creator and Squeak Etoys.)

For some of the systems mentioned in that table that are still undergoing research and development, the available documentation point towards the research focus having been older children, with no preschoolers included in research or development trials, or only included informally (Eco-Pods, Icicle, Magic Forest). In the remaining cases, there are on-line spurious comments, photographs, or user testimonies from non-research settings. At least for some of the commercial products (*e.g.*, the LEGO products), it is plausible to assume that private documentation may exist regarding marketing research or user testing, but I have had no contact with any such documents, nor did I came across any indication of their existence.

Programming system	Creation year	Described on page	References	Summaries
Lemmings	1991	184	VV.AA., 2005	Two participants in a Web forum reported playing Lemmings successfully since the ages of 4 and 5.
Squeak Etoys	1996	146	Kay, 2005 Rose, 2005 Stowe, 2005	Little has been done with preschoolers in Squeak, due to UI design options for older children (Kay, 2005). Some efforts took place in Chicago's Columbia College, in 2004 (Rose, 2005; Stowe, 2005).
Stagecast Creator	1997	151	Nyquist, 2005	A parent reports that when his son was 4 and 5 he managed to make characters move on the screen and follow the Stagecast Creator tutorial.
Tangible Programming with Trains / Intelli-train	1998	174	LEGO, n.d.- 2 Hamilton, 2004	The mass-market product resulting from the original research project was LEGO's Intelli-train series. In its Web pages, LEGO targets these products for the age group "3+" (LEGO, n.d.-2). However, even though LEGO blocks and products are quite popular worldwide, a Web search only retrieved a page with photographs documenting the use of Intelli-trains in a single kindergarten (Hamilton, 2004).
Curlybot	1999	174	Frei *et al.*, 2000	Frei and his colleagues report that the development of curlybot was "*aimed at children in their early stages of development – ages four and up*" (p. 129). They provide limited information on testing with children, though: the only reference to young children, after some short testing, is: "*We hoped to see trends in play between the different age groups, however the only conclusive results we found were that children under the age of four generally could not meaningfully interact with curlybot*" (p. 134).
Magic Forest	2000	163	Cnotinfor, 2005 Correia, 2005	Cnotinfor, the main company behind the development of this product, markets it for children aged 4 and up (Cnotinfor, 2005), but only informal trials were conducted with children younger than 6 years (Correia, 2005).
Icicle	2002	180	Sheehan, 2003	In this article, one year prior to the paper describing Icicle (Sheehan, 2004), the researcher-developer is focusing his attention on children aged 7 and up. Since Icicle includes written captions in its rules, I assume that this focus hasn't been enlarged.
PervoLogo / MicroWorlds JR	2004	160	LCSI, 2004 Hansell, 2005	LCSI markets MicroWorlds JR as being for "pre-readers" or "Pre-K" (LCSI, 2004). I have been unable to find documentation on actual usage, but on the MicroWorlds Web forum a teacher reported "*My older students have seen the younger students use cartoon characters, robots, etc. and they want to use them also*" (Hansell, 2005) – no age is given.
Eco-Pods	2005	178	Kikin-Gil, 2005	According to the developer of this system, the target age groups was children aged between 4 and 6. However, the age of all children involved in user testing was 6, except for two (age 5 and age 7).

Table 27 – Summary on preschool-age use of systems for which there is scarce (or unreliable) information

These scattered pieces of information, although lacking scientific reliability, provide a patchy social and cultural background of undocumented attempts to use programming tools with very young children, even if just at the level of trial-and-error or wishful attempts.

But for a history of more rigorous research, development, and user-testing of computer programming with children aged 3, 4, or 5, one must turn towards the remaining systems (starting with the oldest reported cases): Logo (and derivations: TORTIS, FASTR/TEACH, Valiant Roamer, LEGO Mindstorms products); My Make Believe series; Physical Programming; Electronic Blocks; Topobo; and System Blocks. All these systems were presented in section 3.3.4.

Figure 240 – Radia Perlman
From: http://research.sun.com/people/images/radia/perlman2.jpg

The most notable pioneer on the use of computer programming with very young children was Radia Perlman (Figure 240). Between 1974 and 1976, she developed a set of tangible elements allowing the programming of a physical turtle robot, inspired on Logo's TurtleTalk, creating the TORTIS system (Perlman, 1974), described on p. 171. Her specific goal was "*to overcome the typing hurdle, and make many of the advantages provided by the learning of full computer languages accessible to children as young as three or four years*" (Perlman, 1976, p. 4).

Perlman tested the first component of her system, the Button Box (Figure 241), with "*around 25 children aged 3 to 5*" (*id.*, p. 14), and she describes the initial sessions with several of these children, including the youngest, at age 3½ (Perlman, 1974).

Perlman was a pioneer on several accounts. At the software level, her approach to the use of Logo based on one-touch commands overcame the most obvious barrier to the use of programming in this age groups, by avoiding the typing of full textual commands – an approach that was also being used, at more or less the same time, by Paul Goldenberg, in his FASTR system (Goldenberg, 1974), and Cynthia Solomon, in her TEACH system (Solomon & Papert, 1975); both were however traditionally based on a keyboard (Goldenberg, 2005; Hildreth, 1975).

At the hardware level, Perlman's Button Box was a precursor to the modern command board of the Valiant Roamer turtle robot (*vd.* Figure 128, p. 172). And her Slot Machine shares some insights with the later AlgoBlock system (described on p. 172), such as the ability to program by sequencing physical objects and the use of lights to indicate the card in execution (Perlman, 1976, p. 17).

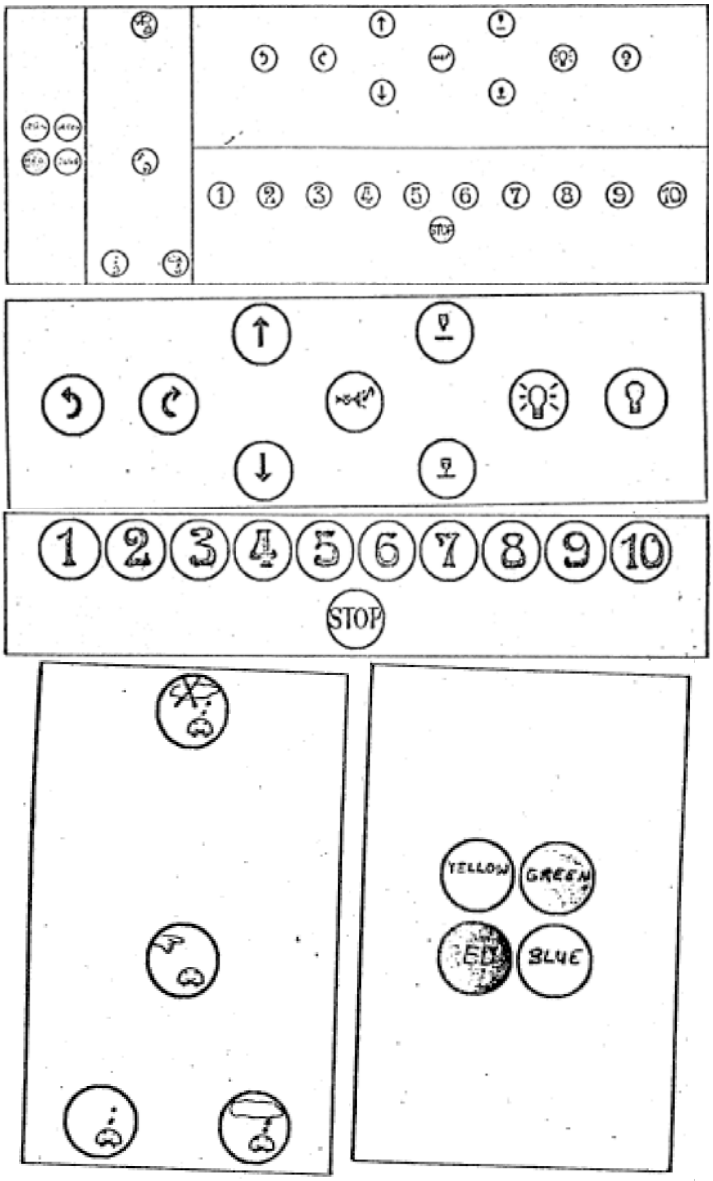


Figure 241 – The TORTIS Button Box and its components
From: Perlman, 1976, pp. 9-12

But in my view, her most remarkable pioneering contribution was **the way in which she paid close attention to the way in which children actually used the system, not just the technical issues or problems**. For instance, she describes parent's and sibling's behavior during the programming sessions, the settings where programming is taking place (office, nursery school), and the reactions, behavior, expressiveness and comments of the children while programming (Perlman, 1974; 1976; 2005). She also pondered over the cognitive difficulties behind some aspects of programming and tried to overcome them.

Some of what she calls "*interesting things*" (Perlman, 1976, p. 14) are quite similar to some of the programming hurdles I describe and organize further ahead in section 6.1. Specifically, my first hurdle is "*Children can easily get weary from unforeseen system behavior or difficulties*" (p. 344); Perlman's 1st and 2nd "interesting things" were similar early observations:

> *« 1) Once a child is confused by something he gets very discouraged and says, "that is too hard for me" or "I'm not smart enough for that", Both of which are certainly attitudes we do not want to cause, so it is important not to give the child new buttons before he is ready.*
>
> *2) Some children require constant interaction and suggestions about further things to try or else they start doing one thing over and over and get bored. »*
>
> *(Perlman, 1976, p. 14)*

My fifth hurdle in section 6.1 is "*Children find it hard to consider the possibility of being wrong and not knowing it*" (p. 350); Perlman reported a similar problem:

> *« If when drawing the picture they had, for instance, made the turtle turn the wrong way at first and then corrected, or stopped to make the turtle toot several times, they were surprised that the turtle did the same thing when drawing a picture the second time. »*
>
> *(id., p. 15)*

However, she differs from me in her interpretation of this problem. My programming sessions with children led me to my conclusion on the issue residing at the very heart of the notion of "wrongness", as I describe on p. 350. Perlman, however, considered this to mean that children "*thought the picture was the procedure as opposed to the set of commands the turtle executed while remembering*" (*id.*, *ibid.*). I believe this interpretation to be a helpful insight – an alternative, parallel perspective to my fifth hurdle (if one takes it under a more general formulation, such as "the result is the procedure", rather than "the picture is the procedure").

Another pioneering stance of Perlman was her concern with providing children with a visible track to the progress of a program's execution. As the Button Box buttons were pressed and the turtle robot moved, the program's instructions were presented on a computer screen. According to her reports, though, children didn't manage to associate the screen commands with the movements of the turtle (Perlman, 1976, pp. 15-16; Perlman, 2005). To address this problem, she built the second component of the TORTIS system, the Slot Machine (Figure 126, p. 171), made of several Plexiglas rows, "*with slots in the top for the user to place cards*" (Perlman, 1974, p. 4), which represented the turtle commands. The reasoning behind it was enabling children to be physically involved in the process of specifying a sequence of commands, which hopefully would contribute to overcome the problem, as she saw it, of mistaking the resulting picture for the program. This conception didn't prove fruitful, though; in her own words, "*initially there is so much that the child has to do to get the turtle to execute a command (find a card, put it in a position, and push a button) that it is sometimes difficult for them to understand that each card stands for some specific command*" (Perlman, 1976, p. 26).

At this level, ToonTalk is quite helpful, since children can act directly upon the objects, doing the actions they intend to do; *i.e.*, it is more immediate. Perlman provided suggestions regarding system improvements or novel systems, precisely with the goal of achieving this immediacy in response to programming. For instance, she suggested the use of the Slot Machine to produce tunes, with a command card for each note, and a number argument specifying the duration of that note; she also suggested turning indicator lights into pushbuttons to allow children to cause the immediate execution of a specific instruction within a program, simplifying debugging (*id.*, *ibid.*). Incidentally, this concern with immediacy is also connected to what was recently expressed by Erez Kikin-Gil while developing the Eco-Pods system (described on p. 178): "*the outputs, whether light or other sensorial stimuli, should not be subtle as the environment 'noise', such as distracting light, sound or movement, tend to dominate it*" (Kikin-Gil, 2005, p. 23).

Perlman was already expressing her concern about the way in which programming systems such as her Button Box, which are based on the reproduction of hidden-away procedures, lack a proper editing procedure (programs for her Slot Machine were easy to edit, simply re-ordering the cards). This is also mostly valid regarding ToonTalk[270]. The problem with this condition, in light of Seymour Papert's ideas, described in section 4.2.2, is that "*without the ability of easy editing the child does not learn the healthy concept that there is no 'right' or 'wrong'* [in learning through programming]*, and that any procedure can be fixed to do what they want, and even a working procedure can always be improved*" (Perlman, 1976, p. 15). Conversely, she also acknowledges that "*adding editing commands before the child has mastered the four memory buttons*[271] *would overwhelm the child with too many new concepts*" (*id.*, *ibid.*). While agreeing in principle with these observations, I consider this issue to be more of a question of the style in which children's programming activities are conducted – as I have argued in section 4.2.2.

Finally, I believe Perlman was basically right in the set of goals for further research that she laid down in 1976, and within which my research pretty much falls. However, I have gone beyond these goals, by having also involved the educational context and the preschool teachers in my research:

> *«* (…) *there is a whole area of research involving working with children with the system, finding difficulties they encounter, and* (…) *inventing new ways to present the concepts they find difficult* (…) *»*
>
> *(Perlman, 1976, p. 29)*

Regarding the involvement of teachers in programming, little was done at the preschool level, but there were pioneer efforts at the elementary level, followed by many others since. Soon after the development of Logo, Seymour Papert provided support in the form of a list of specific suggestions of educational activities using programming (Papert & Solomon, 1971) – an approach I am also following in the cookbook of section 3.4. And at the same time as Perlman was developing her system, Ellen Hildreth published a paper supporting the efforts of teachers of the first and second elementary grades. In that paper, one of the approaches she took was also used by me when working with preschool computer-activity teachers (*vd.* Annex VI): finding adequate terminology to address programming issues with young children (Hildreth, 1975, pp. 6-10).

After the 1974-1976 pioneer work of Radia Perlman, in 1980 Seymour Papert stated "*I have seen hundreds of elementary school children learn very easily to program, and evidence is accumulating to indicate that much younger children could do so as well*" (Papert, 1980, p. 13). Sadly, he does not provide more information on that evidence, although it certainly included Perlman's work, seeing that it was taking place at MIT's AI Lab, of which Papert was co-director.

[270] ToonTalk's limited editing option of using time-travelling only addresses these problems partially, since children can re-route the course of actions of a robot, but not actually edit it; however, as I mention in section 4.2.2, ToonTalk does have editing possibilities in terms of the distributed nature of its programs and the mechanism of behaviors (*vd.* p. 374).
[271] In a more general sense, before the child masters the immediate-response programming system.

Since the creation of Logo, a large body of research has been developed on the use of computer programming by children, but mostly centered on children of formal school age, *i.e.* 6 years old or more (Clements & Nastasi, 1999). Research focusing on the use of computer programming with younger children is less common, as can be seen by analyzing current review studies (Clements, 1999b; Clements & Sarama, 2003; Gillespie, 2004).

Thus, below I present an historical review of reports on the use of computer programming with children aged 3, 4, or 5. Afterwards, I provide of personal overview.

In 1978, a study found that "*four- and five-year-old children had great difficulty both in learning to program a turtle and in transferring what they had learned*" (Gregg, 1978, acc. Mayer *et al.*, 1986).

In 1983, three five-year-olds, using Logo, "*determined the correct length for the bottom line of their drawing by adding the lengths of the three horizontal lines that they had constructed at the top of a tower: 20 + 30 + 20 = 70*" (Clements, 1983, acc. Clements, 1999c). One other study attempted to establish predictor variables for the ability to program, but was inconclusive (one experiment in Canada, with 26 kindergarten children, another in France, with 16 kindergarten children). Children were "*taught to program with the LOGO Turtle using a simplified form of LOGO*" (Munro-Mavrias, 1983), but this consisted only of "*a 30-minute computer literacy demonstration and two 20-minute sessions of unstructured exploration with the four LOGO commands for the procedures*" (*id.*, *ibid.*) Post-testing involved requiring children to draw geometric shapes with the Logo turtle (*id.*, *ibid.*) and attempting to match the results with the candidate predictor variables.

In 1984, there was a report that regarding language use, 3- and 4-year-old children verbalized significantly more about their Logo computer pictures than about their hand-drawn works (Warash, 1984, acc. Coniam, 1992, and Clements & Sarama, 2003).

In 1985, research noted that working with Logo in a narrative context increases kindergarteners' readiness scores on visual discrimination, visual motor skills, and visual memory (Reimer, 1985, acc. Clements, 1999b). In other research, a preschooler abstracted novel ideas from his Logo experience, such as "*the discovery that reversing the turtle's orientation and moving it backward had the same effect as merely moving it forward*" (Tan, 1985, acc. Clements, 1999c). In this same year, another researcher examined five-year-old children's activity choices in an early childhood classroom setting "*designed to foster interactive microcomputer activities*" (Schwartz, 1985). However, there were serious negative assumptions underlaying this study, for it "*was hypothesized that engaging and sustaining the children's active involvement with interactive computer experiences would be very difficult, due to their limited academic skills. Further, it was expected that 5-year-olds would not develop even the simplest programming skills*" (*id.*, *ibid.*). Schwartz concluded that "*some 5-year-old children will choose interactive microcomputer activities, but none will develop programming skills in even the simplest form, although directly taught at timely moments*" (*id.*, *ibid.*).

In 1986, there where two contradictory similar experiments. In one (Degelman, Free, *et al.*, 1986), the intention was to detect whether five-year-olds exposed to 15 minutes per school day of a single-keystroke version of Logo, during 5 weeks, fared better at rule-learning problems than a control group. They found that the Logo group had a significantly higher proportion of correct rule identification responses on the two problems that were administered to all children. In the other experiment (Degelman, Brokaw, *et al.*, 1986), 44 kindergarteners were split in three groups: a Logo group, a non-programming graphics-control group, and a comparison group. For 8 weeks, children had 15 minutes individual instruction twice a week; after the 8 weeks, they were tested on measures of rule-learning and creativity, but no significant measures were detected, although the number of errors was smaller in the computer-experience groups.

Other studies conducted in 1986 focused on children's interactions while programming, concluding that that "*as preschoolers become more competent in working with Logo, they become more self-directed and rely more on peers and less on teachers, forcing a shift from teacher to peer mediation*" (Miller & Emihovich, 1986, acc. Clements *et al.*, 1993), and that this motion towards peer-mediated problem-solving may be necessary for achievement gains to occur, in view of detected effects of peer interaction on retention levels of preschool children (Perlmutter *et al.*, 1989[272]). Also reported were the case studies of two kindergarteners working in a public school computer lab and two preschoolers using computers in a university-based computer classroom. The analysis of the case studies focused upon what the kindergarteners and preschoolers said and did while computing, finding that "*for the kindergarten children, LOGO facilitated collaborative behavior and enhanced the expression of social and language skills. For the preschoolers, LOGO encouraged highly focused task-related behavior, but did not invite collaborative learning*" (Strand, 1986).

In 1987, again the work with Logo in a narrative approach was taken, this time concluding that it enhanced language-impaired preschool students' perceptual-language skills (Lehrer & deBernard, 1987, acc. Clements, 1999b).

In 1988, a study on the use of Logo to learn geometry was initiated involving children in grades K-6. A total of 644 children were used in control groups, subject to the regular geometry curriculum and 12 were involved in geometry activities using Logo (Clements, 1999b); "*students in grades K-1 used a single-key version of Logo, students in grades 2-4 used TEACH, and students in grades 5-6 used an enhanced version of regular Logo*" (Battista & Clements, 1991). In conclusion, "*Logo students performed better over all. They demonstrated flexible consideration of multiple properties of geometric shapes that may help lay the groundwork for hierarchical classification*" (Clements, 1999b). This study involved a large variety of ages, but five-year-old were indeed involved. *E.g.*:

> *«* (…) *an episode with a kindergarten student illustrates another natural role that imagery plays in geometric problem solving. John is asked to use Logo to draw an open path that has three bends in it. He draws a "box" in which the beginning and ending points do not touch. He explains that he "just thought of all the paths we made. One of the paths had the right number." John visually reviews solutions to similar problems, analyzing each to see if it meets the constraints for the new problem. »*
>
> *(Battista & Clements, 1991)*

Between 1988 and 1990, a Portuguese teacher working at a Pre-K to Grade 3 school conducted Logo programming activities with 52 children aged between 4 and 6 (Correia, 1990). He started by having the children role-play the Logo turtle without using the computer. Then, Logo commands were issued to either a floor turtle or a screen turtle, with the teacher serving as computer operator to enter the children's instructions. In order to allow children to use Logo on their own, he adapted a keyboard (Figure 242), so that children could use it iconically, with single keypresses.

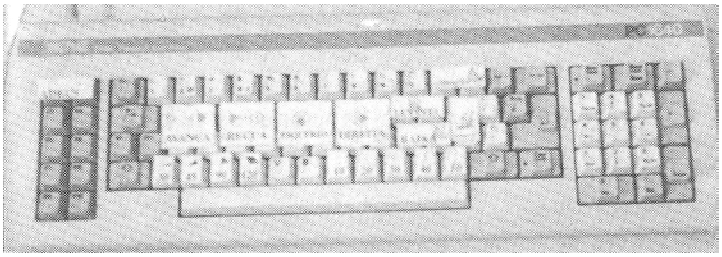


Figure 242 – Rigoberto Correia's keyboard adaption
From: Correia, 1990, p. 15

[272] This research was conducted in 1986, as reported by Clements *et al.*, 1993.

1988 was also the year in which the government of Costa Rica launched its "Computers in Elementary Education" program, which "*reached over 140,000 pre-school*[273] *and elementary school children a year*" (Fonseca, 1999, p. 6). This program was based on a Logo version called LogoWriter (*id.*, p. 8); it "*broke away from the international standard model by focusing on very young children first, and for cognitive rather than computer literacy or computer-assisted instruction purposes*" (*id.*, p. 6). It is also mentioned again further ahead, in the para. for 1998.

By 1989, there was enough public interest on programming with pre-literate children for the SEMERC[274] at Redbridge, UK, to develop a Logo-programming overlay for overlay keyboards[275] (Peixoto & Carvalho, 1990, p. 26).

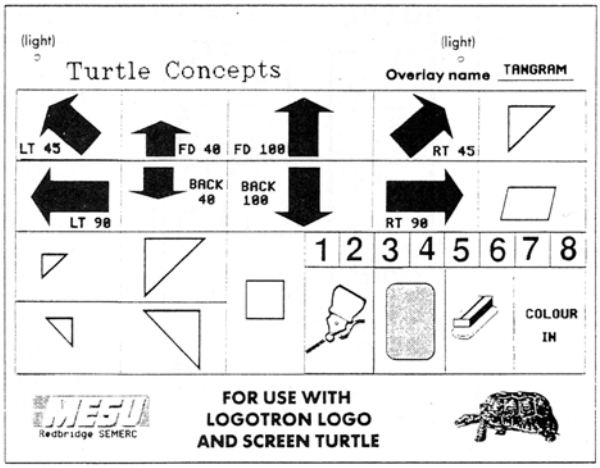


Figure 243 – Redbridge SEMERC keyboard overlay
From: Peixoto & Carvalho, 1990, p. 26

In this same year, a Portuguese researcher published the results of a year-long study of cognitive impacts on 10 five-year-old children within two preschool rooms. Compared with a control group, the study detected a moderate increase in several domains, including time-structuring and mental visualizations (Miranda, 1989). However, some of the major cognitive aspects of the process of programming were not included: "*procedure, bug-debugging, recursion and naming notions were not tested directly, since it was impossible to attain, in the course of the research period, elements for their evaluation*" (Miranda, 1990).

In 1990, a manual was published, designed for kindergarten and grade 1 of elementary school, aiming to "*integrate the teaching and learning of mathematical and computer concepts and skills*" (Cauthen, 1990). It contained 41 lessons, which were primarily based on LOGO (*id.*).

In 1991, a study involved 17 four-year-old and 79 five-year-old children from the suburbs of Buffalo, New York, USA (Weaver, 1991). It explored differences between a floor turtle and an on-screen turtle, in terms of geometric concepts, learning efficiency, and children preference; it also compared children that used Logo with other children, focusing on perspective-taking ability and amount of geometry concepts learned. The research sessions were based on previous geometry-oriented Logo research (Battista & Clements, 1991), with "*two to three computer sessions per week for about 6 weeks*" (Weaver, 1991). The study found no significant differences regarding the two kinds of turtles, and the geometry tests did not reveal differences between students who used Logo and those who didn't, although geometry scores improved overall (*id.*, *ibid.*).

In 1992, a graduation paper associated with the field traineeship of a Preschool Educational Baccalaureate was presented at the University of Costa Rica, on the use of Logo with preschool children (Garro[276], 1992).

It had been previously suggested by observational research with older children that Logo drawing helped students create more elaborate pictures than those that they could create by hand,

[273] One year later, in 1989, a graduation paper by five authors was presented at the University of Costa Rica, regarding the use of computers with preschool children aged between 5½ and 6½; it's only available in a Costa Rican library, so I have had no access to its contents; but it's likely to include LogoWriter use (Ramirez *et al.*, 1989).
[274] The SEMERC (Special Education Micro Electronics Resource Centres) were UK government units for software development and training, but only the Manchester SEMERC survived the termination of UK government funding in 1989 (Granada Learning, 2005).
[275] An overlay keyboard is a flat surface of touch-sensitive cells. The surface is programmed in order to assign specific keyboard functions to pressure within specific areas, and then covered with a paper overlay identifying those areas. As can be seen in Figure 243, this particular overlay was developed by the SEMERC at Redbridge.
[276] As for the paper mentioned in footnote 273, this document is only available in a Costa Rican library, and I have had no access to its contents.

and that new ideas from computational media were transferred to artwork on paper (Vaidya & McKeeby, 1984, acc. Clements, 1999b). Also in 1992, researchers looked into the relation between Logo programming and creativity at young ages, verifying that "*children as young as three years* (…) *show signs of developmental progression in the areas of drawing and geometry during such computer use*" (Clements & Nastasi, 1992, acc. Clements, 1999b).

By 1993, one can deduce from these scattered pieces of research and usage reports that many undocumented cases of computer programming activities with preschoolers were taking place, based upon TurtleTalk, the subset of Logo commands that allow control of the Logo turtle (*vd*. Table 10 on p. 144). In this year, Seymour Papert reported the case of a kindergarten girl that discovered the meaning of zero, by playing with a Logo program "*that allowed objects on the screen to be designed and set in motion.* (…) *The speed of an object was controlled by typing a number* (…)" (Papert, 1993, p. 126). The child got very excited when she realized that when speed was zero, the object stopped – which meant that moving at speed zero is the same as stopped (*id.*, *ibid.*).

In 1994, Logo was shown to increase preschoolers' "*self-efficacy and internal locus of control*" (Bernhard & Siegel, 1994, acc. Clements & Sarama, 2003).

In 1995, the Russian Institute for New Technology-INT jointly developed with the Canadian company LCSI (INT, 2005), an iconic version of Microworlds, aimed at children between the ages of 4 and 8, called PervoLogo, which was described on p. 160. Three Russian researchers presented it to an English-speaking audience two years later, but with no details on actual use by children[277] (Alexandrov *et al.*, 1997).

In 1996, an informal review of the system My Make Believe Castle (described on p. 167) reports on preschool children's interaction with the product, stating that a four-year-old "*loved making his characters turn around and move in the opposite direction, climb up or down a ladder, dance, and jump* (…)[,] *liked recording his own voice for each of the characters* (…) [and] *click on an anthill and helping the ants find their way through the maze*" (Velgos, 1996).

Also in 1997, a study covering three preschool rooms, with 12 to 15 children and two teachers in each room, evaluated the use of Logo word worlds[278] with pre-readers (Hopper & Lawler, 1997). It reported that "*children showed definite growth in letter and word knowledge*" (*id.*, *ibid.*).

There was also a report on the successful use of the Logo version called MicroWorlds by 5-year-old children in a Reggio Emilia infant school[279], to construct "multimedia messages" (Chioccariello, 1997). Regarding the use of programming, the report provides information pointing to the focus on the planning aspects of programming: "*the Microworld interface is very intuitive and consonant with the children's expectation of naming an action and assigning it to an actor that executes it when clicked. This was effectively exploited by the children to add sound recordings, melodies and animations to their scenes.*" (*id.*). To overcome the limitations of textual programming, children would plan the movements to be coded on paper, using arrows to represent the commands to issue, and would also name on paper the various procedures, which were then typed into the computer by the teacher.

In this same year, a paper provided teacher strategies for teaching Logo itself, disregarding integration in other contexts, but starting from off-computer approaches and planning (Turcsányi-Szabó, 1997). It reportedly was the result of accumulated experience over 13 years (*i.e.*, since

[277] A book was published in Russian, on the use of PervoLogo in teaching (Soprunov, 1996), but I have no information regarding its contents.

[278] "*Logo Word Worlds are computer-based learning environments, or microworlds, for inductive learning of language. Children create and manipulate computer animated objects on the display screen by using words of a natural language.* (…) *The main goal of the Purdue-Headstart-Apple Project was to explore the impact of computer experience with logo based Word Worlds on pre-readers*" (Hopper, 2001).

[279] The Reggio Emilia pedagogic model is presented in section 4.1.3, pp. 273-276.

1983/84), working with children aged 5 and 6. Although this approach is not in line with the traditional philosophy on the educational use of Logo, described in section 4.2.2, this paper included valuable remarks about patterns of usage by kindergarten children: it mentioned that they usually do not stick to preset plans, not even their own, and have a tendency of going back to trial-and-error strategies. Seeing as the main goal behind this paper is the learning of Logo techniques, these patterns of usage are seen as something to be avoided, and Turcsányi-Szabó provides interesting suggestions for teachers, such as "*diverting egocentric thinking*" (*id.*, *ibid.*). In passing, she provides accounts of children this young re-using procedures, both their own and written by others, and mentions benefits to reading and writing from children's contact with Logo, attributing these to awareness of syntactical restrictions (due to the need to type commands precisely), "*feeling of writing as a composition in space: letters appearing form left to right, words divided by space, lines continuing under each other*" (*id.*, *ibid.*), immediacy of seeing what is typed, and the drilling effect of retyping words as "*evoking interest in the meaning and function of words*" (*id.*, *ibid.*).

Also in 1997, one finds a first research report on the use of the Valiant Roamer (p. 172), which analyses its impact on the problem-solving strategies of six children aged 4 and 5, after a six-week period of programming experience (Macchiusi, 1997). All children "*clearly demonstrated patterns of typical non-strategic behaviours. Although the children generally exhibited these nonstrategic behaviours, towards the later part of the post-robotic period some of their behaviours reflected strategic patterns*" (*id.*, p. 27), but the study did include considerable focus on the kinds of interactions taking place between the children and the teacher, to conclude that "*teacher scaffolding is an important requirement to assist young children in the development of effective problem-solving strategies and metacognitive skills*" and "*noticeable differences in student's choice of learning strategy are visible only after some quality time and input by the teacher*" (*id.*, p. 28).

An influencial position statement, by the MIT Media Lab, was also made public in 1997, pointing to the importante of developing novel interfaces enabling a better communication between very young children and computers:

> *« The channels of communications between children and computers have been extremely limited: keystrokes and mouse clicks in one direction, text and graphics in the other. By enabling computers to understand and produce gestures and other forms of nonverbal communication, we will enrich the nature of the interaction between children and computers. By the same token, computers that understand verbal and nonverbal communication can open up computing to a broader range of ages and cultural traditions (including non-literate people). That is,* ***children who cannot (yet) type****, can certainly gesture in the direction of their computer, and understand the speech and gesture that the computer returns. »*
>
> *(Negroponte et al., 1997, my emphasis)*

By 1998 there was an increase in the level of use of the Valiant Roamer robot with preschool children, as demonstrated by two documents: a book edited by the company behind the robot, with activity suggestions specifically for its use with preschoolers (Hudson, 1998); and a paper detailing the use of the Valiant Roamer in two child care centers, one in Helsinki, Finland, another in Willimantic, Connecticut, USA (Wright, 1998). These works provide many suggestions and reports about the integration of the Valiant Roamer robot into the preschool curriculum and context.

In this same year, the government of Costa Rica initiated the second phase of its Computers in Elementary Education program, already mentioned in this list regarding its launch in 1988. In this second phase, it reaches "*225,000* [K-6] *children a year – i.e., 30 percent of the country's total elementary school population*" (Fonseca, 1999, p. 6). This second phase of program was based on a Logo version called MicroWorlds, replacing the original version, LogoWriter (*id.*, p. 8). Accounts from other Latin American countries have also surfaced. In Venezuela, for instance, a Logo

adaptation for preschoolers, called Facilito[280], was being used with children aged 4 & 5 for basic turtle commands (Estevez, 1998); a similar adaptation, called DIB, was being used in Argentina (Bergagna, 1998); the father of a 3-year-old Argentinian child with Down syndrome was successfully using a personal adaptation of Logo (Felice, 1998).

Also in 1998, a large study involving over 100 teachers from kindergarten to third grade elementary was conducted on the use of the system My Make Believe Castle (described on p. 167). Unfortunately, while the authors provide many accounts and learning insights involved in the use of this system, they do not provide any information on how many of the children involved were in kindergarten, and the few cases that do mention the current school stage of children are all from the elementary grades (Bearden & Martin, 1998).

In 1999, a mother presented simple games, quite ordinary, which she created for use by her three-year old daughter, and documented the difficulties her child was experienced while attempting to control the computer. It is interesting to follow the progression of the sequence of programs, as the mother describes the control problems and evolution that the child was experiencing: identifying letters on the keyboard, finding the mouse cursor, clicking on a picture, dragging and dropping. The three-year old child wasn't just a passive user: she was involved in the development[281] of the games, suggesting new pictures, new sounds, and other program properties (Tomcsányiová, 1999).

Still in 1999, following the 1998 launch of the LEGO Mindstorms series and various software packages for its use (*vd.* pp. 161-162), reports emerged regarding the participation of preschoolers in the programming projects using Mindstorms products, even though giving little or no details on actual programming methods:

> *« Kindergartners have used LabVIEW and the LEGO bricks to build their town and automated a bus to stop at each house. »*
>
> *(Erwin et al., 1999, p. 1)*
>
> *«* (…) *to teach kindergartners about forces (including the concept of friction), we brought a few RCX bricks and a computer into a classroom.* (…) *After showing them once how to work the* (…) *program, they had no problem making their cars move in all directions and speeds.* (…) *We worked more closely with a few kindergartners, who quickly learned how to do diagram programming. One kindergartner made a little robotic car that would follow you around if you sped up, so would it. He simply had the light indicator brick controlling the speed of the motors, when the sensor no longer read the reflection off of the person, the car would go faster. In general, the kindergartners had little difficulty programming (the earliest we have worked with is 3-year-olds). The panel programming quickly taught them how to think logically and the basics of programming structure. Armed with this, they have little difficulty moving into the diagram programming environment. Interestingly, however, the younger students seem much more interested in the physical construction rather than the programming. If the program basically works, that is good enough. »*
>
> *(id., pp. 8-9)*
>
> *« The RCX Code uses a drag-and-drop interface, and is easy to use and understand. (My 4-year-old wonder boy Isaac can use it without any*

[280] "(…) *in my case, at the preschool level (children from age 4) until the first grade (6-7 years) they work with a Logo util called Facilito (designed by the Educational Computing Center of IBM in Venezuela)*" (Estevez, 1998, translated from the Spanish original).

[281] The involvement of older children in the design and development process of technological projects was being approached in the same year at the academic level (Druin, 1999a, 1999b).

> *problem, and only needs help thinking through the steps of the programs he tries to write.) »*
>
> *(Wilcox, 1999)*

In 2000, following the development the StoryRooms system, aimed at children aged between 7 and 11, for the creation of physical interactive storytelling environments (Alborzi *et al.*, 2000), a research effort was initiated with the aim of allowing kindergartners to program interactions with the physical environment by manipulation of physical objects (Montemayor, 2001), an effort which would eventually lead to the Physical Programming system, mentioned further ahead in this section (see also p. 177).

Also in 2000, there was a report on an observation of 26 children aged between 4 and 5 years using the Valiant Roamer for the first time, basically consisting on the discovery of how the "forward" command key made it move and how this movement depended on the numbers provided (Piper, 2000).

Between 2000 and 2001, children aged between 4 and 6 were involved in a research project which started from LEGO pieces and RCX bricks (*vd.* Figure 102) and developed novel construction pieces and software to support the programming by children of behaviors for those pieces. This project, "CAB – Construction kit made of Atoms and Bits", provided significant contributions at the level of curricular integration of physical construction projects and programming, but its final reports doesn't provide much information on the specific issues regarding the programming process by young children (Askildsen *et al.*, 2001).

In the Logo field, in 2001 were published the results of a major evaluation of a Logo-based geometry curriculum (Clements & Battista, 2001), which included 1,624 students from kindergarten to the 6th grade and their teachers; it employed "*a wide assortment of research techniques, pre and post paper-and-pencil testing, interviews, classroom observations, and case studies*" (Clements & Sarama, 2003). The summary of these results is:

> *« Across grades K-6, Logo students scored significantly higher than control students on a general geometry achievement test, making about double the gains of the control groups. These are especially significant because the test was paper-and-pencil, not allowing access to the computer environments in which the experimental group had learned and because the curriculum is a relatively short intervention, lasting only six weeks. Other assessments confirmed these results, and indicated that Logo was a particularly felicitous environment for learning mathematics, reasoning, and problem solving. »*
>
> *(Clements & Sarama, 2003).*

These results were presented in support of the argument that "*Logo, **used thoughtfully**, can provide an additional evocative context for young children's explorations of mathematical ideas*" (*id.*, emphasis in the original), and that teachers play an essential role in this regard: "*children often do not appreciate the mathematics in Logo work unless someone helps them see the work mathematically. Effective teachers raise questions about 'surprises' or conflicts between children's intuitions and computer feedback to promote reflection. They pose challenges and tasks designed to make the mathematical ideas explicit for children. They help children build bridges between the Logo experience and their regular mathematics work*" (*id.*).

Also in the Logo field, in 2001 two Russian researchers presented a novel version of PervoLogo (*vd.* p. 160), whose first version I have mentioned previously (see the paragraph about 1995, in this section). Still, no information was provided regarding trials with children (Ganichev & Supronov, 2001).

In this year still, two researchers provided some reflections linking the various issues behind developmentally-appropriate educational practice with the usage of Logo programming by children aged between three and six years (Gillespie & Beisser, 2001).

2001 was also the year when the Electronic Blocks systems (p. 175) was presented, along with a study on its use with 28 children aged between four and six years, at an on-campus university preschool. For this study, children "*on average played with the blocks between two and three times during the six days of evaluation*" (Wyeth & Wyeth, 2001). Accounts include children's preferences, misconceptions, and time devoted to the task. An interesting report involves the difficulties children found grasping the relationship between the "*invisible signals passed between blocks and the behaviours of the logic elements*" (*id.*, *ibid*.), and how the use of interacting block structures helped them overcome this difficulty. The researchers also include accounts of frustration problems, albeit few, and report on instances of debugging and reuse of "*code fragments*" (*id.*, *ibid.*).

In 2002, Douglas Clements published a thorough review of various issues on the use of computers in early childhood, including programming topics. Reasoning over previous research, he hypothesized: "*Several studies have reported that Logo experience significantly increases in both preschool and primary grade children's ability to monitor their comprehension and problem-solving processes; that is, to 'realize when you don't understand'* (…). *This may reflect the prevalence of 'debugging' in Logo programming*" (Clements, 2002).

Also in this year one finds the description of an approach to the introduction of robotics and robot programming in early childhood classrooms (Bers *et al.*, 2002), using RoboLab (p. 162), which provides useful methods for technology integration on the part of early childhood teachers. However, in the provided examples the preschool children did not program: instead the teachers developed activities where these children got to use technology in an active manner, but not programming or robotic construction itself. The approach strongly involves children in many technical issues, but reveals a bias against the abilities of preschoolers:

> *« Although they did design the environment for the caterpillar, they did not experience the technical building or programming. These children were only three years old.* (…) *The scale of the LEGOs was too small, and thus required a great deal of fine motor ability, which many four-year-olds have not yet mastered. Due to their young age, it wasn't developmentally-appropriate to engage or expect children to participate in the whole design process of the crane, as their student-teacher did.* (…) [Five-year old] *children were not engaged in either the engineering design of the wheel, or the programming of its movement. However, they were in charge of controlling the contraption and deciding the chronological order of the stages of a tadpoles' life cycle by operating the wheel and its motion. »*
>
> *(Bers et al., 2002)*

Still in 2002 were presented the results of the effort launched in 2001 to allow kindergartners to program interactions with the physical environment by manipulation of physical objects (Montemayor, 2001), now named "Physical Programming" (Montemayor *et al.*, 2002; see also p. 177). Working with 11 kindergartners, aged between 4 and 6 years, the researchers developed a system that does away with a computer screen entirely and relies entirely on the use of physical sensors and actuators, which the researchers called "physical icons" (*id.*). From video-recorded data, these researchers provide information on children's activity patterns and roles. And also this interesting observation:

> *«* (…) [the children's] *main difficulty was in understanding the difference between programming and participation in an already created story. At this young age, children's most common form of storytelling is improvisational*

> *storytelling* (…) *where children freely move in and out of storytelling and "storylistening". We now believe this may be our biggest challenge in supporting children with physical programming. Is there a way to naturally move between programming and participating*[282]*? The magic wand may be only part of the solution. »*
>
> *(Montemayor et al., 2002)*

2003 saw the appearance of several reports on programming in the preschool age group, by using physical materials, as I mention below. Two superficial accounts by teachers also surfaced, on having used Logo with preschoolers in the previous years, in Argentina (anon., 2003).

Regarding the Electronic Blocks system, which had been presented in 2001 (see above in this section, but also on p. 175), a new paper published in 2003 presented the design criteria behind its development (Wyeth & Purchase, 2003), and provided descriptions of the usage of Electronic Blocks by a three-year-old and a five-year-old (*id.*). It also contains more detailed information on the research sessions mentioned in the 2001 paper (Wyeth & Wyeth, 2001).

A new tangible-programming system was presented in 2003, called Topobo (*vd.* p. 178). During its development phase, the researchers had the goal of rendering it usable by children as young as five (Raffle & Garcia, 2003). A small bit of information is provided regarding a single two-hours kindergarten trial (*id.*).

For the LEGO Mindstorms products, during this year some informal documents hint at a social and cultural background where these products are sometimes used with preschool or kindergarten aged children: the announcement of a public talk at a university demonstrating the use of RoboLab (*vd.* p. 162) with children from age 5 onward (Rogers, 2003); and the Web page of a kindergarten, which presents photographs and descrptions of the robotics projects done with children for the schoolyear 2003-2004, including RoboLab programming, but with no details on what were the actual activities performed by the children (Belcher, 2003).

Similar information is found regarding the Valiant Roamer system (p. 172), from two different countries, hinting also at some level of use in preschool environments: a Portuguese paper which I co-authored documents some of its uses in several preschools in the region of Vila Real, Portugal (João-Monteiro *et al.*, 2003); and some photographs providing evidence of its use in a preschool room in Kansas, USA (Witherspoon, 2003). In the field of Logo programming, I found a similar piece of information: at least until July 2nd, 2004, links to several Logo projects done by four-year-olds and five-year-olds were available on a Web page; these links no longer function (Richardson, n.d.).

In 2004, an article described three studies on computer programming that followed children in a kindergarten classroom and a preschool activity room. The kindergarten was accompanied for 2 years, with children divided into two groups (4 children + 5 children), using MicroWorlds Logo and Lego-Logo; the preschool was accompanied for a year, with 13 children involved, without specific groupings, since all children were using MicroWords Logo (however, the data for this classroom is mostly based on 5 children: one 3-year-old, two 4-year-olds, and two 5-year-olds). In the kindergarten, there were 2 or 3 one-hour sessions per week, year-round; in the preschool, the computer was used as any other activity center[283]. "*All children had the opportunity to use Logo multiple times per week as part of their regular classroom curriculum* (…). *Data were collected* (…) *during the time that children were engaged in Lego-Logo activities to show frequencies and proportion of time that children spent interacting with Logo in various ways, as well as various social configurations during children's interaction with MicroWorlds and Lego-Logo*" (Gillespie,

[282] During my field work with ToonTalk programming, which I discuss in chapters 1 and 1 of this thesis, I have witnessed children "forgetting" they were teaching a robot and started playing with it; this observation by Montemayor and his colleagues may offer another perspective on this issue.

[283] *Vd.* section 4.1.3, particularly the parts on the High/Scope model, for a description of activity centers.

2004). The studies were centered on social interactions, rather than on actual programming developments and issues. The article tells how much time children spent constructing, role-playing, etc. and whether they were doing it alone, with a peer, an adult, or a combination of peers and adults. The researchers were satisfied with the level of involvement of children with programming in the kindergarten, where specific sessions were conducted, but not with the level of involvement achieved in the preschool, since only 5 of the 13 children there got involved (*id.*).

At the level of robotics programming, in 2004 more LEGO Mindstorms user stories surfaced, such as an account of a four-year-old having learned how to program a LEGO robot (Rice, 2004).

In Australia, a trial project regarding the use of the Valiant Roamer robot in preschool and primary school settings was initiated, planned to proceed until 2006, "*to establish a developmental scope and sequence across sectors, common language and underlying skill and concept development required*" (Marsham, 2004).

More information was also published about the Topobo system, which had been created in the previous year, as mentioned above. Based on the research conducted in 2003, some activities and ideas were outlined, inspired by Froebel's kindergarten gifts[284] (Raffle, 2004, pp. 74-83). Researchers have also conducted classroom studies with 25 kindergartners, aged between 5 and 6 years (Raffle *et al.*, 2004), but only for three hours, having focused their research on older children. However, they did report that all the children involved in the research associated the Topobo models with "*with their 'familiar knowledge' about animals and machines*" (*id.*); but their kindergarten-related information is limited to this: "*Kindergartners generally programmed only one* (…) [model]. *Some kindergartners puzzled over cause and effect with the programming and playback, while others understood the interface and playfully experimented with creations and storytelling*" (*id.*).

Still in 2004, some user studies for System Blocks (*vd.* p. 177) were presented, including 5 preschool students (Zuckerman, 2004, pp. 86-91), with ages between 3 ½ to 4 ½ years old (*id.*, p. 86). The researcher conducted only one session with each child, lasting between 15 to 30 minutes. The sessions were based around mapping real-world cases such as water flow (Figure 244) or eating cookies onto a system with an input, an accumulator and an output, probing the child's understanding of system dynamics often. "*Out of the five children, three could analogize the moving lights to water, and two could not.* (…) *The three children that could analogize moved on to the cookies example.* (…) *The 4-year-olds successfully simulated the model while describing with joy how they bake the cookies and other kids eat the cookies*" (Zuckerman *et al.*, 2005).

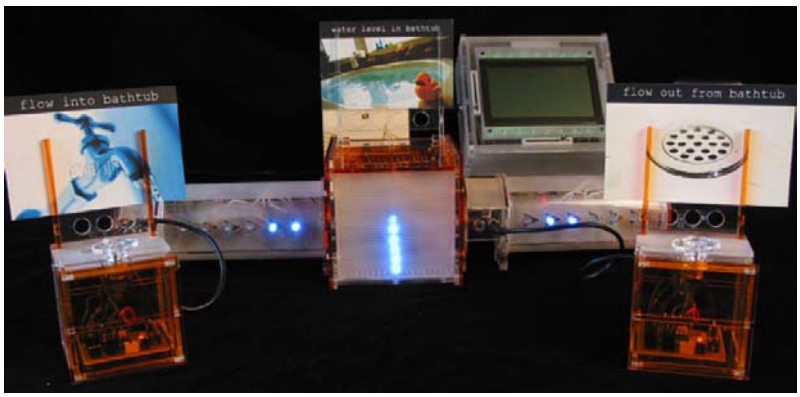


Figure 244 – System Blocks arrangement for a water flow example
From: Zuckerman et al., 2005

So now, in 2005, what information can be extracted from all this history of computer programming with preschool-aged children, overall?

Not much.

There has been a large evolution of the variety of tools and systems at the disposal of educators, but research on educational practice at the preschool level has not seen similar progress.

[284] The description of these gifts can be found in section 4.1.1.

The text-based roots of computer programming were the first and foremost obstacle to the usage of programming by preschoolers. Nevertheless, this obstacle was also the first to be tackled, as described earlier in this section: from an hardware perspective, as early as 1974-1976, by Radia Perlman's TORTIS system, which provided a tangible, non-textual way to program; from a software perspective, at about the same time, by Paul Goldenberg's FASTR system and Cynthia Solomon's TEACH system, which allowed programming to be made using single-keystrokes, thus helping overcome the need to type words.

Since then, developments aimed at helping preschoolers overcome this obstacle were few, until the late 1990s, when various iconic-based or robotics-based programming systems started to emerge, such as My Make Believe Castle, PervoLogo, RoboLab, or the Valiant Roamer. The 2000s have been the stage for the development of more iconic-based systems and several tangible programming systems that allow children to employ various programming techniques without resorting to textual cues, such as Electronic Blocks, Physical Programming, or Topobo.

However, the information available regarding children's use of these new systems is extremely limited. Usually, the documentation reports children's "successful" use of the systems, or describe how children managed to complete some specific activity. And in the few exceptions containing details of the process of programming, such as the data on System Blocks, the reports are based on extremely limited interactions between preschoolers and the researchers developing the systems, who tipically focus on the interaction with older children and approach preschoolers as little more than an extra trial.

Due to this state of affairs, the main corpus of information on computer programming with preschoolers is the set of Logo-based studies conducted since the mid-1970s. This corpus was limited in scope, because the Logo environments, until recently, were constrained in the options available for preschoolers to conduct programming activities. TurtleTalk command subset, for producing turtle graphics (*vd.* Figure 79, on p. 144), as virtually the only simple way for children to enter programming commands and receive feedback, in a non-textual manner.

And this corpus os Logo-based research is not entirely focused on preschoolers; for instance, the largest studies (Battista & Clements, 1991; Clements & Battista, 2001), included children from kindergarten (aged 5) to the 6th elementary grade (aged 11). While considering kindergarteners and their special needs, these studies were necessarily focused on issues prevalent throughout schooling.

The results of such studies have provided quite relevant knowledge, particularly on the quality of students' domain of the subject matter that was studied with the support of Logo (geometry, in the case of the two studies mentioned in the last paragaph). Possibly the most important overall contribution from this corpus of research, is the support it rendered to Seymour Papert's insight in seeing the potential of programming to support the development of new forms of learning and thinking, by demonstrating its effectiveness when "*used thoughtfully*" by teachers (Clements & Sarama, 2003).

Beyond these results, however, the applicability of these findings to typical environments of preschools and kindergartens is limited. These environments, as can be gathered from the sample pedagogic models presented in section 4.1.3, are structurally different from formal schooling. There is a lack of specific research on programming with preschool children, looking at children aged 3, 4 and those that have just turned 5, rather than just those who are almost 6 and about to enroll in elementary school.

At this age level, one finds studies that analysed specific issues, but little information on the overall issue of conducting programming with preschoolers has been provided since Radia Perlman's efforts of 1974-1976. Besides the focus on children aged 5-almost-6, the research efforts with preschoolers often suffered from what Seymour Papert called "*technocentric questions*" (Papert, 1980, p. xiv): researching the impact of Logo programming on children as if it were an independent variable, rather than looking at how Logo programming was used and at who,

specifically, Logo programming was being used with. Several of the research efforts presented in this section simply describe the programming that took place, without looking into the educational processes that are taking place or the educational settings.

> *« Although a venerable body of research on the use of Logo in early childhood environments, including preschool, kindergarten, and first-grade classrooms, has developed since the mid-1980s, none of these studies describes the continuous classroom involvement of young children in Logo-rich environments that Papert envisioned. »*
>
> *(Gillespie, 2004)*

Depending how one interprets the word "continuous", as used by Gillespie in the above citation, "None", may not be the right description; "virtually all" might be more adequate. Indeed there are no studies of continuous classroom involvement of Logo, if one considers "continuous" to mean year-long. But if one interprets "continuous" as the systematic usage of programming integrated into everyday educational activities, not being simply an extra activity, some cases were mentioned above, in the history of previous efforts (Chioccariello, 1997; Askildsen *et al.*, 2001). But indeed virtually all previous efforts fall within Gillespie's judgment. I see them as attempts to assess potential impacts of the **exposure** of children to computer programming, as if it were an environment chemical, rather than a cognitive instrument (*e.g.*, Munro-Mavrias, 1983; Schwartz, 1985; Degelman, Free, *et al.*, 1986; Degelman, Brokaw, *et al.*, 1986; Weaver, 1991). As was mentioned before, regarding mathematics: "*children often do not appreciate the mathematics in Logo work unless someone helps them see the work mathematically*" (Clements & Sarama, 2003), *i.e.*, the learning processes should be at the center of research, not Logo.

> *«* [Programming] *does not in itself produce good learning any more than paint produces good art. »*
>
> *(Papert, 1980, p. xiv)*

More significant have been the descriptive studies, which look into the educational processes and settings besides Logo. These have provided insights on the potential nature of programming hurdles, on possible instruments to their resolution, and on the age-appropriateness of programming activities at young ages (Warash, 1984; Reimer, 1985; Tan, 1985; Miller & Emihovich, 1986; Perlmutter *et al.*, 1989; Correia, 1990; Chioccariello, 1997; Macchiusi, 1997; Wyeth & Wyeth, 2001; Montemayor *et al.*, 2002; Gillespie, 2004).

Still, these contributions have provided only fragments of information on the specific technical and cognitive features involving the use of programming by preschoolers. I believe part of the reason for this may lie in an inadequate framing of the programming activity by the research so far, particularly in the common classifications of programming as an "open-ended" activity or a "problem-solving" activity. These are just two examples of such classification:

> *« It appears that the effect of mediation is greater when children are using Logo as compared to Games software. One possible explanation for this finding could be related to the fact that Logo is an 'open' program presenting problems of a visual spatial nature thus 'inviting' spatial mediation* (...) *and consequently having a more significant effect on children's vocabulary and visual spatial reasoning. »*
>
> *(Klein et al., 2000)*
>
> *«* (…) *even compared to other problem-solving, open-ended computer environments, Logo environments engendered a high-level type of conflict resolution involving coordination of divergent perspectives. »*
>
> *(Clements & Nastasi, 1999)*

I do not wish to dispute the validity of these comments, *per se*. In fact, I have chosen to quote them here precisely because I found them in respectable, quality research papers. But by classifying programming environments as "open-ended" environments or "problem-solving" environments, these researchers are also risking diverting the focus of their readers' reasoning over the data. For instance, a reader whose background lies essentially in the field of educational practice may decide (if lacking programming experience[285]) to consider programming activities as any other open-ended or problem-solving activity he/she is familiar with, in terms of planning and assumptions.

If one considers the kinds of activities educators traditionally think of when speaking of **"open-endedness"**, regardless of age, typical cases might be the assemblage of construction blocks without imposing a plan, role-playing with some script leeway, developing a civilization in a computer game like Civilization III (*vd.* p. 72; Squire, 2004), or even using a computer simulation game, such as The Sims (Maqsood, 2001). As for **"problem-solving"**, the name itself prompts an idea of an immediate target or purpose, an hurdle that one faces and must overcome[286].

Indeed programming can be used for these purposes, and in these ways. But these classifications fail to appreciate the main distinctive features of the activity of programming. To clarify my position, I'll approach each of them in turn.

First, regarding open-endedness: by framing an activity within this classification, one is indeed focusing on issues quite relevant to the process of programming. There is a need for creativity; there is a need for purpose, for the ability to avoid getting lost, be it a global plan, or a momentary interest that one elects to follow, a need for design abilities. For a teacher, there is the need to be able to anticipate potential outcomes or difficulties, to suggest possible paths of development for the activity. In terms of collaboration between learners, as Clements & Nastasi put it in the quote above, there is a need of "*coordination of divergent perspectives*". And many other similar concerns could be listed.

But when using programming in an open-ended activity, other issues are quite relevant. Rather than consider the path to take, and simply maneuver along it, programming requires the awareness of how the maneuvering decisions are going to be made, before or while maneuvering takes place. I am not referring to sorting out a technical issue or discovering a technique – the major issue is the need to ponder on the process of doing. On the strategies for doing, on the information required by those strategies; here, I am not just referring to information that is required by itself or by any possible ultimate goal, but also to that information which becomes required to define and follow up with very process of doing. All these thinking issues and techniques are being disregarded when one dangerously elects to classify programming as an "open-ended" activity.

What about problem-solving? Isn't programming a problem-solving activity? Indeed it is. The same way programming is an open-ended activity.

The above thinking issues involved in programming can indeed be seen as part of a problem-solving process. I mentioned "maneuvering" – towards a destination, certainly. I mentioned "strategies" – towards a goal, certainly. I mentioned "doing" – to accomplish something, no doubt.

Butwhy should those destinations, goals, or accomplishments be restricted to the ones determined before programming takes place? In an honest discussion, the participants are not only trying to present their viewpoints, but learning from each other and facing the contradictions or gaps in their arguments – and the outcome isn't predetermined. Papert put this beautifully when he spoke

[285] In section 4.2.2, p. 284, I addressed the way in which the lack of programming experience may render difficult the perception of its significance and features.

[286] Here, I'm addressing the typical, narrowly-interpreted way to understand "problem-solving". But "problem-solving" can also be understood in a broader, albeit less common, sense: the development of personal approaches and strategies to deal with the unknown and overcome difficulties. I have no qualm whatsoever with the concept of "problem-solving" under this perspective.

of "*new ways of thinking and learning*" (Papert, 1980, p. xiv) – instead of new ways of solving problems.

By classifying programming as a "problem-solving" activity, one is disregarding the requirement imposed by programming to develop strategies to think, to develop strategies to interpret. Indeed, when "problem-solving" one is considering strategies to approach and solve a problem, but not to approach the unknown.

In the research supporting this thesis, my concern has been the encompassing of programming by children, the way to empower them with it, rather than giving them a driver's license for it. As was put in 1960 by the pionners of the educational use of computers, at a time when such use would necessarily involve programming (*vd.* p. 87), "*to use computers better and more wisely by virtue of understanding them first as general tools to be used in reasoning through solutions of problems per se rather than as devices to solve particular problems*" (Alan Perlis, acc. to Katz, 1960, p. 522).

In view of this concern, I didn't want my ToonTalk-based research to be just another piece of limited-scope information on "successfulness of programming by children", as those provided by studies devoted to programming systems other than Logo, as I mentioned on p. 312. I wanted to provide more in-depth data, gathered from larger numbers of children and settings, over longer time periods than just a few brief sessions with limited numbers of children. And to avoid restricting the applicability of the field information to children with some command of general reading or writing, I avoided activities based on any kind of reading or writing beyond the children's own first names. Further, in view of the integration of computer use in the context of preschool rooms, I worked with children aged 3, 4 and 5 years old, rather than just those aged 5 or older, to attain a broader view of the issues involving computer use in preschool education settings. And finally, also in the scope of attaining this broader view, I combined research efforts over different settings: in and out of preschool rooms; working with several children at a time, with just a pair of children at a time, and with only one children at a time; directly conducting activities with children myself, but also just providing guidance to educators, to gain insights on the applicability of my research and ideas by others. My efforts under these directions are described in chapter 1.

In terms of data-gathering and results focus, most research done on preschoolers, since Radia Perlman's early efforts took place, was technocentric, mainly concerned with immediate impacts of the existence of programming than with the actual processes that should guide its use; I wanted my research to pick up on Perlman's focus on the processes of using programming, and evolve from this starting point, by combining the directions pointed by the few later studies that focused on the educational processes and settings besides programming itself (which I have summarized on p. 313). For this reason, my research contributions are presented in four different parts: section 3.4 contains examples of usage of programming techniques to develop other learning objectives; section 6.1 provides a structured view on the hurdles faced by preschoolers in the mental process of programming; section 6.2 provides some insights on issues facing the educational use of programming by preschool teachers; and chapter 1 provides a framework to help preschool teachers integrate or embed programming activities and systems in everyday educational settings.

5. Exploratory activities on preschool computer programming

5.1. Finding a suitable tool

When considering conducting computer-programming activities with preschoolers, their limited ability to conceptualize rules and abstractions (*vd.* section 4.1.2, on Piaget's developmental stages) may be seen, from a broader perspective, as the main issue at stake. However, perhaps the most obvious hurdle regarding the introduction of computer programming concepts in preschool is the absence of basic reading and writing skills. Although this is a purely technical computing hurdle, and icons or logography-based writing can be envisaged as means of expression for the "programmer's" intentions, it seemed to me that in order to better reach to some conclusions I needed a way to allow children to better express their desires.

Using drawing boards as a way to design programs (having a computer educator translate children's drawings into programming, *i.e.*, being a "human compiler"), and resorting to physical, theatre-like plays[287], were possible starting points. But I don't think such approaches would be entirely adequate, since they would not give children the same level of control over a computer-programming system. The theatre plays would require an educator or child to act as "judge", rather than the computer (which however would be in itself a valuable educational activity, programming or not); and both the "theatre" method and the "human compiler" method would introduce a human-level of interpretation of the children's intentions, rather than have each child deal directly with the behavior originating from the computer-interpretation of her commands.

Other possible starting points would be tangible programming systems, or even developing new graphical programming tools, devising them in order to suit my research needs. Both options would require the commitment of significant financial amounts, compared to an approach entirely based on pre-existing software, without custom-developed elements or physical props. Further, pre-existing software can readily be replaced in case the principles behind its design prove flawed or inadequate in mid-research; in contrast, developing graphical programming tools to a level where they are mature enough for field testing can take years. This flexibility in research would be forfeit should I opt to develop customized software or use tangible programming systems. Thus, following a principle of risk-minimization, my starting point involved the identification of software-only, pre-existing, programming systems for children, potentially usable by preschoolers.

Luckily, as can be gathered from sections 3.3.4 and 4.2.3, I was not alone in this desire to bridge the gap between children and computers. And albeit works dealing with preschool children and programming are fairly limited, as mentioned in section 4.2.3, some of the methods and tools being used with older children seemed suitable enough for the initial phases of my research.

Among the various available software-based programming tools, two in particular drew my attention: Stagecast Creator (described in pp. 151-155), and ToonTalk (described in section 3.3.5). Both systems allow the expression of complex programs without any need for mathematical expressions, reading of text or interpretation of sets of icons. A striking difference, however, exists: in ToonTalk, the programmer is immersed in a virtual world (a virtual city), and programs by performing live actions with an "avatar", moving around, picking up objects, and using virtual tools such as a vacuum cleaner (*vd.* section 3.3.5). In Stagecast Creator, the programmer is not part of the environment; rather all takes place in a more traditional interface, a dashboard-like environment (*vd.* Figure 87-Figure 91, pp. 152-154).

As was discussed in section 3.3.3, p. 136, programming languages for children are "*those that were designed to facilitate the metaphorical mapping between programming concepts and typical world views of children*". A crucial element in this metaphorical mapping is the user interface itself:

> *« Ideally, a visual language for novices should completely integrate the programming paradigm and the user interface metaphor. »*
>
> *(Blackwell & Green, 1999)*

[287] Recent research in this direction was branded as "*embodied programming*" (Tholander & Fernaeus, 2003; *id.*, 2004).

ToonTalk's approach of immersing the programmer in a virtual world, with strong metaphorical connections to our own world[288], presents a much stronger starting point for children, under these concepts, than the dashboard metaphor Stagecast Creator, meaning that from the very start, ToonTalk seemed to me to be the most adequate tool for the target age group (3-5).

Besides this general concepts-based evaluation, a set of other concerns, of more practical nature, also tipped the scale in favor of ToonTalk (Morgado *et al.*, 2001): -

- it employs larger control elements, likely providing easier control for children which are still developing mouse-control skills;
- it seemed to be more easily configurable (by developing objects and behaviors within ToonTalk itself) than Stagecast Creator; this could allow me to program situations and elements as required, therefore lessening the risk of having to change the programming environment to suit the research requirements;
- it requires the use of only three mouse skills: moving, clicking and dragging, against the more full array required by Stagecast, which also employs click and hold, and right clicking;
- several of its functions could be reached by pressing single keyboard keys, which could allow me, if necessary, to employ keyboard overlays or conceptual keyboards.

On the negative side, even though I personally liked ToonTalk's overall graphical aesthetics, my opinion was not shared by several people: in an informal inquiry of 4-5 preschool teachers and educators, responses varied between "I think it looks awful and strange" and "I think it's very nice, but children won't like it". And yet, ToonTalk seemed to be successful with children only slightly older than preschoolers, as I could gather from the contact I had with a then ongoing EU research project that employed the programming languages ToonTalk and Logo (Playground Project, 2001). This contradicting information was cause for concern: it was important for children to find the programming environment visually appealing: "*The issue of beauty and aesthetics in the design of interactive learning environments is one that should be taken seriously* (...)*, especially in environments designed for children*" (Sedighian & Sedhigian, 1997). This contradiction meant that selecting ToonTalk involved the risk that children might not enjoy its appearance, which in turn might involve changing software environments, should this occur[289].

At this level, in the informal inquiries I found no objections regarding Stagecast Creator and its aesthetics. But in my view, its main strength is the way it allows a child to devise a story and specify simple behavior rules in a simple fashion. Eventually, however, the need in Stagecast Creator to devise rules for all circumstances that might arise, and the overall look of the application (which, albeit simple, I feared might prove too complicated for widespread usage by preschool children), contributed for my option to use ToonTalk.

Although I am not aware of any research that directly compares the use of the interfaces of ToonTalk and Stagecast Creator, much less with children in my target age group, there has been limited comparative research between ToonTalk and another programming system, called Squeak Etoys (*vd.* p. 146), with fourth-grade Swedish students. The level of control required by this latter system, in terms of fine-motion skills, and its reliance on the interpretation of static symbols, bear

[288] This became quite evident in my experience during the research sessions. In those sessions, children often managed to employ, using ToonTalk, concepts and intentions which they often cannot communicate to the adult researcher or educator. It also often the case that one realizes that the children are mentally dealing with concepts which they cannot easily tackle due to insufficient coordination of their gestures, in terms of interface control. Similar experiences have also been reported by other researchers, *e.g.*, "[Children's] *suggestions indicate that they do not master the language necessary to only communicate verbally with the interviewer regarding issues of programming. However, through the concrete properties of the programming environment ("in there") they may refer to program elements through pointing and short references without knowing a specific terminology*" (Tholander, 2002).

[289] Fortunately, as is mentioned in section 5.2.1, the children responded more than favorably to ToonTalk's environment.

enough similarities with the interface distinctions between ToonTalk and Stagecast Creator, for me to deem this research relevant regarding a selection between the two. It thus can be seen as a late endorsement of my decision in 2000 (it was only published two years later):

> *«* (...) *the children had more difficulties when programming with Squeak than with TT*[290]. (...) *It seemed that pictures and animations were easier to understand than tiles with text.* (...) *Perhaps TT and Squeak are designed for partly different users? »*
>
> *(Eljaala, 2002, p. 79)*

Finally, ToonTalk and Stagecast Creator aren't completely language independent. For instance, ToonTalk employs a talking "Martian" as a help system; also, some behaviors of its controls are language-dependent (the button that makes the bike pump enlarge or shrink objects, for instance, has command letters on it). And several elements on the Stagecast Creator environment are also language-dependent. Having these language-dependent elements in English would be yet another hurdle for Portuguese-speaking children. This also tipped the scales towards ToonTalk, whose European Portuguese version was launched in May 2000.

[290] ToonTalk.

5.2. Finding a style for introducing programming

5.2.1. Focusing the research

As I stated in section 1.3, the objective of my research was the "*exploration of the possibilities and hurdles involved in conducting computer-programming activities with 3, 4, and 5-year old children, in the environment of preschools and kindergartens*" (p. 24). In order to progress towards this objective, I needed to decide what I wanted to record and analyze.

One option, perhaps the one that would most likely result in elements available for immediate comparison and straightforward analysis, would be to bind the situation, deciding on a set of hypotheses and from those establish firsthand the relevant elements to record, disregarding any others. But this line of action was not viable under my research goals: it would require the existence of an interpretation model or theory against which such hypotheses could be set (Alves-Mazzotti & Gewandsznajder, 1998; Figueiredo, 2003). Under my general goal of exploring unknown elements (possibilities and hurdles), I had to abandon this option.

Therefore, I based my research on qualitative data, namely, observations and records (Quivy & Campenhoudt, 1995; Estrela & Ferreira, 2001). These, I decided, would bridge the nature of children's and their preschool educator's interactions with programming.

To do so, I would start by concentrating my attention on children, on their interaction with the programming system and programming concepts. Rather than just analyze how children overcome difficulties of tackling predefined activities, I would rather try to support the development of their programming activities, either directly or by manipulating the overall context of programming activities, and observe the development of their relationship with the programming system and the various concepts involved in programming. For this purpose, I could resort to specific activities, but not impose them for the sake of quantitative data: it would be more important to keep supporting the human relationship with programming and learn from its development.

> *« (...) when any device user says he or she is programming, we should not question whether this activity is "genuine" programming, but instead analyse the user's experience in order to develop a more generic understanding of programming activity. »*
>
> *(Blackwell, 2002)*

Afterwards, I would concentrate on preschool teachers, under the same light, albeit in partly different circumstances: not just as programmers, but as people involved in supporting the development of programming by the children.

Overall, the method of investigation will therefore be one of action research (Silva, 1996), "*a cyclic or spiral process which alternates between action and critical reflection and, in the later cycles, continuously refining methods, data and interpretation in the light of the understanding developed in the earlier cycles*" (Dick, 1999).

5.2.2. The initial session duration assumption

An initial issue regarding the planning of programming sessions was their duration. Regardless of whether sessions should be short and focused, or longer, with more time for children to try out things and explore, in order to plan ahead or at least reason over goals and methods, I needed to know what time-span was viable, with children this young. A very short duration would mean that sessions would have to be more focused, whereas a longer duration would allow more room to explore, to ponder, to get acquainted with the media, and also, "*time to think, to dream, to gaze, to get a new idea and try it and drop it or persist, time to talk, to see other people's work and their reaction to yours*" (Papert & Harel, 1991).

According to the personal experience of my wife, Rosa Cristóvão Morgado, a pre-school computer-activities teacher, and of one of my thesis advisors, Maria Gabriel Bulas Cruz, regarding computer activities with children aged 3 to 5, the expectable duration would be anywhere from 10 to 20 minutes. Their experience was that children would start to get bored, should computing activities take longer. However, programming in ToonTalk bears many similarities to playing a multimedia game, more so than to traditional educational computer activities. And this was an indication that perhaps children might enjoy longer sessions. "*Children, almost universally, will spend longer uninterrupted sessions at the computer than they will in other non-computer activities. This is especially true with lively, animated multimedia titles*" (Early Connections, n.d.).

In May 2000, just after the release of the European Portuguese version of ToonTalk, three preschools in Vila Real (Portugal) agreed to cooperate in my research: "As Árvores" (AA), "S. Pedro Parque" (SPP), and "Araucária" (AR). In each, two children were selected by the teachers, for weekly ToonTalk sessions, which lasted until the end of June, in that same year.

I intended to try out the session-duration assumption by conducting 10-minute sessions in SPP, 15-minute sessions in AR and 20-minute sessions in AA.

However, from the very first session, the assumption that sessions needed to be short proved to be wrong: children loved the ToonTalk environment and didn't feel bored at all! In fact, I felt more time was required for adequately exploring the children's ideas and let them practice ToonTalk skills. Table 28 details the evolution of time occupation.

Session	Date	Start time	End time	Duration
S. PEDRO PARQUE (planned duration: 10 min.)				**Average duration: 19 min.**
1	May 2nd, 2000	11h42	12h03	19 min.
2	May 23rd, 2000	11h37	11h58	21 min.
3	June 5th, 2000	11h43	12h00	17 min.
4	June 8th, 2000	14h23	14h45	23 min.
5	June 12th, 2000	11h50	12h00	10 min.
6	June 15th, 2000	11h35	12h00	25 min.
ARAUCÁRIA (planned duration: 15 min.)				**Average duration: 34 min.**
1	May 30th, 2000	11h00	11h17	17 min.
2	June 15th, 2000	10h30	11h15	45 min.
3	June 20th, 2000	10h45	11h25	40 min.
AS ÁRVORES (planned duration: 20 min.)				**Average duration: 28 min.**
1	June 9th, 2000	15h20	15h50	30 min.
2	June 16th, 2000	10h45	11h20	35 min.
3	June 19th, 2000	10h30	10h50	20 min.

Table 28 – Time occupation along the several sessions

As one can see, the 10, 15, and 20-minute groups turned, almost immediately, into 20, 30, and 35 minute groups. And the 40-minute and longer sessions made me realize that actual 45 minute to 1 hour sessions were most likely viable. (This formed, in fact, the basic assumption for the sessions that took place during the following year's research.) It is worth mentioning that session 5 in SPP was shorter than usual due to hardware problems and session 3 in AA was abbreviated, due to the rehearsal of the end-of-year play. Counting out these sessions, the average duration rises to 21 and 33 minutes, respectively.

5.2.3. Preparations for acquisition of preliminary data and information

Initial issues: intervention approach and children's ability for generalization

In 2000, the research started only in May, as detailed in the previous section. Given the small number of sessions that could be performed before summer holidays ensued, I decided to center the research on two main issues: -

- the children's ability to use ToonTalk's method for generalization[291], given generic concerns about preschool children's limited abstraction skills;
- the results yielded by a using a directed approach for my personal intervention, against a coached approach.

Regarding the first of these issues, generalization, two different approaches were used: one on SPP, another one on AR. Therefore, my approach to intervention, in the role of researcher, was identical in both of these preschools (directed approach). In the third preschool, AA, the coached intervention approach was used.

	SPP	AR	AA
Generalization	Explaining, demonstrating, and practicing.	Creating a situation where it was useful.	-
Researcher intervention	Directing	Directing	Coaching

Table 29 – First year research, detailed by preschool.

At the time, I was considering the first issue to be important for further, more complex experiments with ToonTalk, under the reasoning that the children's inability to use ToonTalk's method of generalization would severely compromise any options of rule programming for generic circumstances. The choice of researcher intervention as a second issue was aimed at helping me decide how I should proceed with the sessions on the following year. Of course, this implies a basic assumption: rather than let the children use the programming environment on their own, and simply observe and record what might occur, I would be personally involved, supporting their experience. This decision was based on existing research with preschool-aged children and computers, demonstrating the advantages of an adult-mediated learning experience (*vd*. section 4.2.1). *E.g.*:

> *«* (...) *findings demonstrate that adult mediation to children using computers enhanced the children's abstract reasoning, analogical and reflective thinking. »*
>
> *(Klein et al., 2000)*

In the **directed** approach, I would propose activities to the children, and then would try to assist them, to the point of conducting them on how to achieve the proposed activities, if necessary (providing explanations or doing demonstrations, for instance). In the **coached** approach, I would let children choose whatever they wanted to do, and simply help them achieve their goals. This could involve explanations and demonstrations, but also mere hints. And obviously, most help would have to be improvised on the spot, meaning that I would often have to reason over a problem alongside the children, a methodology recommended under Papert's educational philosophy (*vd.* section 4.2.2).

An important note is that the trial of using two different approaches was not performed to define which approach would be "better" or "more adequate". Rather, my intention under the overall research focus described in section 5.2.1 was to build a personally-meaningful relationship with both methods, by assessing the feasibility and results of each, in the context of my personal educational abilities. This way, in the coming years I would have a better grasp and anticipation of my performance and conduct, and would be able to intervene in a more effective manner. The

[291] Erasing or vacuuming contents of robot's thought bubble (*vd*. section 3.3.5, particularly Figure 189, p. 209).

option to try out specifically these two distinct approaches was an attempt to build my own style – build up my personal understanding – by reconciling in my personal practice the two major recommendations from the existing research literature. The first can be summed up as: "*if* (...) *high-interest software is used, teachers need to offer plenty of instruction, monitoring and guidance for children*" (Early Connections, n.d.); as for the second, it is rather that "*The teacher is no longer the holder and disseminator of knowledge, but rather a questioner, guide and risk-taker willing to explore, experiment, and incorporate technology into their learning environment*" (*id.*, *ibid.*).

In the course of the following years, I eventually came to adopt an approach of intervention that was mainly under the coached style (*vd.* section 5.3), and these initial trials enabled me to do so with greater confidence and insight, blending both styles according to each particular context and each child.

The detailed reports I wrote after each of the sessions of Table 28 are presented (translated into English) in Annex IV, on p. 475.

The children, their pairing, and the preschool rooms computer environment

To ensure the children's privacy, I will refer to them by their first name initials (Table 30).

PRESCHOOL	CHILDREN	AGES
SPP	Z (boy) and O (boy)	5 and 4
AR	M1 (boy) and R (boy)	4 and 5
AA	J (boy), M2 (girl) and S (girl)	4, 4 and 5

Table 30 – Children identification and ages

The selection of children was another variable that was influenced by the short time available to conduct research before the end of the school year. Specifically, I asked the preschool teachers to select children that had previous experience with computers (freehand drawing, using CD-ROMs, etc.), so that I could focus as much as possible on programming issues, rather than other initiation problems. Regarding children's prior achievements, for these initial sessions I made no particular specifications, which was perhaps an error: in general, the preschool teachers selected children that they saw as intellectually above average. The only exception was the incidental inclusion of S, in the AA preschool. This was fortunate, for the way she overcame her original shyness to devote time to ToonTalk when she got a chance was an event that contributed a great deal to my confidence and reassured me that I was on the right track (the circumstances of all this are detailed in Annex IV, on p. 475).

The computer environments in the preschool rooms of SPP and AA were fairly similar, as detailed in Table 31. The AR preschool was the only one with a separate computer room, isolated from the main preschool activities room.

Preschool	Computer environment
SPP	Windows 98 computer on a table, by the windows, amidst the preschool room. Usually turned off, but children could use it on call, by requesting permission from the teacher (Figure 245, right-side photograph).
AR	About four computers with Windows 98 were located in a separate room, adjoining the gym, which in turn was connected to both the preschool room and the elementary school (under the same building). Children were not allowed in the computer room without supervision.
AA	Windows 98 computer on a table, by the windows, amidst the preschool room. Usually turned off, but children could use it on call, by requesting permission from the teacher. The table was too high for comfortable use with the preschool chairs, meaning that children would use it while standing up, a situation since corrected (Figure 245, left-side photograph).

Table 31 – Preschool rooms computer environments

Figure 245- Computers at the preschools AA and SPP

My option was to work with the children in pairs, rather than one at a time. The reason for this was the desire to introduce a level of social and partner interaction in the computer activity. In doing so, I hoped, firstly, for my research activities to contribute as examples to the preschool teachers of integration of the computer activities, rather than examples of group alienation of the children operating the computer. Secondly, I wanted children to enjoy the benefits of reasoning together over problems and situations, rather than face problems alone and thus get "stuck" more often and eventually quit. And thirdly, the conversations between the paired children as they work together often yield important data. Research has shown that at least with adult programmers, working side-by-side at one computer is beneficial to the end-result of programming: "*This method* [of pair programming] *has been demonstrated to improve productivity and the quality of software products*" (Williams & Kessler, 2000). Its success also strongly requires that both programming partners conduct in a socially-responsible manner, by respecting each other, cooperating, and lessening their egos (*id.*, *ibid.*) – all skills that are quintessential to the social-development component of preschool education. Children-related educational literature is also fertile in reporting the advantages of using the computer in a social manner:

> *« In order for children to reap the benefits of social interaction at the computer, the teacher or adult in charge must design the setting to allow children to interact at the computer. This covers everything from having enough room around the computer for at least two children to making sure that equitable access is afforded all kids. »*
>
> *(Early Connections, n.d.)*

Activity for the directed sessions: the exchanger robot

Directed sessions were based upon a simple activity: programming an "image-swapping" robot, *i.e.*, one that would take two images and then exchanges their places (Figure 246).

This can be achieved by placing two boxes in the ground, so that they are combined into a two-hole box; different images are then placed on each hole (in the sessions, I used a tree and a flower). Upon giving this box to a robot, one floats into its thought bubble, commanding the robot with the mouse. The robot then is made to pick the flower and drop it outside the box. Then, it is made to pick up the tree and drop in the hole previously occupied by the flower. One concludes the robot's programming by making it pick up the flower and place it in the hole previously occupied by the flower.

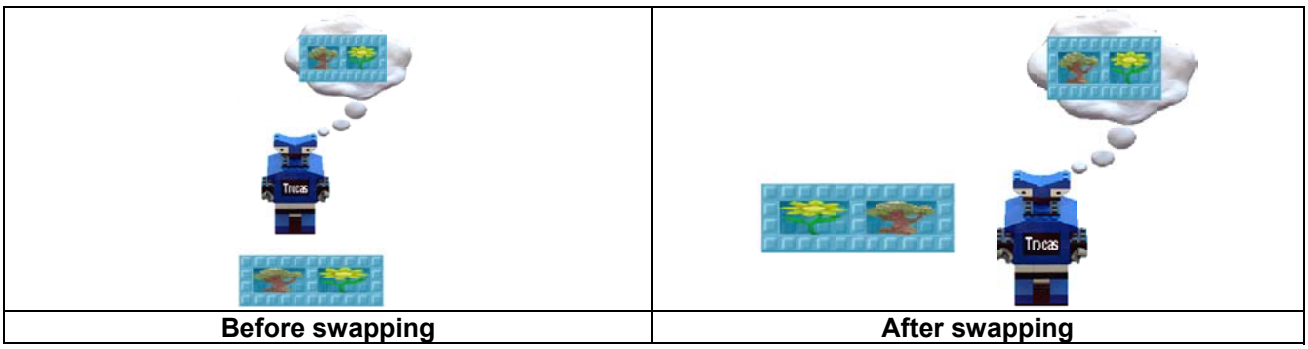


Figure 246- The exchanger robot

This robot can then be generalized by setting ToonTalk's vacuum cleaner tool to "Clean" status, under which it "erases" the surface of images on its thought bubble, leaving only a generic, blank picture. Figure 247 show such a robot (the left one): it will accept a two-hole box with any pictures in the holes. But it will only take pictures: it won't accept number pads, text pads, scales, bombs, or any other ToonTalk elements.

However, there is a simpler and more powerful generalization method (*vd*. section 3.3.5). By setting the cleaner to "Vacuum" status, one can vacuum the entire images from the thought bubble, which also generalizes the robot. The right-side robot in Figure 247 is the result of such a generalization. The difference between the two robots in that figure is that this right-side robot will take any two ToonTalk elements, not just pictures. It will take pictures and swap them, but it will also take number pads, birds, trucks, etc.

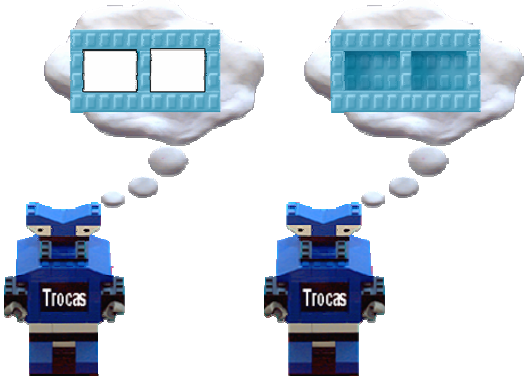


Figure 247- The exchanger robot generalized to different degrees

I elected to try the vacuuming approach with the children, since it is visually simpler to execute and interpret, while still allowing me to evaluate the feasibility of employing generalization.

Initiation of the directed sessions

I went over the initial ToonTalk environment with the children: identifying the helicopter as such, and training its controls; controlling the hand with the mouse and noticing that it made the object under its pointing finger wiggle; identifying robots as such and learning that the cloud near the robot's head is a thought bubble, displaying the robot's thoughts. (The full reports of these sessions are presented in Annex IV, p. 475.)

Instead of exploring the use of the environment tools (vacuum cleaner, bike pump, and magic wand), the children were led straight into robot programming, learning the required skills along the way. For instance, box manipulation was practiced, but in the context of immediate use, right afterwards, to provide parameters to a robot. By-activities (getting pictures from the Images notebook, shrinking the tree), were conducted either by me, in the presence of the children, or by the children themselves.

In the very first session, I demonstrated how to program the Exchanger robot described in p. 326, which during the course of the following sessions was replicated by the children.

This is the full list of actions that were done entirely by the children, while they replicated the first session's robot-programming demonstration. Of these, the only action they didn't execute entirely by themselves was picking up the Images notebook from within the main notebook:

- Pick up and drop a robot, then pick up and drop a box.
- Pick up another box and drop it over the side border of the first one.
- Seek a tree in the Images notebook, and pick it up.
- Set the pump button on "P" (for "pequeno" = "small") and shrink the tree[292].
- Place the tree in a hole, within the two-hole box.
- Seek a flower in the Images notebook, pick it up and place it in the box.
- Hand the two-hole box to the robot.
- Perform the image swapping, in the robot's thoughts.

These actions are quite similar to the ones performed by the robot of Figure 181, on p. 198, while swapping two numbers.

One should note that in a traditional computer programming language, such exchanges cannot be done without resorting to a special abstraction technique. In traditional languages, there is no "floor" on which to "drop" items that are being swapped, so the programmer needs to specifically create a place for temporary storage of the values while the swap takes place.

```
var left_hole, right_hole, temp:string;
left_hole := "tree";
right_hole := "flower";
...
{The following code swaps the contents of
the two holes}
...
temp := left_hole;
left_hole := right_hole;
right_hole := temp;
...
```

Figure 248 – Sample Pascal code for exchanging two values

For instance, Figure 248 shows how such an exchange would typically be done in the Pascal language. The programmer needs to store the contents of the left hole in a temporary variable, `temp`. Afterwards, the contents of the left hole can be destroyed, by storing in it the contents of the right hole. Finally, the previous contents of the left hole, stored in `temp`, can be stored in the right hole.

Table 32 shows what can happen if the programmer mixes-up this sequence (in this table, by making `right_hole := temp` before `left_hole := right_hole`): the resulting program will be accepted by the computer and executed, without reporting any errors. However, instead of swaping the values, it makes both holes have the original value of the left one. Often, such errors can lead to erratic behavior whose origin is troublesome to detect (one needs to slowly navigate along the code, checking the values of all variables, until the source of the error is found).

Right code	Result	Wrong code	Result
`left_hole := "tree";`	left_hole	`left_hole := "tree";`	left_hole
`right_hole := "flower";`	"flower"	`right_hole := "flower";`	"tree"
`temp := left_hole;`		`temp := left_hole;`	
`left_hole := right_hole;`	right_hole	`right_hole := temp;`	right_hole
`right_hole := temp;`	"tree"	`left_hole := right_hole;`	"tree"

Table 32 – Result of issuing instructions in the wrong order

[292] This is not fundamentally necessary to accomplish the exchange of pictures, it is rather a practical necessity: with a large tree, one can't see where it is being dropped, and sometimes it ends on the flower picture or in the wrong hole.

5.2.4. Preliminary data and information

Directed sessions, generalization trials (approach 1: explanation/demonstration)

In the SPP preschool, after programming the exchanger robot, the children tried it out. They saw that it would only work with the tree and flower in the right positions. Then I explained to them that cleaning the images from the thought bubble would prevent the robot from being so picky, and demonstrated the method.

This approach yielded no effective results whatsoever: the children only got confused. Therefore, I let them simply play around with the ToonTalk elements (trucks, bombs), rather than attempt pushing the concept through.

On the following session, a theatre-play approach was used for presenting the explanation, using the following physical material as stage props: two square baskets, similar in color and size to the ToonTalk boxes; an A4/Letter sheet of paper on which I drew a thought bubble with a ToonTalk box (Figure 249, left-side image); two paper squares, one with a tree, another with a flower, attached to the thought bubble's box with scotch tape (Figure 249, left-side image); two empty A5 sheets of paper (half of a A4 sheet); six markers: 2 black, 2 yellow and 2 green.

Figure 249 – the A4 sheet for the robot's thoughts (with scotch tape) and the A5 sheets for the blue baskets

The robot's generic behavior was demonstrated to the children again, who were completely puzzled over it. I then had them initiate the "robot" theatre play. The children used the markers to draw a flower on the blank A5 sheets, while I drew a tree (Figure 249, right-side image). One child would play the robot, holding the A4 sheet with the thought bubble over his head; the other would fill the plastic baskets with the A5 sheets with the pictures and hand them to the "robot". This way, the child playing the robot would have to check his "thoughts" content before proceeding, which I hoped would help clarify the robot's behavior.

Only one issue was detected: the child presenting the baskets would have a different perspective of left/right than the robot-playing child! I overcome this problem by acting as "in-between" trader, turning the baskets around; this would ensure that both children had the same left/right perspective.

All worked smoothly: they would check the thought-bubble sheet, and the robot would only exchange the pictures when they matched. However, when I pulled off the taped pictures from the thought-bubble sheet, leaving only the drawing of an empty box, a curious behavior ensued: the robot-player started to exchange the images continuously, without looking at the thought bubble to check their applicability!

While this would indeed be the actual behavior of a ToonTalk robot under those circumstances, the child's lack of understanding of what was going on, just previously, makes me doubt that he was interpreting the contents of the thought bubble as rules for the robot. In my view, he was just mimicking the apparent robot behavior. Clearly then the concept of "entry parameters" or conditions, while clear for concrete examples, was completely disregarded for the abstract case of generalization: **The children were not interpreting the behavior of the robot as the result of rules, but only in terms of its observed behavior**.

I thus deemed this line of inquiry as unlikely to produce successful results, in terms of children's understanding of their involvement in the programming process. I find it plausible that these children would eventually understand the concept, if they practiced it sufficiently while I provided adequate support. But to me, this didn't seem likely provide an approach usable in preschool education in an integrated, immersive way, under the guidelines described in Table 25, on p. 281. Therefore, **I decided that I would no longer follow it in my upcoming research**.

However, anyone wishing to further probe this abandoned line of inquiry might want to consider the following further experiments about generalization. Supposing that the flower-tree exchange was to occur within a 3-hole box like [flower | tree | truck], instead of a two-hole box, and that the generalized version of that 3-hole box, after vacuuming, was [| | truck], would the children, under these circumstances, check the thought-bubble while playing the robot game? The reasoning for conducting such an experiment would be that in such a case, the robot-playing child should only proceed with the truck in place. Would he/she if a truck was still there even though this robot doesn't use it for anything? One might also wish to probe further with a similar situation, using a mutable element (for instance, a radio-controlled car that might be driven away, or a small PDA computer displaying a changeable image) instead of a static, paper-drawn truck. Would the children check the presence of that element, since they probably could be made aware that paper doesn't disappear or change, but the mutable element might?

Directed sessions, generalization trials (approach 2: usefulness)

At the AR preschool, whose sessions started after I had already completed two at the SPP preschool, I wanted to use a different approach. As shown in Table 29, on p. 324, this came to be what I called the "usefulness" approach. The idea for this approach came when a child expressed his wish to make a robot exchange a truck with a bomb, after having already programmed a robot to exchange a tree with a flower (the report giving the full description of the session where this occurred is presented in Annex IV). I then decided to try and have children program robots to exchange flowers and trees, and then robots to exchange trucks and bombs, to see if I could leverage from these and present generalization as a time-saving and effort-saving technique.

After conducting the sessions using this approach, I observed that the programming of robots with distinct entry parameters **allowed children to more easily understand the robot-programming concept**.

But it also allowed me to present, in a clear manner, the case of both robots, in fact, doing the same thing, except being too picky. The vacuuming method of generalization was then presented simply as a time-saver and effort-saver, allowing me and the children to avoid the pickiness of the robots.

My expectations were for this approach to yield more success than the one used at the SPP preschool, but the actual level of success surprised me: **children saw this as obvious, as a perfectly natural way to do things**. Even though the experiment ground was extremely limited, due to time constraints (only 3 sessions), this nice result, with two children (a 4-year old and a 5-year old), raised my expectations regarding the possibility **of larger-scale use of ToonTalk for definition of rules involving generalization**.

However, in the following years I ended up not following up on the concept of generalization. While conducting further sessions in the following years, I acquired an increased small-grained awareness of more basic hurdles and problems regarding the use of programming with young children, both on part of the children and on part of early-childhood teachers. Under the priorities described in section 5.2.1, I then decided that I should further focus my research on acquiring knowledge regarding those basic hurdles and possible techniques to tackle them, rather than explore this more advanced issue of using generalization in programming. Nonetheless, I believe that the provided data establish the feasibility of researching the use of generalization in programming, with children younger than 6, by employing the "usefulness" approach.

Coached sessions (initiation and developments)

In the coached sessions, which took place in the AA preschool, the children could decide what they wanted to do, and only sporadic suggestions were presented. As a result, a much larger focus on the interface tools occurred, due not the least from them being some of the most visible elements in ToonTalk. Birds and nests were also prime attention targets, likely because they provide highly animated, amusing activities on the screen. In the initial session, the children freely explored these basic activities.

In the second session, I introduced the concept of robot programming. One of their options was to propose programs related with the tool manipulation activities they had done previously: J wanted to make a robot to vacuum up the room. Since this cannot be implemented in ToonTalk, I responded to the challenge by suggesting to him the programming of a robot for vacuuming box contents. This eventually became a robot specialized on the vacuuming of boxes with nests. On her part, S decided that she also wanted to make a box-vacuuming robot; only hers would be specialized on vacuuming up bombs from boxes.

In contrast, the child M2 wanted to make a tree-chopping robot! Although I could simulate this quite easily, using small pictures to crop a larger tree picture, J had unintentionally limited access to the Images notebook[293], so M2 ended up using, as an alternative activity, my unoriginal improvised suggestion of an exchanger robot, like the one used in the directed sessions (at least it involved a tree).

These two initial sessions were the only coached sessions I managed to conduct entirely, because the third and last one coincided with the rehearsal day for the end-of-year party. However, on this last, very short session, I still discovered that S, while not being too active in the previous sessions (J and M2 more or less took over control of the mouse), had enjoyed playing around with ToonTalk to the point of not leaving it to eat cakes and refreshments along with the other two children. Instead, as soon as the other two children left the computer, she took advantage of the opportunity to be on her own and started creating birds, moving objects around, enlarging and reducing them… Playing, plainly.

The main advantage of this coached approach, it seemed, was the greater control it gave children over the manipulation of ToonTalk's objects and tools, resulting in an overall greater ease of operation. However, the robot-programming experiments were very simple, and therefore mostly inconclusive. It was encouraging for me, however, that even a child that apparently was displaying little interest, like S, would prefer playing with ToonTalk over cakes and refreshments. It was also very encouraging that the two more active children, J and M2, would be very excited while at the same time very focused on ToonTalk. Therefore, I decided that **this coached approach seemed promising enough for further, more lengthy experiments to be conducted over the following years**.

Directed sessions: other developments and information

On both preschools where directed sessions were conducted, SPP and AR, the robot programming and generalization trials were done on the initial two or three sessions. Therefore, I tried to collect further information on children's reactions to the several ToonTalk elements. I then introduced them to birds and nests, to check out their responses. Since these children had already been introduced to robot programming, bird use was combined with robot programming, to achieve more dynamic (and hopefully, enticing) results.

As expected, birds were a success: children loved playing with them, given the rapid response and animation.

[293] As described in Annex I.

On the AR preschool, where a single robot-and-bird session was conducted, M programmed a robot that would send boxes of robots, endlessly, into a bird's nest. And, most strikingly, **he had the intuition that by copying a nest, its bird could carry things to both copies**. This was a surprising and most encouraging result.

On the SPP preschool, I had the opportunity to conduct three sessions with birds. On the first one, I simply went over the bird use as carriers. Their behavior was explored (or should I say "tried out") by the children, by handing different objects on them. Birds were also used to clean up the room, carrying stuff into their nests.

For the following two sessions, only Z was available, and therefore he benefited from having more time available on his own, as well as from my undivided attention. On the second session with birds (only 10 minutes long, due to hardware problems), I introduced Z to the concept of working with nests placed in a robot's box (entry parameters). On the third session, I told Z that he could use an exchanger robot, similar to the one he had programmed previously, and combine it with birds. So he programmed a robot that would pick a flower from a box and give it to a bird in another box. This allowed me to suggest to Z that he could place the bird's nest (now with a flower on top) in the box hole where a flower was expected. And Z quickly realized, as I confirmed by questioning him, that this would allow the robot to keep juggling the flower endlessly.

Finally, I suggested that he could create a second robot, which he did (copying it with the magic wand) and then cross the parameters between them – therefore creating the effect of two robots sending a flower to each other.

Although I had to perform this last, more complicated procedure myself, he was following it with keen interest – not like the previous puzzling situation of generalization at all. This final, briefly appreciated result also encouraged me to pursue ToonTalk activities with young children, with some favorable expectations regarding the degree of complexity they might achieve on a larger number of ToonTalk sessions, spread throughout a school year.

5.3. Outlook of the exploratory research activities

5.3.1. Settings for the exploratory activities

As I described in section 5.2.3, field research activities were initiated in May 2000, upon the release of the European Portuguese version of ToonTalk. Since then, there were 7 major settings for these activities, detailed in Table 33, below.

Setting	Period	Description	Groups of Subjects (Age | Gender)
1	May/2000 to Jun/2000	Inside the preschool room, two children at a time, initial trials.	(5M, 4M), (4M, 5M), (4M, 4F, 5F). **Total: 7**
2	Feb/2001 to Jun/2001	Outside the preschool room, 6 children (in pairs) at a time. The pairings would be informal (*i.e.*, the children would form the pairs themselves). The groups with 3-year olds were conducted separately.	(4F 3F), (3F), (5F 4F 5M 5M 5F 5M), (4M 4F 5F 5F 4M 4M), (5M 5M 4M 4F 4F 5M). **Total: 21**
3	Nov/2001 to Feb/2002	Identical to setting 1, but benefiting from acquired experience. Interrupted due to lack of time to perform the sessions.	(4F 4M), (4F 5M), (4M 4M). **Total: 6.**
4	Oct/2002 to Mar/2003	6 computer-activities teachers received specific training and conducted ToonTalk activities in one preschool room each, once per week. There was a weekly meeting for coordination and development of new activities.	Number of children involved in each preschool: 19, 11, 14, 21, 22, 12. **Total: 99**
5	Feb/2004 to Jun/2004	Inside the preschool room, one child at a time, exclusively 3-year olds	3M, 3M, 3F, 3F. **Total: 4**
6	Jan/2004 to Jun/2004	Inside the preschool room, activities conducted by 2 trainees of the 2nd-year degree in Early Childhood Education, without researcher supervision, (either on-site or off-site), during the entire period. Children divided into a larger group and a smaller group.	Large group: 10 boys, 7 girls Small group: 2 boys, 2 girls **Total: 21**
7	Jul/2004 to Aug/2004	At a home in Medford, MA, USA, a five-year old boy attended ToonTalk sessions conducted by an adult from his family. I had no direct contact with either the boy or the adult, other than information provided before and after the sessions, by e-mail.	5M **Total: 1**
			TOTAL: 159 children

Table 33 – Research settings

On most settings, my intention was that 5-year olds and 4-year olds were not to be segregated into specific groups, but rather work together in pairs. However, in setting 6 some instances of age groupings did occur. And in setting 4 they were in fact recommended, since I wanted to minimize the risk of teachers letting the 5-year-olds do almost everything, out of a desire to "show results". As for the 3-year olds, altough some exceptions to this occured in setting 4, typically they were not paired with older children: when they took part in the activities, they were paired among themselves or worked alone.

5.3.2. Setting 1 summary

Due to the short time before summer holidays ensued, the initial trials were set around specific themes that I could use as a basis for preparing the research in following year. This setting was detailed in section 5.2. In summary:-

- Session durations from 20 to 40 minutes are viable, possibly even longer ones.
- Generalization in programming was achieved, employing a "usefulness" approach.
- Two approaches to teacher/researcher intervention were assayed:-
  - Directed: children were suggested activities, and then tried to achieve them, following the teacher's (or researcher's) hints. The teacher/researcher would query the children, debate their interpretation of events and help them decide what to try out.
  - Coached: children decided what they wanted to do; only sporadic suggestions were presented. This was also the general rule for teacher/researcher intervention on queries and discussions.
- Activities in the coached sessions were simpler, but children achieved greater autonomous control over the programming environment: moving objects, resizing, erasing, etc.

5.3.3. Setting 2 summary

Children came in groups of 6 to a University room, set aside for this purpose; they would play in pairs, in 1-hour sessions (actual computer use time would vary, but the usual duration was around 40 minutes). The activities were conducted in mixed groups of ages and genders. Three of the children were 3-year olds, and those were in two separate groups.

With all groups of children aged 4 and 5, the coached approach from the previous year (setting 1) was used, for the reasons explained in section 5.2.4. I would frequently provide suggestions on new paths to explore, or even propose activities, but always with the agreement of the children. Regarding the children aged 3, given that my trials in the previous year had been with older children, the directed approach was used, thus reproducing the trials that had been conducted with children aged 4 and 5. The general description of these sessions is provided in Annex V.

At the cooperating preschools, a meeting was held with the parents of all enrolled children, and no selection was performed: the children that took part were all whose parents came to the meeting and expressed their willingness to have them participate.

With 3-year olds, major aspects were:-

- all children managed to program an "exchange" procedure (ToonTalk robot) as early as the second session, following hints and suggestions; at least one child programmed several different robots;
- children made reference to ToonTalk knowledge from previous sessions and used it (*e.g.*: the notion of "tidying up" things before giving them to robots, or that turning pictures around turned off their behavior);
- children managed to get oriented in the ToonTalk environment, and enjoyed using it;
- there seemed to be a noticeable gap between what 3-year olds could understand and their limited ability to make use of it due to mouse-control woes.

These observations, albeit limited, were combined with the reports of teachers involved in setting 4. Together, they provided the basic insights behind the sessions focused on 3-year olds, in setting 5. The major contribution of this setting came from the sessions with 4-year and 5-year olds: the establishment of specific hurdles facing children's comprehension of programming. These hurdles are presented in section 6.1.

5.3.4. Setting 3 summary

These sessions were performed in the fashion of Setting 1, but always using the coached approach. The focus was using the experience from Setting 2 sessions on children without previous ToonTalk experience. Only very few sessions took place, because other pressing professional demands on my part were making it impossible to continue, as can be deduced by the lack of enough adequate records of these sessions. Even so, they served to further consolidate the acquired experience, and to try out some novel approaches, such as project development over several sessions.

Children were selected by the preschool teacher, on the basis of being the ones more likely to attend regularly (*i.e.*, maximizing the chances that they would be present at the preschool at the session schedule). It was emphasized to the teacher that she should not simply pick the "brightest" children, something to which she agreed.

An interesting fact, however, stands out: some of the children on the same preschool had taken part on Setting 2 sessions at the University, the year before; ToonTalk was installed on the preschool computer, but its use was scarce, if at all (and without any adult assistance whatsoever), because the teacher did not know how to use ToonTalk. And yet, several children came to me and could correctly explain the sequence of actions used to build a house, as well as mention other aspects of the game. This was a demonstration that children had understood the process, because not only could they reproduce it, but they could explain it to other children that hadn't used ToonTalk before. But it also shows that they had enjoyed the sessions, because there was a fair amount of enthusiasm: children wanted to try out what their colleagues had explained, even though they had not been selected by the preschool teacher.

5.3.5. Setting 4

Six preschool computer-activities teachers, involved in the ICEI[294] project, received ToonTalk training, before setting to use it on one preschool room per week each. All children on these rooms took part. In the first session, the teachers let the children get acquainted with the environment. Then three different approaches were suggested to the teachers[295]: -

1. "Straight into programming". Or "programming as the basic activity". At the second or third session the latest, robot-programming activities would be suggested, and other manipulation activities (enlarging, vacuuming, etc.) would only be practiced as necessary.
2. "Automated behaviors". Or "programming as an extension based on automated behaviors". Children would explore environments based on cause-effect elements, like joining a truck, a box and a robot to build a house; playing with automated sensors; giving objects to carrier pigeons that carry them to their nests; combining properties and see the resulting behavior of objects.
3. "Acquaintance first". Or "programming as a development of generic activities". Children would gain basic control over the programming environment, enlarging and shrinking objects, copying them, customizing the looks of the environment, and using it as a place for conducting usual kindergarten activities: pattern matching, knowledge of professions, of animals, of cookery, etc. Programming would only be introduced afterwards, integrated in the environment.

[294] ICEI, "Informática em Contextos de Educação de Infância" (Computers in Early Childhood Education Contexts) was a subproject of a larger Digital Cities & Regions project, "Trás-os-Montes Digital." Several computer-activities teachers would spend each day of the week at a different preschool room, conducting computer activities with all the children in that room. They would return once a week to each preschool room, for the duration of the school year, and also provide computer training for the preschool teachers, in some evening training sessions (Cruz *et al.*, in press).

[295] Mostly, results from these approaches were inconclusive, because approaches 1 and 2 proved more demanding than expected in terms of teacher preparation, training, and time required by the teachers for activity design and set-up.

As the sessions went on, the reporting process was tuned (explaining to teachers what information was relevant), and weekly meetings have helped me identify several issues: -

- Teachers themselves came across some of problems also faced by children (programming without a purpose, controlling tools, etc.);
- Teachers actions tended to shift towards their beliefs ("The 3-year olds are too young, I haven't used it with them"; "They are too young to program, I just did mouse-control"), persistence and examples were needed to keep them on track, and even so most did not;
- Teachers would often see ToonTalk itself as the goal, hands-on examples were required to show them how to fit ToonTalk into the existing kindergarten themes and activities, and maintain research goals.

Initially, the idea was for teachers to conduct ToonTalk sessions throughout the school year and provide weekly reports on their progress. However, after a few months, it had become obvious that in most cases the teachers were not feeling comfortable enough with their ToonTalk activities, and rather than expand them they would stick to tried examples, such as house-builder robots and swappy robots. Children were also not benefiting from further pursuing ToonTalk activities in these conditions, since from the point of view of some of the children, the activities in ToonTalk seemed pointless (this is accounted in the annexes VII & VIII).

Under these conditions, it was agreed with the teachers that they could shift their focus towards other computing activities, not focusing entirely in ToonTalk. As a consequence, although the computer-activities teachers and the preschool teachers reported that children still used ToonTalk from time to time, in most cases the teachers stopped supporting the development of children's activities in ToonTalk altogether.

There were two notable exceptions, however:

- in the Mondrões preschool, there was no further robot programming, but the computer-activities teacher started developing the theme "professions" and this led to the customization by the children of a ToonTalk city, where each house would match one of the professions that were being explored in that preschool. This exploration included much more besides ToonTalk itself: cardboard boxes had been turned into houses, for instance, and a floor turtle had received role-playing "shells" in order for children to program its movements from house to house;
- in the São Pedro Parque preschool, the teacher was enjoying the children's progress in ToonTalk, the large number of tiny or not-so-tiny achievements they were making in each session, and above all the eagerness demonstrated by most chldren to use ToonTalk. So she kept developing programming activities, albeit never moving on from the examples and basic robot suggestions I provided (house-builders, house-decorators, house-blowers, box-cleaners, etc.). But she was able to feel confident enough to experiment and introduce into the ToonTalk activities elements from other activities taking place in the preschool, something she had done from the start in terms of freehand drawing, but not otherwise. This was a different kind of thematic integration into the preschool context.

These exceptions were most valuable and allowed me to realize first-hand the importance of providing teachers with hands-on examples of how to integrate ToonTalk in the preschool context, and that the very basic blocks of programming itself needed to be mastered, in order for teacher to be able to move beyond basic sample programming and on to taking advantage of the children's interests. These insights and lessons are discussed in section 6.2.

5.3.6. Setting 5

In the previous settings, my experience with 3-year old programmers had been very limited. My only direct information sources were the few sessions with two 3-year olds in setting 2, who both had previous computer experience (albeit not ToonTalk experience) and some summarized reports from the teachers in setting 4. I decided that I needed to further improve my contact and experience with 3-year olds, and so this setting was entirely focused on this. The sessions took place in the Mondrões preschool, because the teacher there had previously worked with my wife and welcomed the opportunity to have me conduct that research there. It is a small preschool, and the sessions were conducted individually with all the 3-year olds, which were just 4 children.

For four months, I experienced first-hand the issues, problems, and achievements of these three-year olds in programming: the amazement of a girl upon seeing that the "toys" that she had left "on" in the ToonTalk session, were, in her words, "still working!?!"; the difficulty in achieving (with their hands controlling the mouse) the execution of an intention they had clear in their minds; the joy and amazement at seeing that a programmed robot did send back the ball it was given; the extremely fast switching of mind focus between different goals, objectives, and points of interest. All in all, these sessions provided little data that was significantly different. But they did help me refine my reflections and interpretations of the data I had so far collected. They also allowed me to experience how the issues described in section 6.1 presented themselves in three-year olds, something I had only glimpsed until then.

5.3.7. Setting 6

After a one-semester course on computer use in preschool, which included the development of a ToonTalk activity for preschoolers, two second-year students from the baccalaureate in Early Childhood Education at the University of Trás-os-Montes e Alto Douro, Portugal, embraced my suggestion to try out their planned activity in the following semester. It would be part of one of their traineeships, which was rendered possible because their allocated traineeship preschool for the semester had already been developing, since the start of the school year, a larger project (Teixeira *et al.*, in press), under which several activities were already being conducted, matching their planned activity ("professions"). This was the preschool room at Parada de Cunhos, where there is a computer which is usually turned on and accessible to children throughout the day. Having been their teacher in the previous semester, I had provided thorough advice, support and guidance during the development of their activity plan. Before they initiated the actual sessions with children, I provided them with some extra pointers, but let them conduct the sessions on their own, with no further contact. I did provide very occasional phone counselling on technical ToonTalk issues, but I have provided no extra support or guidance on educational issues or on how to conduct the sessions. The students, named Irina Brandão and Liliana Miguéis[296], divided the children into two groups. With the larger group, their aim was simply to explore ToonTalk generically; but with a smaller group of 4 children they intended to try out their planned activity. They proposed it to the children (as mentioned above, under a more encompassing project).

Overall, it was a huge success. The larger group enjoyed ToonTalk but didn't develop any particular novel situations. But the smaller group demonstrated that there is a large potential in a focused, context-integrated approach. The children started exploring the possibilities opened up by the world they had developed. The various professions exchanged items, using ToonTalk birds: cakes, flowers, pine trees for planting, etc.; children would come accross situations where a parcel was wrongly delivered, and would interpret the situation and decide on how to correct it. They set up and used 12 communication channels (birds), used them, and explored in a rich setting how different professions would require products and services from one another.

The final report from these activities, complete[297] with pictures, is provided in Annex IX.

[296] Another student, Mónica Pereira, also took part in the planning, but did not participate in these sessions.
[297] The report included a CD-R with pictures, ToonTalk cities, and crash dump files, which is not provided in this thesis.

5.3.8. Setting 7

At a home in Medford, MA, USA, a five-year old boy, N, attended ToonTalk sessions over three weekends, conducted by an adult facilitator from his family (F Berthold). I had previously provided this adult with suggestions on how to introduce ToonTalk and use it with N. The full reports of these sessions are provided in Annex X.

The adult followed the approach of demonstrating to N how to do a particular thing, by doing it himself, and then letting N do either a personal version to complement it, or do part of the main goal.

From initial customization activities, they moved on to robot programming using pictures and then numbers, since N was already familiarized with digits. The adult soon wanted to show N the potential of computer programming, and the sessions became even further oriented around this master-apprentice organization, with N watching as the adult developed a program, but being involved in the discussion and even doing small pieces of the larger program.

Altough it is doubtful that N would have been able to develop these programs on his own, he managed to follow the reasoning behind the development of the complex programs, discuss approaches to use, participated in planning decisions, and overall felt involved and followed along in a process that was always a little ahead of his own abilities, but never too much.

> *« I'm particularly excited by the fact that this is the first time that N has taken a creative interest in what we're doing it. It suggests that he's both become sufficiently comfortable with the medium to imagine what it can do and remained sufficiently interested to want to impliment it. »*
>
> *(F. Berthold, August 7th, 2004, after showing N how to program a robot)*

6. Programming isn’t magic

6.1. Conceptual hurdles for children while programming

6.1.1. Of hurdles

A common theme throughout the literature is that, at least for adults and adolescents, "*programming, even at a simple level, is a difficult activity to learn*" (Bonar & Soloway, 1983). Echoes of this reach the children programming community, as mentioned in sections 2.4.4, 3.1, and 3.4, when both children and teachers find themselves constrained by what their limited time and expertise allows (Hoyles & Noss, 1999). But there are fundamental differences in goals between teaching programming at a professional level, to college students or adolescents, and simply being able to use it for educational purposes other than programming itself, as is the overall goal with children and their teachers. Children and teachers don't need to control all aspects and details of programming. Rather, they need to be able to take advantage of it, and much as possible, avoid getting constrained.

With this in mind, in section 3.4 I presented several programming techniques and computer science topics, and provided examples of how they could be adapted to a preschool education setting. The purpose, in that section, was to allow educators to try and use different concepts, thus enriching their portfolio of "programming recipes" and, hopefully, achieve a richer development of both the educator's and their children's programming skills.

Those adaptations were partly inspired by actual activities conducted with preschool children, partly devised so as to be feasible for such children. This notion of feasibility was a consequence of the experience acquired in the programming sessions conducted with preschool children and preschool teachers, described in chapter 1. Further details on the possibilities of adaptation and integration of computer programming in preschool education settings are provided in chapter 1.

An important element contributing to these efforts is the awareness of specific difficulties that preschoolers (and their teachers) face when trying to employ computer programming techniques. This awareness was also a result of the aforementioned sessions, and is the subject of the present chapter.

One advantage that came from having conducted coached-style sessions in the 2nd, 3rd, and 5th research settings (*vd.* sections 5.3.3, 5.3.4, and 5.3.6), rather than directed sessions with predetermined activities, was the occurrence of a larger set of unexpected circumstances to overcome, which provided a richer field of observation and interaction for harvesting awareness of children's difficulties, from my personal perspective as a person with a technical programming background.

The hurdles presented herewith are a novelty in that they are associated with specific cases of programming sessions with children aged 3, 4, and 5, over several years and settings, and thus are the result of reflection and ponderation over a greater set of observations than previous research of programming with preschoolers (*vd.* section 4.2.3).

Usually, studies on the technical difficulties of people learning to program focus on "novice programmers", who are, typically, college students or sometimes high-school students (*e.g.*, Dijkstra, 1989; Holmboe *et al.*, 2001; Jenkins, 2002). Such studies consider, and rightly so, that children are not learning programming to become professional programmers. As a consequence, the analysis of the technical difficulties children find while programming is scarce, mostly based on the intuitions developed by several researchers (*e.g.*, Papert, 1980; Cypher, 1993; diSessa, 2000; McIver, 2000) and on multiple isolated accounts. In 1990, for instance, Mitchell Resnick pointed out several classes of bugs in concurrent programming by children: problem-decomposition bugs[298], synchronization bugs[299], and object-oriented bugs[300] (Resnick, 1990). Such isolated accounts often

[298] Difficulty in decomposing larger problems into subtasks.
[299] Misunderstanding of the implications, temporal or otherwise, of concurrently-executing subtasks.

present general programming difficulties of children mingled with specific environment-related issues. *E.g.*, in 1997, an analysis of the children's understanding of the programming system KidSim (a precursor of Stagecast Creator, *vd.* section 3.3.4, p. 151), mentions that children "*must figure out how to map the desired behaviors onto the KidSim language capabilities*" (Rader *et al.*, 1997) and also that they must "*consider how to distribute behaviors among the various pieces to achieve a global program operation that matches the science concept or phenomenon of interest*" (*id.*, *ibid.*), both of which are sensible global issues in programming. But that study then goes on to focus on four system-specific issues: interpretation of before-after diagrams, ordering of the rules provided by those diagrams, matching of pictures in the diagrams, and the interaction of objects represented in the diagrams (*id.*, *ibid.*).

Even regarding the above-mentioned most often researched "novice programmers", the existing knowledge is fragmentary and still requiring consolidation. For instance, the Psychology of Programming Interest Group currently considers that although "*certain issues regarding the psychology of programming have recurred in workshop presentations and papers over the years, they often appear to be addressed without continuity*" (PPIG, 2004). Nevertheless, some of those recurring issues may provide a helpful background for the interpretation of the hurdles presented in this chapter.

One particular notion is that sources of bugs can be divided into **construct-based problems**, *i.e.* misconceptions regarding the constructs in a programming-language, and **plan-composition problems**, consisting in difficulties to break down a program into elements and combining them towards the desired result. This last group has been shown to constitute the major source of bugs in novice programming by adults (Spohrer & Soloway, 1986).

Another notion is that the problems can be seen from the perspective of the human interaction with the skill of programming, rather than from its technical features. This duality was already somewhat present in the division presented in the previous paragraph, but has been more closely under scrutiny since. And in terms of human interaction, important factors can be divided into **cognitive factors**, such as the learning styles and learning motivation of each individual, and **interaction factors**, such as the facts that programming is an activity involving multiple skills and multiple processes; and that learning how to program requires the simultaneous devotion to learning novel skill areas and to precision-intensive learning, since the slightest error can lead to a complete failure of a program, among others (Jenkins, 2002).

Under this scenario, an attempt at consolidation of the theoretical background of programming difficulties is beyond the scope of this thesis. Therefore, I opted to present the hurdles simply by using an *ad hoc* taxonomy, its aim being merely to present the information to the reader in more convenient fashion. When applicable, the intercrossing factors mentioned in the two previous paragraphs are presented in association with each hurdle, along its description, in section 6.1.2.

Overall, children found it easy to personally engage with the ToonTalk environment. All but one of the children involved in this research enjoyed using ToonTalk, sometimes to the point of demanding their parents to purchase it or telling other children all about it in minute detail. And as can be seen from the details of sessions provided in Annex V, the child that didn't enjoy it was a very specific case regarding personal motivation: she considered ToonTalk not to be "serious" enough, *i.e.*, too playful. I believe it is a fair assumption that activities specifically geared towards this motivational background of that child would also have resulted in an enjoyable experience for her. This strong engagement of children with ToonTalk in general is contrasted with a more skeptical attitude of adults, particularly adults with a background in the preschool education field, who in general dismiss it as something that children won't enjoy.

[300] Confusion regarding the properties of specific objects and the relations between different objects.

Although beyond the scope of the current research, it would be interesting for psychological and sociological studies to be conducted in order to determine factors influencing both this strong engagement by children, and this common skepticism by preschool education professionals.

But it must also be said that, throughout the sessions, it would not be uncommon for a child to lose motivation or became disappointed, when specific circumstances or events would cause him/her to lose the notion of what could be done, or of what was happening. This is the first hurdle I chose to present, but rather than just consider it and other hurdles in isolation, it must be kept in mind that the hurdles can occur in sequence or coincidence, and thus are not just hurdles, but causal factors that mutually influence each other.

I chose to name these problems as "hurdles," because I see them as obstacles to one's personal progression and development, rather than a payload one has to carry while programming. And I believe that the now-common habit of labeling difficulties as "challenges" or "opportunities" would be justly applicable to them.

In hindsight, a major distinction between the interaction factors mentioned previously in this section and the presented hurdles is that the hurdles often lie at the level of general personal development, and are not simply technical programming issues or issues of learning difficulty. Perhaps such a condition should have been expected, given that preschool children are still in the first stages of their personal development. Thus, planning towards paths to help children overcome these difficulties can and should involve broader approaches to the development of children's cognition, social interaction, and self-attitude. In terms of programming activities, in the next section I have provided the context in which these general hurdles may appear.

With all these general ideas taken into considerations, as much as possible, I have formulated the hurdles in a language-independent way, so as to allow them to be more readily mapped and converted by researchers willing to conduct similar research with other programming environments, or wishing to explore programming from other perspectives. Such research avenues have already contributed and may further contribute to better establish the general underlying hurdles behind programming by preschool children.

1. **Children can easily get weary from unforeseen system behavior or difficulties.**
2. **Children may expect human-like interpretation of their intentions and autonomy in its execution by the computer.**
3. **It can happen that during programming (seen as "teaching", "showing", or "explaining") children combine actual events and actions with fantasies.**
4. **The concept of sequencing activities can be confusing.**
5. **Children find it hard to consider the possibility of being wrong and not knowing it.**

6.1.2. Descriptions and strategies

Children can easily get weary from unforeseen system behavior or difficulties

Spohrer & Soloway (1986) classification:	construct-based & plan-composition
Jenkins (2002) classification:	**Cognitive:** motivation **Interaction:** interest & pace

The immediate consequence of this is that children lose interest or dismiss potentially interesting paths of exploration. They may return to strategies and activities they feel comfortable with, until they no longer satisfy them, at which time they may give up on programming entirely.

Such circumstances can be originated by multiple causes, from the complex to the trivial. For instance, when a ToonTalk picture is unintentionally dropped on another, they are bammed together and the child is momentarily abducted from her intended path of action. She may be able to vacuum the dropped-on picture and resume. Or she may not. Or find the situation awkward and abandon it. Should the child come across system bugs and errors, they may also contribute to this, as may disruptive actions by other children.

One must be aware that these difficulties are often located mainly at the physical level (*i.e.*, fine motricity control), not at the cognitive level. It was not uncommon for children seemingly unable to execute complex procedures in ToonTalk to be able to "coach" or even "teach" other children, more skilled on the use of the mouse, to perform beyond what themselves had managed to accomplish.

Another, more subtle version of this problem occurs when a child find himself/herself in an uncomfortable cognitive situation. For instance, it's fairly easy for something to distract children from general goals. For instance, when a child interrupts a programming activity to perform other required tasks elsewhere, such as drawing in Windows Paint, that same child may end up forgetting what was the purpose of the drawing after returning to ToonTalk; or he/she may remember the purpose of the drawing, but not the current state of affairs in the context of the activity being developed.

It was fairly normal for children, in such circumstances, to change course and objectives entirely. The most likely outcome is that further detours from their plans occur, and eventually the children will find themselves amongst a mess of unfinished ideas and elements.

Teachers and educators must therefore pay particular attention to the setting and development of goals by/for children. But one must be aware that it can happen for a child to abandon a goal simply because he/she came across a new focus of interest, not because he/she lost track of previous ideas and/or status. This, in itself, is not an occurrence of this hurdle.

An approach that helps in this regard is the contextual integration of programming activities within the general context of activities taking place in the preschool room (Morgado *et al.*, 2005), which provides mutually-reinforcing contextual support (*vd.* sections 6.2 and 7.1).

Children may expect human-like interpretation of their intentions and autonomy in its execution by the computer

Spohrer & Soloway (1986) classification:	construct-based & plan-composition
Jenkins (2002) classification:	**Cognitive:** learning styles **Interaction:** educational novelty

Hastily, this problem might be classified as a matter of emotional or personal immaturity of preschool children. But this is not so: it is a fundamental problem in computer programming, which has previously been found also on adolescent and adult novice programmers:

> *« A major finding is that the students attribute to the computer the reasoning power of an average person. »*
>
> *(Sleeman et al., 1986)*
>
> *« Pea (1986) observed that a substantial proportion of college students assumed that there is a hidden mind somewhere in computer programs. »*
>
> *(Kaasbøll, 1999)*

This problem reveals itself in various forms: the most obvious one would be the appearance of semantics problems, due to lack of consideration by children to the possibility of different interpretations of their actions, as picking up the contents of the fourth hole in a four-hole box as discussed in section 3.4.3: is it the fourth hole, or the last hole?

But there are other, more subtle instances of this, like when children expect a ToonTalk robot to "get to work" immediately after being taught, without giving it a box to work on. One must recall that in ToonTalk children are teaching robots with the actual items they will be working with, and so it might be assumed that the "natural" behavior for robots would be to pick up the box used in its training and start working.

Another instance is when children teach a robot and don't expect any further results; rather, they had simply been enjoying the interaction provided by the interactive programming process, during which they had been in a powerful role, which can be described as "teacher" or "leader". They were showing their intentions to a captive audience (the robot).

Yet another example occurs when children see a robot repeating the redundant mistakes they made while programming it (like dropping a box on the wrong place and picking it up again). Typically, children will be puzzled by this. Not only would they expect a robot to dismiss these redundant actions, but sometimes they would even forget about having performed such deviations from the intended path at all.

To tackle this problem, the teacher or educator needs to be observant regarding all events taking place during the programming activities, avoiding the temptation to dismiss attempts of humanizing the computer as ordinary. The overall idea behind this keen procedure is to detect specific instances of this hurdle, and react according to the specific instances and circumstances.

Below, I provide a sample of specific instances of this hurdle, and suggestions on how to approach it.

The programming itself may easily be perceived by the child as the whole point of the activity, rather than a path towards an objective

This is a common situation in which the child shows a ToonTalk robot how to do a certain set of actions, but then is perfectly content with that, not bothering to actually use the robot that he/she programmed. It is as if the child was playing the role of teacher, leader, football coach, priest, ruler, celebrity, or simply enjoying the powerful role of showing someone else how to do something – that someone else being the ToonTalk robot (a captive audience, as I mentioned above). During the research sessions, children did enjoy programming robots and some would indeed "try it out" and see if the robots worked. But it often this problem would surface.

It was found that the children would keep focused more easily, avoiding this problem, by having previously established with them a goal that would only be achievable by using a robot (obviously, a goal to their liking or choosing). For instance, playing many sounds at once, or building houses fast enough to fill up the city before they had to abandon the computer.

The initiation of the programming process may not be understood as an example or starting situation, but as an "interrupt" or "switch", like turning on a toy

In other words, the child may not perceive the start of the programming process (in ToonTalk, giving a box to a robot) as such, but rather as a mandatory process that he/she needs to complete to move on towards what he/she intended.

The consequence of this is that by acting in this manner, it is common for children to lack the association between preset conditions and intended actions. But this is an essential element in programming, regardless of whether a programming language is transformational or reactive (*vd.* section 3.2.1, p. 111): to build a program, for each subpart the programmer must associate a cause with its intended consequence. It doesn't matter for the present discussion whether the cause is a request, an event, a function call, or the asking of constraints, nor whether the consequence is the execution of code, a simple acknowledgment, an operation over arguments, or the telling of constraints. Other ways of putting this are: lack of association between an activity, its pre-conditions and its post-conditions; lack of consideration for "before" and "after" state; or even, in more traditional technical jargon, disassociation between input and output.

In ToonTalk terms, this means that the robot's box will not be seen as a container for acting upon (or at least, it won't be absolutely clear). During the research sessions, quite often a child would start teaching a robot by giving it an empty box (thus, using it like an "on" switch for the robot), and then "teach" with objects taken from the tool box or notebook, but involving contextual information that at execution time would only be available if provided explicitly.

For instance, a child teaching a robot how to put a picture on the wall might teach it by making it access the pictures notebook or the main notebook, and using a picture from there. Upon running it, the same picture will be put on the wall and the child is left with no way to change the robot's behavior[301]. Furthermore, it would have the consequence of running without end.

A useful technique, in this regard, is to lead children into reasoning over what is necessary to achieve a task, before initiating the programming of that task. This technique as been useful even with adult novice programmers, and is detailed in section 6.2.

Most children find it hard to grasp the "end of program" concept

This problem is visible when a child completes the actions required by a specific program or subprogram and continues to use the programming environment, unaware that his/her actions are still being incorporated into the program.

In ToonTalk, this occurs when, after teaching a robot all its necessary actions, children proceed by playing around with the toolbox objects, regardless of the fact that they are moving them with the robot's hands, not the programmer's, and that there is still a cloud background visible, instead of the floor backgroud. One can imagine equivalent problems in other programming environments, but I am not aware of detailed research regarding this situation. For instance, in a traditional text environment such as Logo (p. 139), a child might forget to close a function or procedure; in a visual environment such as Stagecast Creator (p. 151), a child might decide to keep on using the programming elements without closing the rule-creation dialogue.

To overcome this problem, during the research sessions adult intervention at the right moment (when the programming activities had just been completed) was found to be crucial to make children realize the importance of "telling the robot that the lesson is over."

[301] In this particular example, it could indeed be changed by editing the notebooks; but the conscious use of that technique assumes an ability to employ a plan-composition skill more complex than the one being addressed.

Children expect subprograms to run without being set up

In other words, children often create a small subprogram, but expect the computer or the subprogram itself to "know" when to run and on which items.

To put it in ToonTalk terms, children expect robots to start working the moment they are taught, by using the same box that was used to teach them. Children would forget or find unacceptable that they had to give the robots a starting box.

However, an important factor to consider is that this may be a consequence of programming based on small sets of actions, which have an immediate and "obvious" application. During the research sessions, children wouldn't make this kind of assumption for robots intended to be used in a specific situation, such as, "when it reaches the new house", or "when the picture is turned around".

Therefore, I believe a useful approach to tackle this problem is to introduce in the planning phase, before the actual programming actions, elements that cannot be included in the program itself, such as a schedule, a location, or an external event. Or, in more formal terms: to plan and develop a context that imposes such conditions on the execution of an activity.

It can happen that during programming children combine actual events and actions with fantasies

Spohrer & Soloway (1986) classification:	construct-based
Jenkins (2002) classification:	**Cognitive:** learning styles **Interaction:** educational novelty

While this situation may seem to be easy to detect, that is not necessarily so. A child may create a program that seemingly does nothing, or that appears to be composed of random commands or actions, and such situations provide a clue to the occurrence of this hurdle. However, it can be masked in minute details and expectations, undetectable unless one probes the child's understanding and intentions.

For instance, in ToonTalk a child may teach a robot how to pick a box, then drop it, pick up a picture, move around, put the picture back, put something else in the box, put objects on the ground and combine some pictures, numbers and text with Bammer. And then, if that child is asked to describe the robot's actions, the description can be revealed as an imagined story or a mixture of the actual actions and fantastic interpretations of them; or even include imagined, invisible events that are supposed to be occurring between actions. In the particular case of ToonTalk, since programming is an interactive activity between the child and a robot-persona, this may be seen as being analogous to doll-playing or figurine-playing with the robots and other programming elements, instead of programming.

Regarding the detection of more subtle occurrences of this hurdle, it may be wise to inquire from time to time, asking children to describe the behavior or purpose of a specific subprogram.

As for dealing with it, the research sessions provide little help, for they had not been planned in a way as to facilitate the detection of this. And the mere absence of this hurdle cannot serve as an indication that it was avoided: it might as well have passed unnoticed. However, there were specific circumstances in which children did describe the actual events taking place, and those may be seen as indications of potential paths for tackling this; and at least one case where a child maintained her interpretations in the fictional realm, which can provide an indication of what may not work. Also, one must recall that programs are constructions with an actual existence, and thus must pass what Seymour Papert called the "*test of reality*" (Papert, 1999, p. xii): they're only complete and successful if they work; "*if they don't work they are a challenge to understand why and to overcome the obstacles. They can be shown, shared and discussed with other people*" (*id.*, *ibid.*).

Among the specific circumstances that occurred in the research settings, particularly valuable were situations in which a child would describe the behavior of another child's robot, as long as it was simple enough to allow this analysis, and would flatly said that it wasn't doing the necessary actions. Asking other children to provide advice was also helpful. Finally, also valuable was the lack of success of the attempt to make a child realize the situation by asking her to describe the robot's actions to the other children, using a comic-strip version of the robot: the child stuck with a fictional interpretation of the depicted actions, and although the other children soon ignored her strange story, she nevertheless attempted to tell them the "story of her robot", never recognizing that there was no actual story to tell from what was there.

The concept of sequencing activities can be confusing

Spohrer & Soloway (1986) classification:	plan-composition
Jenkins (2002) classification:	**Cognitive:** learning styles & motivation **Interaction:** multiple processes & pace

Programming, even in an animated virtual world, often requires planning in advance, or at least being able to mentally keep track of series of events and consequences. But young children can easily get mixed-up in both situations. When planning in advance, they may forget parts of the plan, or lose the overall notion of what is being planned. When reacting to ongoing events and results, it is also frequent for them to miss the link between an event and its consequences, or vice-versa. The complexity level that each child achieves depends on himself/herself, but also on many circumstances, such as context, mood, and familiarity to the subject.

This is something that must be carefully considered when planning programming activities for children: requiring them to follow a strict plan or series of events may be counter-productive. They may be unable to devise a plan to its conclusion, hindering their motivation to develop it. And they may find themselves subject to various events which they haven't had the opportunity to interpret, which in turn places them in the first hurdle that was described in this section.

Understandably, in the research sessions, children have performed better when the required planning and events involved interactions they were already familiar with, and where just a few elements required careful thought and concentration at a time, rather than having at each step a series of novel events requiring them to regain their balance and bearings. Also, seeing as this particular hurdle is close to Papert's concept of procedural knowledge (*vd.* section 4.2.2), and therefore closely associated to the development of new thinking strategies, one should make sure the children realize their efforts will eventually work.

An important approach to this hurdle is therefore the careful consideration of an overall context or metaphor which children can use to map their knowledge and worldviews, in each specific activity. This can be done as mentioned for the first hurdle, by contextual integration of programming in the context of other preschool room activities (Morgado *et al.*, 2005).

Children find it hard to consider the possibility of being wrong and not knowing it

Spohrer & Soloway (1986) classification:	plan-composition
Jenkins (2002) classification:	**Cognitive:** learning styles **Interaction:** educational novelty

It is common knowledge among programmers that hardly any program can be found to be free of errors. A fundamental skill in the process of programming is precisely the ability to review a program's execution, structure and overall operation, and correct any errors that are detected – the process commonly referred to as "debugging."

> *« Debugging is a central part of computer programming. Programs seldom work when first coded. Many superficial syntactic errors are easy to find* (...)*. But a substantial number of bugs require concentrated problem solving. Though various debugging strategies and tricks are often taught in programming classes, much of the skill of debugging is learned through the experience of writing programs and getting them to run. »*
>
> *(Gugerty & Olson, 1986)*

Most studies of how children – even preschool children – and novices in general learn debugging assume that the correction of errors is a natural necessity, and that the main problem is the acquisition of better overall understanding of the operation of programs:

> *«* (...) *young children debugging their Electronic Block stacks were limited by a lack of understanding about the nature of interactions between blocks. Older children successfully debugged code stacks, as their obviously deeper understanding of the blocks allowed them to alter structures to produce the behaviours they desired. »*
>
> *(Wyeth & Purchase, 2002)*
>
> *«* (...) *the primary reason for the experts' superiority was the ease with which they understood what the program does and is supposed to do.* (...) *Novices* (...) *did not have sufficient programming knowledge to focus in on good hypotheses. »*
>
> *(Gugerty & Olson, 1986)*

However, in the course of the research sessions with preschoolers, I became convinced that there is a more fundamental issue to address, regarding young children. And that is the very notion that it is possible to explain something in the wrong way, but without knowing that it is wrong.

During the sessions, while programming ToonTalk robots, most children assumed that a robot learned exactly what they meant to teach him. They would not consider either the possibility of having made a mistake or that their actions may very well have unintended consequences. This was gathered from the fact that often children would not welcome suggestions for "testing" or "trying out" their robots. I sometimes urged them to test the robots nonetheless, before actually using them in the intended situation. But in general, children would do so without much enthusiasm, and the mere suggestion of "testing" usually left children puzzled about the purpose of such testing.

I have not been able to overcome this problem: when robots had an immediate application, "testing" would occur naturally, since testing and execution would be identical. I can only point out to research literature on older novice programmers, which recommends that learners of programming "*enhance their debugging skills by completing carefully planned debugging activities*" (Chmiel & Loui, 2004). I therefore propose taking advantage of any opportunities where preschoolers programs can behave unexpectedly, and help them confront this situation.

6.2. Lessons from Working with Kindergarten Teachers

> *« The kind of intensive monitoring of a programming activity, are quite demanding for a facilitator. Firstly, it requires detailed skills of the programming environment, which one cannot always expect a teacher to have. Secondly, and perhaps more important, it requires an active participation of the construction process of the children. The facilitators have to be co-constructors together with the children. It would be difficult to step in and give children adequate support. This would probably be difficult to manage in school situations. »*
>
> *(Tholander, draft, 2002)*

The above opening quote presents an overview of the complexity of the task standing before the computer-activities teachers that took part in the research sessions of the fourth research setting. However, would this still apply to activities conducted with preschoolers? After all, most of the activities that I had previously developed with preschoolers, in two initial research settings, were extremely simple. So perhaps those "*detailed skills of the programming environment*" as Tholander calls them, above, would no longer be necessary; and the common educational approach of preschool teachers already involves active participation in the construction process of children, which could further contribute to simplify the use of computer programming.

As the sessions[302] under the fourth research setting revealed, and section 5.3.5 summarized, these assumptions were not correct. In terms of programming skills, the teachers came across some of the problems that the children were also facing; and regarding the active participation of teachers, quite often they ceased to act as such: they were in an uncomfortable situation, and most fell back on instructing children on the few samples I had provided while introducing them to ToonTalk. Finally, possibly also due to the same situation of discomfort, teachers started over-relying on personal assumptions on the performance of children, often failing to conduct keen observation of the children, as they would normally do. This much is visible from their reports, where often contradictory statements are made in the same sentence, regarding observations and evaluations, as pointed out in the sessions overview presented in Annex VII.

It is all the more telling that some care was taken to avoid simply sending off the teachers, totally unprepared, to conduct novel activities, a common equivocal situation:

> *« Too frequently, new and complex expectations are downloaded onto classroom teachers without a realistic consideration of the resources available to teachers to achieve these expectations. Often, there is an assumption that technology itself is the panacea, and so, little consideration is given to preparing teachers to use the technology effectively and in support of their own teaching and learning goals. »*
>
> *(CSTA, 2003, p. 11, footnote no. 2)*

Rather, due attention as given to the launch and development of ToonTalk programming activities by the teachers. Although positive results were scarce, that was not the starting expectation; given the positive indications of children abilities, in the previous research settings, my estimation was that there was a nice potential for development of interesting activities, since the sessions were to occur throughout the school year, and be conducted by people accustomed with performing computer activities in preschool settings.

[302] Annex I provides an overview of these sessions, and Annex I contains all the reports provided by the computer-activities teachers. Another contribution to my personal construction of these concepts was my experience as lecturer of the course "Informática no Ensino" ("Computers in Education"), part of the curriculum for the bacallaureate in Early Childhood Education at the University of Trás-os-Montes and Alto Douro (since 2001).

The soundness of the original approach used with the computer-activities teachers can be evaluated by comparing it with recommended key conditions and approaches to help teachers grow and change in educational settings:

> *« 1) Teachers need regular time together* (...) *2)* (...) *it's important that teacher change efforts be collaborative, social experiences* (...) *3)* (...) *significant change needs leadership from the top, even while the support itself emphasizes teachers' decision-making and initiative* (...) *4)* (...) *we need to support and strengthen teachers' latent professionalism* (...) *5)* (...) *the introduction of new classroom strategies is most effectively initiatied by inservice programs using concrete experiential activities, rather than starting with educational philosophy or research data.* (...) *6) After experiencing new classroom strategies, teachers need to reflect, to analyse, to compare – to build knowledge and theoretical understanding. »*
>
> *(Zemelman et al., 1995)*

The teachers conducting activities in setting 4 (section 5.3.5, p. 335) were computer-activities teachers for preschools (profile: computer training + education degree or specific training). Each provided computer-activities support at 4 or 5 preschool rooms per week and worked directly with about 100 children, on average (each teacher used ToonTalk only in one of its assigned preschools). Furthermore, they provided computer training for regular teachers based at those preschool rooms.

One can see how the various conditions and approaches enumerated in the above citation were met in the preparation and development of the fourth research setting:

1. There was a weekly meeting between all teachers, to discuss the week developments, lay foundations for the coming weeks, and plan joint actions.
2. The computer-activities teachers were not working alone in their allocated preschools: at each, the main teacher was typically present and the project hiring the activities-teachers had specified as a directive that these main teachers should gradually become autonomous in the use of computer resources in their educational practice (Cruz *et al.*, in the press).
3. The participation of the teachers in the ToonTalk sessions was originally proposed by their project manager, but the teachers were given freedom to adapt and determine the course of their practice, and encouraged to adapt the ToonTalk sessions as they would in their usual computer-activities practice.
4. From the start, although I had some ideas that I wished to try out, I left clear to the teachers that they would be entirely responsible for the educational orientation and development of the sessions. I would not impose personal educational perspectives, and trusted each of them to provide the educational know-how bestowed by their training and experience.
5. Rather than provide educational philosophy[303] or hard research data, the training sessions conducted prior to the development of activities were focused on allowing teachers to explore the ToonTalk environment and try out specific sample activities[304], which had previously been done by children.
6. Each teacher conducted only a single ToonTalk session per week; this meant that the teachers would reach weekly meeting (where progress was reported, issues discussed,

[303] As sessions took place, I would sometimes during the weekly meetings leverage teachers' accounts as context to broaden their understanding, including some theoretical background.

[304] These were based on children's achievements during the first and second research settings. For example, decorating houses, building houses, organizing objects on the floor, composing pictures, playing hide-and-seek and exchanging gifts with birds, and programming robots to perform these various activities, as well as others such as swapping the contents of boxes or passing a ball around using birds.

and further actions debated) with an interval of a few days between it and the previous and coming sessions.

And these short descriptions don't cover the entire spectrum of support provided. Specific documents were laid down for guidance, setting forth the data recording style and elements, explaining the approach styles mentioned in section 5.3.5, emphasizing the importance of maintaining a given approach, providing examples of activities for each approach style, explaining the issues put forth in section 6.1, establishing terminology, and providing some ready-made logical explanations for children's anticipated questions. (Example: "Why must we stop teaching the robot by pressing ESC?" – "Because it's flying, while in the Robot School in the Clouds. If we don't tell it to come down, by pressing ESC, all the stuff will fall and on the ground and get broken.") These documents are presented in Annex II, on p. 445.

Yet, as I already mentioned in this section, encouraging results were scarce. And this constituted the most important outcome of the fourth research setting: the awareness that it was not that the general conditions had not been met (although indeed with plenty of room for improvement), or that the general approach to teachers had been laid on wrong educational principles. Rather, as mentioned at the opening of this section, the problem laid on wrong starting assumptions on the particular nature and features of computer-programming activities with preschool children and preschool teachers.

Quickly I found out that computer teachers involved in programming were facing some of the hurdles I had previously found with children. For instance, the very first time they taught a robot – they did it with an empty box! In other cases, they would be happy to see the children "teaching the robot", without caring about **what** was being taught to the robot, or **with what purpose**. (In both cases, instances of the second hurdle mentioned in section 6.1.)

Therefore, a specific strategy was proposed for presenting robot programming (Morgado *et al.*, 2003a), following the general sequence of activities that compose the classical definition of programming (Blackwell, 2002): **defining requirements**; **perform a specification**; "***design the technical features****"*; **perform the coding**; and "*anticipate and account for departures from the intended behaviour (**debugging**)*" (*id.*, *ibid.*):

Step		Challenge question	Intended consequence	
1st		What do you want to do?	Setting a purpose. (*defining requirements*).	
2nd		What do you need?	Define needed objects and assemble them in a box. (*perform a specification*).	
3rd		How do you do it?	Execution of actions upon the objects. (*design the technical features*).	
4th		Can you teach a robot that?	Demonstrate to robot a specific set of actions. (*perform the coding*).	
5th	(a)	Did it learn properly?	Emphasize the need to test programmed actions.	*Debugging*.
	(b)	What went wrong?	If the robot wasn't properly programmed, or if it doesn't connect to other objects as expected, discuss the reasons and possible solutions.	

Table 34 – Strategy devised for teaching robot programming

The teachers found this set of question helped them see programming in a clearer way, but by the time I developed it, they didn't manage to try and evaluate its impact and usefulness with children.

Another problem the teachers faced was that they found ToonTalk and programming to be so different from what they were used to do, that their activities would end up closed within ToonTalk and its environment, disconnected from everyday preschool activities, to the point of sometimes there being no sense of purpose for the activities, beyond ToonTalk itself. In one case, for instance, a teacher was conducting activities with birds and nests, having been told that she could focus,

computer-wise, on different combinations of birds and nests: 1 bird to 1 nest, several birds to 1 nest, 1 bird to several nests, and several birds to several nests. This she did, but only by discussing with children expected behaviour for these situations and trying them together with the children, to see what would happen. And what had been the ongoing project at the preschool, that week? Ethnic groups. Simply by providing as a suggestion, in the meeting of the following week, that different houses could have been customized to represent different countries/continents, that birds could have be used to send letters to children living in those countries, and that this arrangement could be have used for a "1-to-many", "many-to-1" activity made her feel much more comfortable and able to provide further suggestions on her own.

One late result of the weekly meetings, was the conjoint realization that teachers would welcome some quick reference cards on computer-based educational topics, in order to help them build their activities around those topics. *I.e.*, one for birds, which could include the example I have just mentioned; another for robot-programming, presenting the strategy of Table 34; another for working with ToonTalk sensors, mentioning the need to focus on preparing different consequences to the same actions, so that children can be encouraged to analyse the impact of the sensor; and so on.

The consideration of these particular observations eventually led me to create the "cookbook" of computer-programming topics, which was presented in section 3.4, and to focus on reasoning over the research results and their presentation so as to help preschool teachers integrate the computer-programming activites in their educational practice and contexts (*vd.* chapter 1).

The two latest research settings (described in sections 5.3.7 and 5.3.8) were already conducted under the influence of these considerations, and provided positive indications regarding their feasibility and effectiveness. For this reason, the guidelines presented in chapter 1 represent my reflections resulting from these two sets of data: the issues described here (and in the previous section, regarding the children themselves), and the results of my initial attempts to overcome them, in settings[305] 6 and 7.

[305] Besides the brief descriptions in sections 5.3.7 and 5.3.8, further information is provided in Annex I (final report on the activities conducted in setting 6) and in Annex I (full logs detailing the activities conducted in setting 7).

7. Framework for computer programming in preschool

7.1. Guidelines for computer programming in preschool

In chapter 1, I presented the research conclusions under two categories – children and their teachers. It would perhaps be expectable to provide guidelines similarly divided, in two matching categories. So why haven't I done that? Why did I combine the guidelines into a single integrated section?

Basically, my decision came as a direct consequence of reasoning over the first of the hurdles presented in section 6.1.1: "Children can easily get weary from unforeseen system behavior or difficulties." I have no qualms about letting children explore a programming environment by themselves. And indeed for some children involved in the research settings this turned out just fine: they learned by themselves and from each other, and in the process developed a strong personal relationship with the products of their exploration. But for most children, this was not the case: some couldn't manage to control the system, and discarded it; others would embrace it, but just repeating over and over similar patterns. And even those children that broadly explored it on their own had huge gaps in their understanding; gaps preventing them from reaching more complex patterns of use of programming – or even from doing programming altogether. Case in point, the four-year old AX, from group 10 of the sessions conducted at the SPP preschool during the fourth research setting.

AX's brother had learned ToonTalk programming during the sessions I conducted in the second research setting, and he had then shown AX how to play. However, one year later, when AX was taking part in ToonTalk sessions conducted by a computer-activities teacher, his programming skills suffered from contradictions and lack of design[306]: "*As sessions progressed, the teacher noticed that although AX understood the need to compose an entry box for the robot, he had the tendency to give it an empty box, since he got used to the fact that this would take him into the robot's thoughts. The teacher also reported some confusion between his actions and the robot's actions, when playing in the thought bubble, but his answers to queries show otherwise. But yet another teacher comment is that AX forgets to exit the robot's thoughts, and this may provide a clue to solving these contradictions.*"

It is therefore the first precept of these guidelines that **preschool teachers must be deeply involved in the programming activities conducted by children**, lest most children fail to climb the various cognitive steps involved in the craft of programming and forfeit any benefits. This much is a common recommendation in existing research literature, as the following citation sums up:

> *« The third and present phase* [in the teaching and learning of programming] *suggests more direct instruction and mediation by a teacher rather than the discovery methods of instruction. Littlefield, Delclos, Lever, Clayton, Bransford, and Franks (1990) reported that "mastery of the programming language has not been achieved when LOGO has been taught in a discovery-oriented environment" and "if mastery is not achieved, the discovery and transfer of general thinking skills as a result of learning LOGO cannot occur"* (...). *More researchers (Lehrer, Guckenberg, & Sancilio, 1988; Clements, 1990) are noting that if the techniques of programming are to be learned, they must be taught and if they are taught in a mediated way, then transfer to problem-solving skills can occur. »*
>
> *(Kushan, 1994, p. 249)*

But teachers can be of little help beyond simple manipulative assistance, if they are themselves groping to achieve some understanding of the whole process. As was mentioned in

[306] This quote is part of the overview of these sessions, provided in Annex VII, and the full reports are provided in Annex VIII.

2004, in recommendations from the U.S. Department of Education for states, districts and individual schools, on the effective use of technology to enhance learning:

> *« Ensure that every teacher knows how to use data to personalize instruction. This is marked by the ability to interpret data to understand student progress and challenges, drive daily decisions and design instructional interventions to customize instruction for every student's unique needs. »*
>
> *(USDE, 2004, p. 41)*

This recommendation can be combined with the information provided in section 6.2, regarding the difficulties found in teacher intervention in a computer-programming environment. The result of that combination would be a recommendation along these paraphrased lines: **each teacher should know how to use computer programming well enough to be able to personalize instruction**; this is marked by the ability to interpret computer programming to understand children's progress and challenges, drive daily decisions and design instructional interventions to customize instruction for every child's unique needs.

This thesis has already provided some contributions towards these ends. Section 6.1 has provided several insights from practice on particular hurdles that preschool children face when embracing an animated, virtual programming environment – helping understand children's challenges. Section 3.4 has provided a cookbook approach to computer science topics – helping teachers better understand children's progress.

It might sound like a reasonable expectation that the enhanced understanding brought forth by both of these contributions, by itself, will help teachers "drive daily decisions" and "design instructional interventions to customize instruction for every child's unique needs." But one of the realizations provided by the research settings with preschool teachers is how hard it is for them to achieve these ends, just by relying on information on computer programming; how often samples and examples end up being used for their own sake, the teachers insufficiently comfortable with computer programming to use them as inspiration for these higher-level ends.

The present chapter thus presents this lacking contribution to the use of computer programming in preschool settings. It focuses on providing teachers with a design structure. In other words, it provides a framework to help teachers plan activities, customizing them to suit the context of a particular child, preschool room, activity, or educational theme/project.

In this framework, the key idea is **integration**. It relies on the assumption that teachers have a latent professionalism that can be tapped, if it is adequately supported and strengthened, as mentioned in section 6.2.

This assumption is supported by the developments registered in research settings 4, 6, and 7 (*vd.* sections 5.3.5, 5.3.7, and 5.3.8). During the weekly meetings of research setting 4, most teachers didn't realize the potential educational value of computer programming, even after undertaking training sessions, even after seeing first-hand how most children relished their activities in ToonTalk. They only managed to realize this potential when, during the meetings, I provided examples of how the activities of the previous week might have been integrated in the overall events and goals of their specific preschool and children.

And the two teachers that did conduct ToonTalk sessions until the end of the school year were those that had managed to integrate their ToonTalk activities in this manner, at least partly. And I say "partly", because these instances of integration were seriously lacking: as described in section 5.3.5 and in Annex VII, in the Mondrões preschool a "city of professions" was created, customizing the environment, but no programs or programming constructs were used; in the São Pedro Parque preschool, children's drawings and images from other events in the preschool were used, but beyond

that, the activities were those provided as samples, such as teaching a robot to re-order the contents of a box (only now the box had pictures from a tale, for instance).

In settings 6 and 7, however, the trainees from the early childhood baccallaureate and the adult facilitator managed to achieve a much stronger involvement of chldren, and provide richer educational situations, after receiving my support, in the form of tentative versions of the ideas described in this framework.

This key idea of integration is also supported by existing research, which recommends that the computer is integrated in other activities, with broader scope, rather than simply used to duplicate current educational methods or worse, be used just as a reward or an electronic baby-sitter (Davis & Shade, 1994; Clements & Sarama, 2002; Tsantis, Bewick & Thouvenelle, 2003).

> *«* (...) *computer activities yield the best results when coupled with suitable off-computer activities. For example, children who were exposed to developmental software alone showed gains in intelligence, nonverbal skills, long-term memory, and manual dexterity. Those who* ***also*** *worked with off-computer activities gained in all these areas and improved their scores in verbal, problem-solving, and conceptual skills. »*
>
> *(Clements & Sarama, 2002, p. 342, my emphasis)*

To integrate computer-programming, my overall guideline is to **use the programming environment and programming constructs, making them crucial to non-computer activities, rather than being a simple add-on**. To clarify this idea, in the following sections I present it under two separate concepts.

Firstly, in section 7.2, my contribution is a four-step guide to the general integration of computer use in the daily practice of preschools. Before integration of programming is achieved, the teachers must be able to integrate plainer computer use in their daily practice. The last step in this guide is an explanation of how computer programming may be employed in the previous three steps.

Secondly, in section 7.3, I typify the activities that can be performed in a virtual, animated computer-programming environment, each type providing a progressively more complex exploitation of features.

7.2. Integrating the computer in off-computer activities

7.2.1. Overview

As mentioned in section 7.1, here I present a four-step evolution, progressing in the amount of integration of computer use in educational activities. These steps are presented resorting to small examples that illustrate them, and with each example, comments specify the actual integration and contextualization idea it aims to cover.

However, these steps are not intended as successive steps for the development of activities. Rather, they are simply steps in this presentation of ideas to the reader. I have not researched potential impact in following/not following these steps when planning the development of activities.

The four steps described in the following subsections are, regarding the use of the computer in preschool settings:

1. **As a destination:** it is the final setting of the activity, which originated outside the computer.
2. **As a source:** it is a source of pieces to be used in an activity that takes place off the computer.
3. **Mingled:** it is both a starting point and an end, but neither the sole starting point nor the sole end.
4. **Automated:** a place where programming is developed to automate or accomplish a task developed with one of the three previous steps in mind.

To present these steps, I'll use examples from an important set of typical preschool activities: that of world-knowledge activities (Ministério da Educação, 1997, pp. 79-85). Their aim is to draw the child's attention to particular features of the world, such as the names employed to identify colours, the distinction between geometrical shapes, the composing parts of an animal, the features of each season, the origin of each kind of food, etc.

Besides the particular knowledge content of each activity, these activities' strong ties with real-world phenomena render them particularly rich in opportunities for learning. For instance, knowing where milk comes from obviously involves finding out what a cow is, how it is milked, and so on; but it can also involve notions such as the size of a cow and comparing it to the preschool room, and estimating how many cows would fit there; it can involve the exploration of how the milk goes from the cow to the supermarket shelves, and involve social elements such as transactions, buying, selling, advertising; it can involve playing a child-theatre piece or a playground game about herding cows, which will include significant physical activity; and it can include making cow puppets or drawings, with associated skills of representation and fine control.

The use of preschool educational activities in this manner is often within the scope of design projects. In such projects, students design and create artefacts (either alone or in a group), which can then be shared and discussed with others, or enjoyed as a final piece. Such artefacts can be physical items such as a flower garden, or a doll-house (or even a doll-igloo). But they can also be immaterial items, such as a story, a theatre play, or a virtual computer game or environment. The educational value of such design projects has been achieving growing recognition (*e.g.*, Harel, 1991; Papert, 1993; Kafai, 1995), and is fundamentally connected to the constructionist principles presented in section 4.2.2.

7.2.2. Computer as a destination – "I drew that tail"!

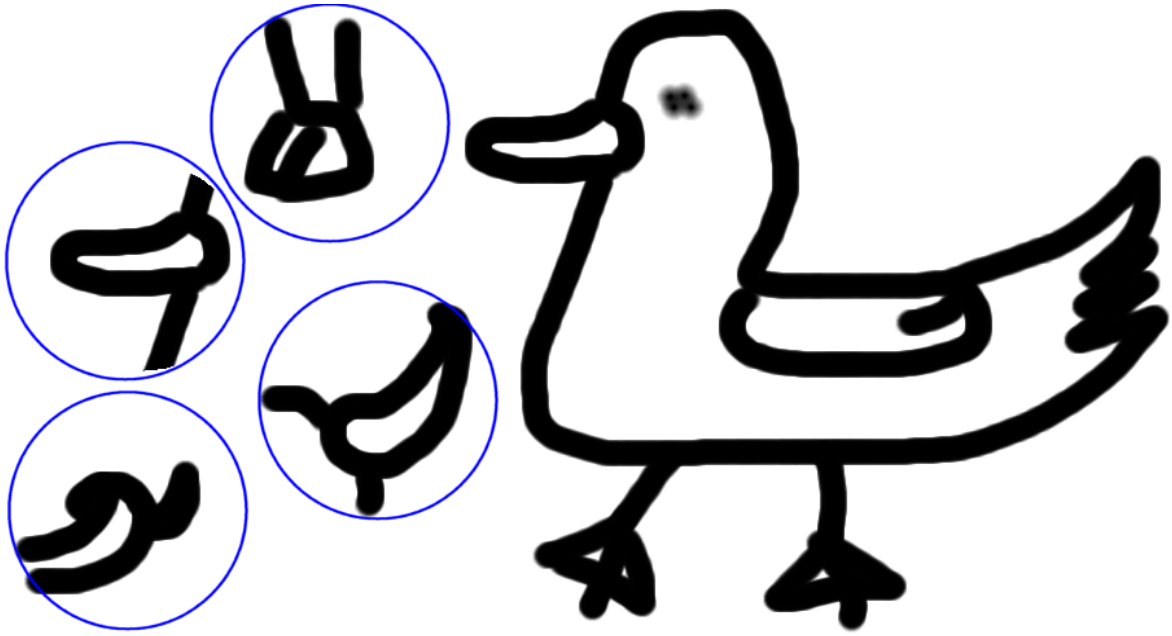


Figure 250 – Matching animal's parts to an animal

Figure 250 shows a simple image-matching game. The four circled images on the left are animal parts (top to bottom: hoof, beak, horn, pigtail). One of them belongs to the bird on the right.

Typically, this activity might be developed in preschool when the overall context of activities is one of "animals are made of parts" – hopefully, in this way the child player would be more prone to focus on the concept.

But rather than being mere players, children can be involved in the complete design of such as activity.

In the above figure, its composing drawings are amateurish, and this aims to represent an important feature: the bird and other animals are meant to be drawn[307] by children. These images could then be used by the preschool room teacher to build the image-matching activity, using any program usable for building interactive activities, such as Microsoft's PowerPoint or Animated Programs' ToonTalk. The teacher can also cut pieces from each animal picture and use those pieces and the original pictures to develop the matching game. With this simple feature, the children when drawing are not just ending their activity there: they are contributing to a game to be developed by the teacher, which they will then play afterwards.

But besides the images themselves, other features of such a game cannot be presented in an image such as Figure 250. There is no need to confine children's participation to the drawing of pictures. They could (and I personally believe they should) be involved in the entire design process.

In this example, children's involvement in design could be achieved through some or by all of these actions:

- defining the game's rules – what is right, what is wrong;
- define the game structure - what size is the animal, how many pictures to match, what sizes, what shapes, locations of pictures, etc.;
- define the content organization – any animal goes? or only birds? can extict animals be included? what about plants?

[307] They may also be phtographs of real animals, taken or selected by children; or photographs of physical clay pieces or other products of child play; or even, scanned pictures of cartoon animals, as long as the children are involved in their selection and production.

- define responses – what should happen when the player fails? what about when the player gets it right?
- produce content and assign tasks – what are the actual images to be used? where are they coming from (drawn, live photograph, scanned image, objects made+photographed,...)? who takes part in producing these images?
- produce auxiliary items – where do "right" and "wrong" sounds or messages come from (recorded, written, etc.)? what should these messages say? are they appropriate?

This entire potential can be explored further, if the teacher develops this activity within a computer programming environment that is age-suitable, for then children can be involved in the production of behavior and responses themselves, as described further ahead in section 7.2.5.

The idea being presented here is: **if the computer is the final setting of the activity, then that activity can originate outside the computer beforehand** (animal parts identification with pets, toy animals, etc.), and in the computer itself it can involve pre-activities (drawing, in this case) aimed at bridging those original off-computer activities with the final intended computer activity.

7.2.3. Computer as source – "The larder is well-stocked!"

The converse of the example in the previous section occurs when the final activity takes place off the computer, rather than at the computer. Such as example is when the result of a computer activity determines something in the real world. Perhaps the simplest example is when the preschool teacher or a preschool child acts as referee or judge to confirm the eligibility of the computer activity (but, as described in section 7.2.5, all or part of this judging can be programmed by the children and/or the preschool teacher, depending on the level of complexity of the rules).

For instance, suppose that in Figure 250, instead of a bird and animal parts, there was the dark silhouette of a fruit and several fruit pictures as matching options.

Getting a proper match could also mean that the child was entitled to receive a real fruit (or a toy one, or a fruit card, *vd.* Figure 251), and stock it in a "larder" in the preschool room. That, in turn, could be an activity involved in a broader context, such as the benefits of keeping a larder stocked during winter, when one lives far from drugstores or supermarkets, or depending on roads that may become temporarily blocked by snow or landslides.

Figure 251 – Fruit cards and toy fruits, usable in preschool activities

From: http://www.hifivelearning.com/images/fruitVegCart.jpg and http://www.toy-choice.co.uk/graphics/refAgentaFruitinTray2.jpg

The idea being presented here is: **in the context of an activity that was planned to take place off the computer, the computer may be integrated, by turning it as a source of pieces to be used off the computer**.

Other examples of uses in this fashion include using the computer as a tool for recording the status of an activity, as is shown in Figure 252 (using it in the same way as a paper table posted on the wall, children can check those records to determine their current status later on), and using the computer to communicate by e-mail with another organization (and then wait for an answer to arrive).

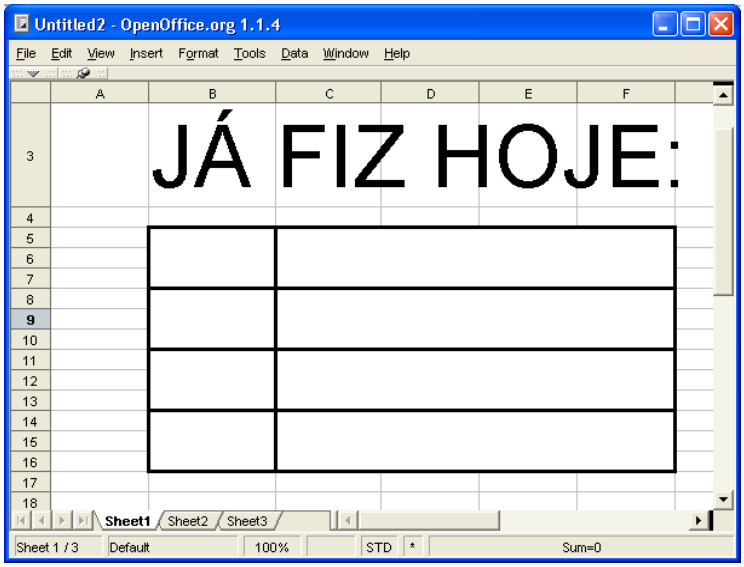


Figure 252 – Using the computer to keep status records as starting or returning point for activities

7.2.4. Mingled computer – "Who asked for bread!?"

The combination of the two previous ideas is that the computer can be both the destination of off-computer activities and a source of materials for off-computer activities. But this means that the computer can be entirely involved in an activity, rather than being an "extra" part of it.

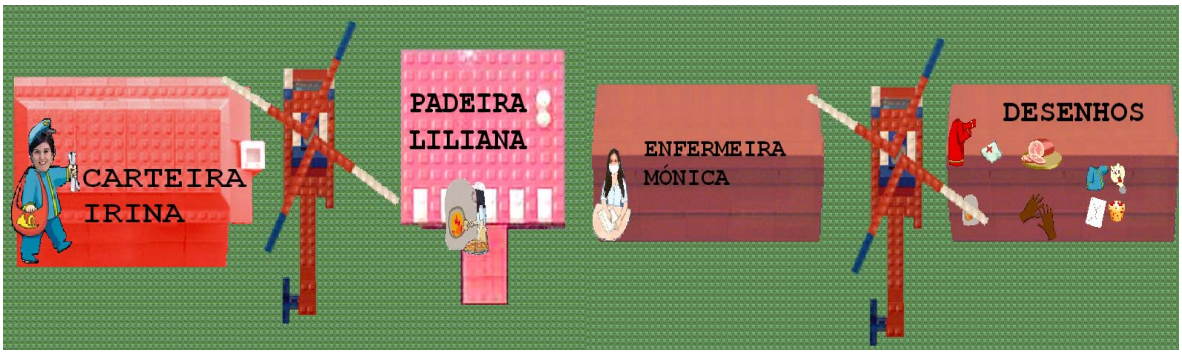


Figure 253 – City with houses for three professionals (plus a "pictures" house)

Figure 253 and the other figures in this section are from an activity developed[308] in the ToonTalk programming environment, based on the concept of interchanging services among professionals (this was part of research setting 6, described in section 5.3.7). The example presented in this section is based on this activity, but extends it in various ways, and this extended version is presented in full description in Annex XII. It is intended for use in a preschool room where the overall theme/project "professions" is already being developed. The mailwoman needs bread from the baker, who needs shots from the nurse, who needs letters and bread...

Children can develop many activities under the context of that general theme/project, and in that process assemble several pictures associated with professions. For this example, I'll consider the professions of mailwoman, baker and nurse. The pictures can be uniforms and items such as stamps, pastry, bread, etc.

Figure 254 – Postwoman's items and uniform

Some of the assembled pictures (originating in Web sites, books, children's drawings, etc.) can then taken into the ToonTalk environment, available for the children to use, along with other pictures brought in by the teacher (Figure 254). Each group of children, having been acquainted with a specific professional's "tools of the trade", can use the pictures to decorate a house for that professional (this was also done in the Mondrões preschool, during research setting 4, *vd.* section 5.3.5).

So far, nothing renders this activity different from the previous examples. But this decoration is just the activity preparation; the actual activity kernel takes place afterwards.

In the course of the preschool day, in another (off-computer) activity, the need may arise (or be created by the teacher) for a specific product – say, a cake from a baker. The child that needs that cake can go to the computer, enter the ToonTalk environment and enter the baker's house. There, that child can leave a request for a cake. This can take several forms: the number "1", the text "cake" and the name of the requesting child; or a picture of a cake and the photograph of the

[308] The sample activity described in this section was inspired in a much simpler version of it, developed between November 2003 and February 2004 by three second-year college students under the supervision of Leonel Morgado (Irina Brandão, Liliana Miguéis and Mónica Pereira, course "Computers in Teaching", Early Childhood Education baccalaureate, University of Trás-os-Montes and Alto Douro, Portugal), and further developed and assayed with four children during a traineeship, as described in section 5.3.7.

requesting child; or the picture of a cake and an object representing the profession role-played by the requesting child!

For instance, in Figure 255 several requests have been found at a house: top to bottom, there is a request for a parcel and a stamp from the nurse (identified by the nurse's cap); a request for a cookie, also from the nurse; a textual request for band-aid from the baker; and a request for a medicine shot from the postwoman. Clearly, some of these requests must be in the wrong house, since no professional can deal with them all (unless some child was role-playing a services contractor!).

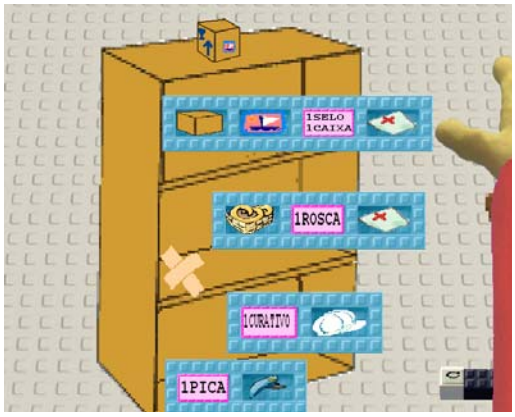


Figure 255 – Set of requests in a house

These seemingly "wrong" requests pose a nice learning opportunity. Supposing this is the nurse's house, the two bottom requests can be satisfied, but the others shouldn't be here, but rather taken to the appropriate houses (postwoman's and baker's). This can be used to start a preschool-wide debate on the organization of requests within a house. Should one's own requests be kept alongside requests from other people? Should one keep a "record" (*i.e.*, a copy) of the sent requests, and if so, should that copy by stored alongside external requests?

But what if this was another professional's house (a truck driver's house, for instance)? This child could call the child that placed an inadequate request there, and together decide how to solve the problem. However, the teacher could also take part, by prompting the child to decide how a wrong request should be handled. Should it be ignored? Should it be returned to the sender? Should the sender (the real child, a preschool colleague) be called so that he/she could learn about the mistake? Should the request be helpfully forwarded to the proper professional? And if so, should a note about the mistake be sent to the child who made it?

Getting back to the original request in this example, there was the need for a cake. A child would compose such a request, and place it in the baker's house. Other requests would also have been made during the school day. Sometime during the school day, the child playing the baker will go to the ToonTalk environment and find that there is a request for a cake. That cake can be delivered to the house of the child requesting it, who later will find it has been delivered.

Having been fulfilled the request for a cake, the teacher can be notified, and this can lead to another off-computer activity. For instance, a toy cake, a cake card, cake token or something to that effect can be handed to the child, for continuing with the activity that originated the request for that cake.

But there may not be the necessity for a physical cake token at all! The consequence of delivering the cake in the ToonTalk environment may be that a check mark is painted in a classroom "today's To Do list"; or that beans of toy money needs to be exchanged to "pay" the baker for that "virtual" cake!

The overall idea is: **a computer environment can be used not a starting point or ending point of a specific activity, but simply as yet another educational play setting in a preschool activity room, completely integrated in the context of other, off-computer activities taking place there**. Actions off the computer can require a computer action to be complete, and a computer result can lead to off-computer consequences.

7.2.5. Programming in context – "You've got it all, you can swim"

Environments like the ones presented in the previous examples provide a nice context for using programming skills and constructs, or even full programs.

In fact, the planning of the activity described in the example of section 7.2.4 already included the use of the computer construct known as "communication channels" (*vd.* section 3.4.12): ToonTalk's carrier pigeons would be used by children to render easier the sending of requests and the delivery of items. But rather than build up on one of the previous examples, I opted to present these ideas in yet another different setting.

The example[309] for this section is centred on sports activities and the necessary equipment. For instance, to swim one needs a bathing suit or shorts and a cap, to play tennis one needs a racket, tennis shoes and tennis equipment, etc.

Overall, there are many similarities between this setting and the one about professions in the previous example: instead of houses, one now finds buildings for sports; but each child or group of children is in charge of exploring several notions revolving around the kinds of sports taking place in that building, what equipment is necessary, etc.

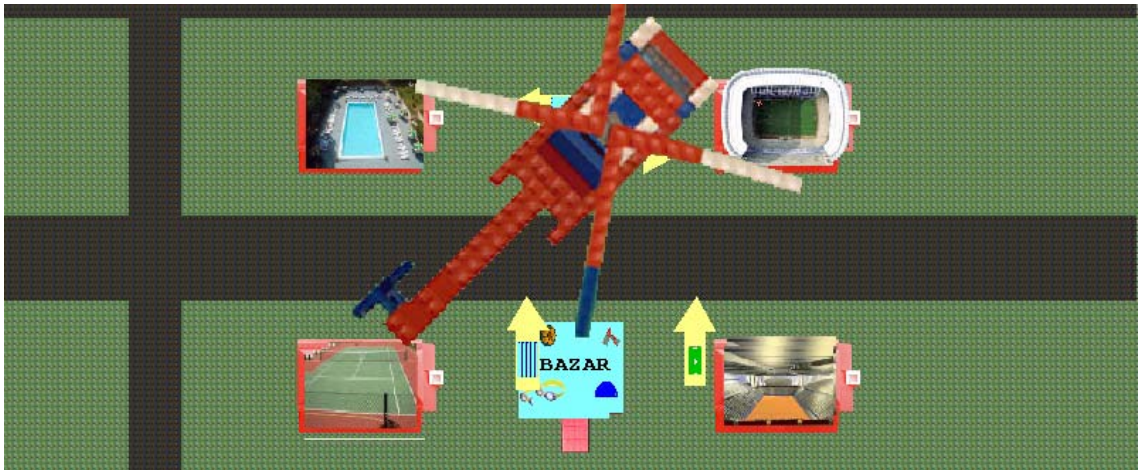


Figure 256 – ToonTalk city with buildings for sports

There is, however, a crucial difference: children don't live in the sports facilities. Rather, there can also be more houses in the city, where the children "live". And in those houses[310], pictures for the equipment used in several sports can be made available.

Now one can suppose that for some reason induced by an off-computer activity, a child needs to use a sports facility, such as the tennis court, for instance. That child goes to "her" house and must assemble all the necessary equipment for playing tennis. Taking that equipment into the tennis court (Figure 257), the child in charge of the court must check if all the required equipment is there.

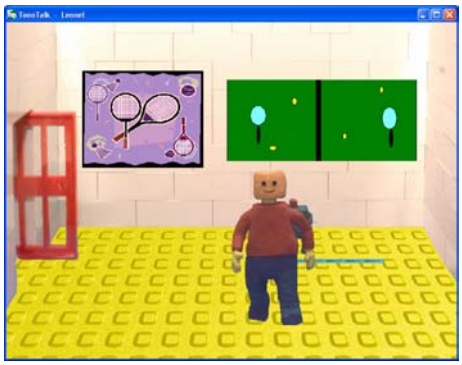


Figure 257 – Inside the tennis court

But since checking for the equipment is but a matter of comparing pictures, those could be standardized (*i.e.*, same racket picture for all children) and easily compared. So rather

[309] This example was inspired on a ToonTalk activity developed under my counseling, between October 2004 and January 2005, by two second-year students from the course "Computers in Teaching", Early Childhood Education baccalaureate, at the University of Trás-os-Montes and Alto Douro, Portugal (Maria Fernanda Figueiredo M. Silva and Susana Maria Alves Faria).

[310] In Figure 256, the students adopted a different approach: to have a central "*bazaar*" usable by all children. However, if children don't have their own place to customize or use as a base for activities, the development of activities is seriously limited, because each child lacks adequate control over the events of at least some part of the virtual city.

than having the child-court-caretaker to be there whenever another child wants to "play tennis", that child-caretaker can be suggested (or come up with the idea) of making a ToonTalk program (or several) to do the necessary comparisons[311]!

This setting places children in a situation where there is a nice motivation for the development of the interactive, animated cartoon-stories of ToonTalk programs. The programs can be as simple as asking, using a pre-recorded sound, "Did you bring your racket?" The child wanting to play tennis presents his/her racket and if the comparison succeeds, then a pre-recorded sound can state "you can play."

After playing tennis, the child can proceed with whatever was required by the activity that originated the tennis session (for instance, recording on paper that one tennis session was performed during a specific weekday).

Another way to conduct these activities is for the robot-caretakers to provide an "entry ticket" or some other token when a child presents the suitable equipment, and such tokens can then be presented to the preschool teacher or child acting as referee, or even printed, and used to access some other off-computer activity.

The overall idea is that **within an integrated activity there are often opportunities for including programming embedded in a larger context. Not just for a purpose, but for a purpose within a context**. Programming in such conditions becomes a tool for automation of the environment – an empowering concept.

[311] Currently, ToonTalk provides no primitive operation to check for the presence of an item within a list without iterating it. This prevents a generic "matching" robot from being programmed by virtually any preschoolers. However, the problem is simplified by instating the rule that the robot's thought bubble is not to be ignored, *i.e.*, children should present the sports equipment in the exact same order it is asked. This is far from being a perfect solution, but is it a usable solution nonetheless.

7.3. Typology of approaches to the programming environment

7.3.1. Usage typology 1: space for self-expression

To perform computer-programming activities in a visual environment, one of the very first problems faced by young children is their difficulty in controlling the mouse (Strommen, 1994; Hourcade *et al.*, 2004). To this, one must add the necessity of getting to know and exploring an unknown environment, where activities get ever more complex.

In the various research sessions, many children managed to develop from the very start activities with quite some level of complexity. However, in almost all cases these activities involved the constant support of an adult, due to the children's difficulty on manipulating and controlling the on-screen objects. Given than one wishes children to become gradually autonomous regarding the use of the programming environment, such a constant physical support is not desirable.

For this reason, a simple approach to programming can start by focusing on activities where children explore the programming environment and acquire some control over it. Specifically, children can handle the pre-existing programming environment objects, aiming to create "useful" results (in the sense of "personally-meaningful", *e.g.* ego-syntonic – *vd.* section 4.2.2).

In a virtual, animated programming environment such as ToonTalk, these results can be mere playfulness or preliminary environment customizations (*e.g.*, Figure 258, Figure 259, and Figure 260). But they can also involve the manipulation of images produced by children outside of ToonTalk, taken by adults (or the children themselves) into the programming environment, where they are simply yet another object (Figure 261).

Figure 258 – "Bird's mass", by J (boy, age 5), AA preschool, 2001

Figure 259 – "Painting", by J (girl, age 3), SPP preschool, 2001

Figure 260 – "Flower garden", by DU (boy, age 4), SPP preschool, 2001

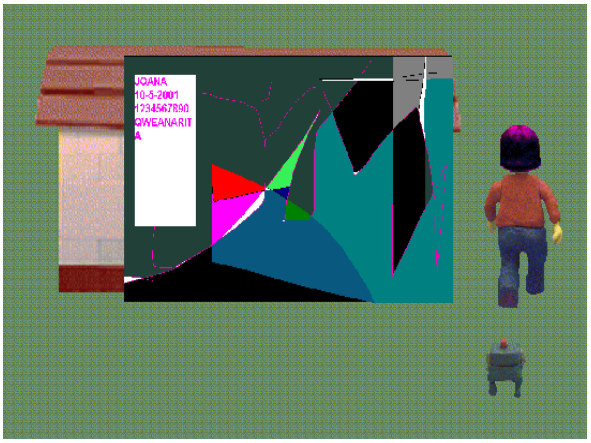


Figure 261 – Personal composition on the wall, by J (girl, age 4), SPP preschool, 2001

In other visual programming environments, which don't possess a virtual animated environment, this approach can also be used, at the very least by composing scenes and using the environment customization tools to produce meaningful scenarios or pictures ("meaningful", both as artistic creations or customizations, and as a graphical preparation and planning of more complex activities). A particular noteworthy case is Squeak Etoys (*vd.* section 3.3.4, p. 146), whose first children-aimed tutorial in the Web site Squeakland concerns the use of Squeak's Painting Palette to let children draw freely (anon., n.d.-2); these drawing tools can similarly be used to produce a meaningul scenario. A typical example is the car example from another Squeak Etoys tutorial, visible in Figure 81 (p. 147, reproduced below), which is already a preparation for another activity.

Another example is Stagecast Creator (*vd.* section 3.3.4, p. 151) where a children might draw or assemble the necessary pictures to compose a programming scenario, such as the starting picture of the iceman scenario, presented in Figure 87 (p. 152, reproduced below for the convenience of the reader).

Figure 262 – Environment customizations in Squeak Etoys and Stagecast Creator

From: vd. references of Figure 81 and Figure 87

In a tangible programming system such as the Valiant Roamer (p. 172), this can involve extremely personal customizations, involving physical items and actions. For instance, in research setting 4 (described in section 5.3.5) the computer-activities teacher at the Mondrões preschool let children customize ToonTalk, creating a "city of professions." This was a larger school-wide project, under which conducted many activities, some of them with a Roamer robot. Before those activities were initiated, however, children participated in the construction of cardboard houses, one for each professional, and they customized "turtle shell" for the Roamer robot, for the robot to "role-play" each profession (Figure 263).

Figure 263 – Roamer robot in disguise: flower seller, firewoman and chef

From[312]: João-Monteiro et al., 2003

[312] These photographs, made public by João-Monteiro *et al.* (2003) as generic customization examples, were actually devised by the computer-activities teacher, RC, at the Mondrões preschool, as described in this sections' text, and customized by the children there, under the project "city of professions".

7.3.2. Usage typology 2: sorting and organization

This is seen as a second level of complexity, in terms of activity planning. The basic idea is that it builds on the first usage typology (section 7.3) by integrating a few programming concepts and skills.

Primary elements of this, as was often the case during the research sessions, are activities involving data-organization skills. And these can range from ordinary thematic organization (*e.g.*, which images will be on the left side of the floor, and on the right side of the floor; which will be placed inside a house and which will be placed outside the house, etc.), to more complex and elaborate cases.

Central topics, for this typology, are:

- list manipulation;
- grouping;
- hierarchic organization.

In ToonTalk, one way of performing list (or vector) manipulation is by linking boxes (*vd.* section 3.3.5), forming a list, vector, or tuple, as shown in Figure 264. In this unordered example from the São Pedro Parque preschool, under research setting 4 (*vd.* section 5.3.5), the topmost box formed a list of items which had to be matched by the children below[313]. Other versions of the activity, however, could include ordering. For instance, in placing inside the boxes images depicting a specific moment in the Peter Pan story, as was done in the research setting involving computer-activity teachers.

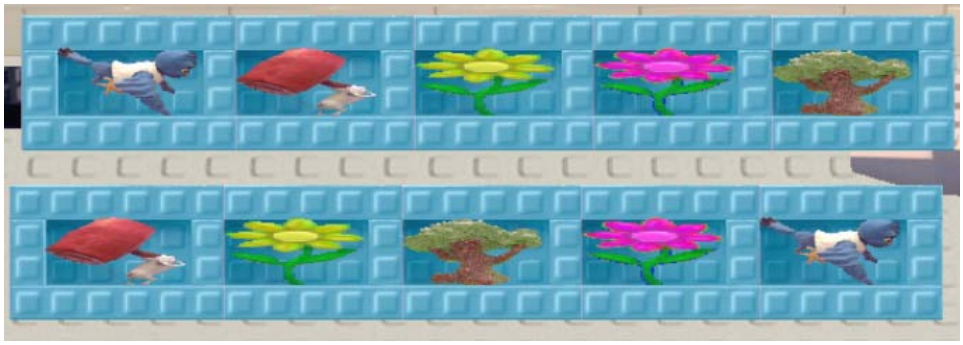


Figure 264 – List matching activity

Yet another way to use lists is to keep organized records. For instance, in Figure 265, from research setting 3 (*vd.* section 5.3.4), shows a list box containing a hole for each ingredient required to make soup (previously, all children had been working this theme within a project in the preschool room). Two children involved in the ToonTalk sessions were going to teach a robot how to make soup. But in order to do that, they had to draw the ingredients, and this list was used to keep the drawings they had already made (in this case, the carrot and the water drawings were ready).

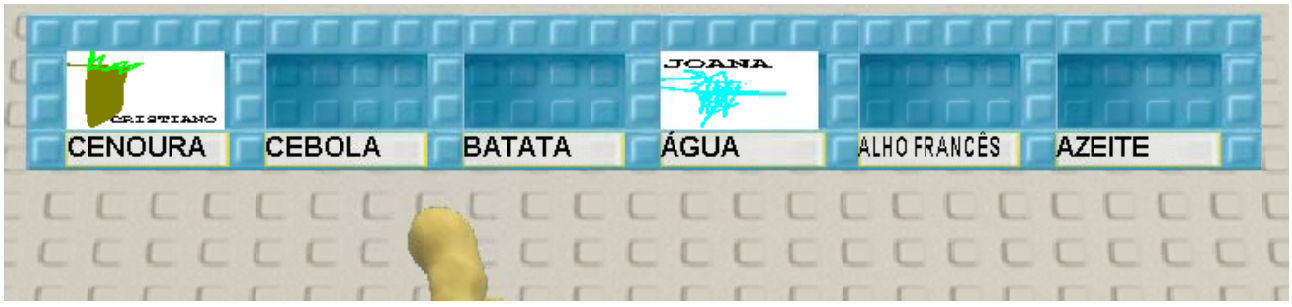


Figure 265 – List activity for organized record-keeping

[313] After performing this activity, some children, following a suggestion from the computer-activities teacher, taught a robot how to perform the matching for this particular case (not for any general case).

Another activity on the theme of sorting and organization is grouping. This can be achived visually, simply by bringing order to a disorganized set of elements. For instance, in Figure 266, on the left picture there are many scattered objects. Using the hand of ToonTalk's interface[314], children can bring order to this, separating the pictures into thematic groups. On the right-side picture of the same figure, for example, the objects were aggregated into "animals", "fruits", and "farm tools." This example is from the sample[315] activity presented in Annex XI.

Figure 266 – Organization by grouping elements visually

Such grouping provide an entry point for the last example of sorting and organization: hierarchic organization. In ToonTalk, this can be achieved with lists, by placing "lists within lists", *i.e.*, placing a box inside the hole of another. However, for the sake of providing a larger variety of examples, I'll present it using the other storage metaphor of ToonTalk: notebooks[316].

Hierarchic-organization allows the organization of Figure 266 to be stored with more space-efficiency (and in a easy-to-iterate format). Each picture is placed inside a matching notebook (in this example, one for animals, another for fruits, and yet another for farm tools). Then those notebooks can themselves be stored inside another notebook, which will hold the entire activity, as shown in Figure 267.

Figure 267 – Hierarchic-organization using notebooks

This organization can still increase in complexity, by increasing the total number of storage levels. For instance, animals can be divided into mammals, insects, etc., with a notebook for each kind of animal, inside a general "animals" notebooks.

In other programming systems, matching approaches to data organization can be devised, but only if they allow direct manipulation of the data, without resorting to commands.

[314] The background picture of a tree is covering the entire ToonTalk viewport.
[315] This example was inspired on a ToonTalk activity developed under my counseling, between October 2003 and January 2004, by three second-year students from the course "Computers in Teaching", Early Childhood Education baccalaureate, at the University of Trás-os-Montes and Alto Douro, Portugal (Elisabete Gonçalves, Ercília Carocha, and Sónia Carvalho).
[316] ToonTalk notebooks can be used for list manipulation activities, by placing an element on each page.

In Stagecast Creator, for instance, this approach would amount to letting the child programmer manipulate the elements, rehearsing them or trying them out (as illustrated in Figure 275), without actually being involved in establishing rules.

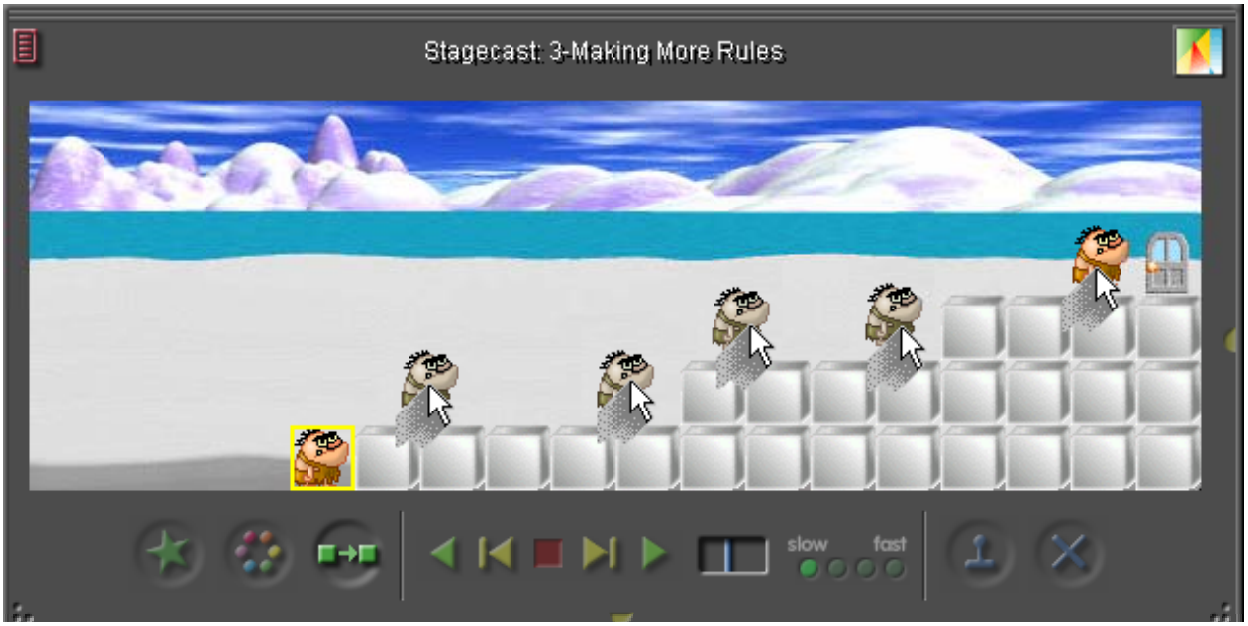


Figure 268 – Moving a Stagecast Creator character using the mouse, instead of programming rules

In many programming systems and environments, however, this approach does not stand out clearly. On some cases, such as some implementations of Logo (p. 139), the manipulation of data is done by issuing system commands or some other method of program specification, in which case one is employing the usage typology *no*. 4 (*vd.* section 7.3.4). On other cases, such as Squeak Etoys (p. 146) and some tangible-programming systems such as Electronic Blocks (p. 175), the manipulation of data implies changes in other environment elements, in which case one is employing the usage typology *no*. 3 (*vd.* section 7.3.3).

But often tangible programming systems allow for this style of "data manipulation" in a sense similar to the Stagecast Creator example, above. When the elements of a tangible-programming system are themselves programmable, as in the case of the Valiant Roamer (p. 172), the children can reproduce the intended movements and actions of those elements, in fact "rehearsing" the intended results of programming. But this rehearsal can be done even before children are aware of the available commands, or only aware of their limitations (*e.g.*, "the Roamer cannot jump"), and this is why I have chosen to consider such activities as part of this typology.

7.3.3. Usage typology 3: exploration of constructs and behaviors

Intertwined or built upon the second typology (section 7.3.2) one can develop a different kind of activity, which is framed under this third typology. And that is the exploration of automated behaviors possessed by some of the objects in the programming environment. For instance, in ToonTalk, when a bird receives any rectangular object, it takes that object into its nest.

In the course of such activities, the child is involved in the analysis of cause-effect relationships. The aim, within the scope of developing programming skills, is to lead the child into knowing ways to employ and combine these behaviors of objects, in order to achieve more complex goals. In short: develop plan-composition skills (*vd.* section 6.1.1).

Object behaviors can go from the extremely simple to the extremely complex. In ToonTalk, the aforementioned events that occur when giving an object to a bird are an example of a simple behavior. But what if the child has previously copied the bird's nest? In this case, the bird will clone itself, in order to take a copy of the object she was given into each nest (Figure 269).

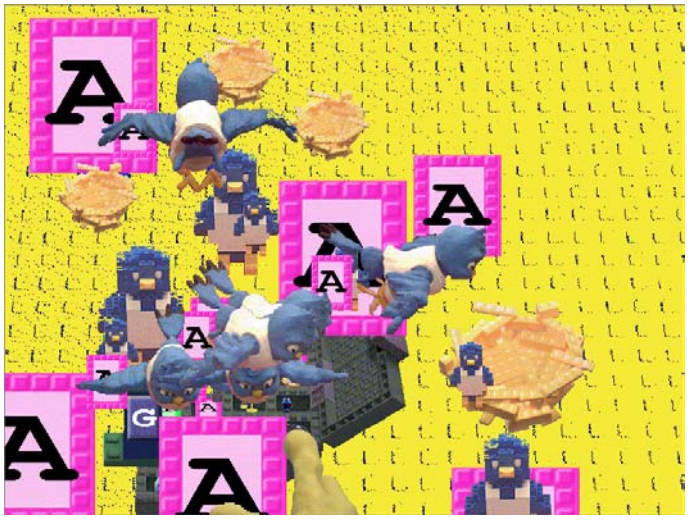


Figure 269 – Bird cloning itself to deliver an 'A' into several nests.

There are many ways to use this bird behavior. For example:

- to create, rapidly[317], a large number of copies of several objects;
- to conduct activities involving the distribution of objects by several different locations[318].

Other readily employable activities in ToonTalk are house-building (by combining a truck, a robot, and a box) and house demolition (by using a bomb inside a house), both of which (as reported in the various research sessions) were a widespread success among children (*vd.* section 5.3). Yet another ToonTalk activity in this fashion is comparing numbers with scales (to interpret the resulting tilting of the scales).

One particular kind of ToonTalk activities also falls under this typology, albeit not being usable with most children in preschool settings, due to its current reliance on text and numbers[319]: flipping an image (*vd.* Figure 197, p. 214), taking out some of its controls, such as "Width", and observe how the change in the control's values reflect changes in the image, and vice-versa. A

[317] Without programming a robot to copy nests, the child will have to create the copies of the nest by hand using the magic wand, but this only needs to be done once. Say a child copies a nest four times, getting five nests in all. From then on, whenever she needs 5 copies of any object, she just has to hand that object to the bird associated with those nests, to get 5 identical objects.

[318] A sample activity using this concept was summarized in section 7.2.4, and is fully described in Annex I.

[319] It might be possible to place graphical behaviors on the back of a picture to act as controls, but a nicer approach would be if ToonTalk allowed the creation of libraries of pictures whose control notebooks had been edited to replace numerical and textual controls by graphical versions.

similar approach (which is also similarly limited in its use by preschoolers) is found, for instance, in Squeak Etoys: in this system, when a childs acts upon an environment object (such as one of her drawings) she can observe the changes in the various properties of that object, such as the "Heading" value, as described and illustrated in section 3.3.4, p. 147.

ToonTalk also allows for a more complex approach, which was not explored during the research sessions. This would involve the combination of images with behaviors on their backs, such as those developed by the Playground project (Playground, 2001; Noss *et al.*, 2002) – an example is presented in Figure 270.

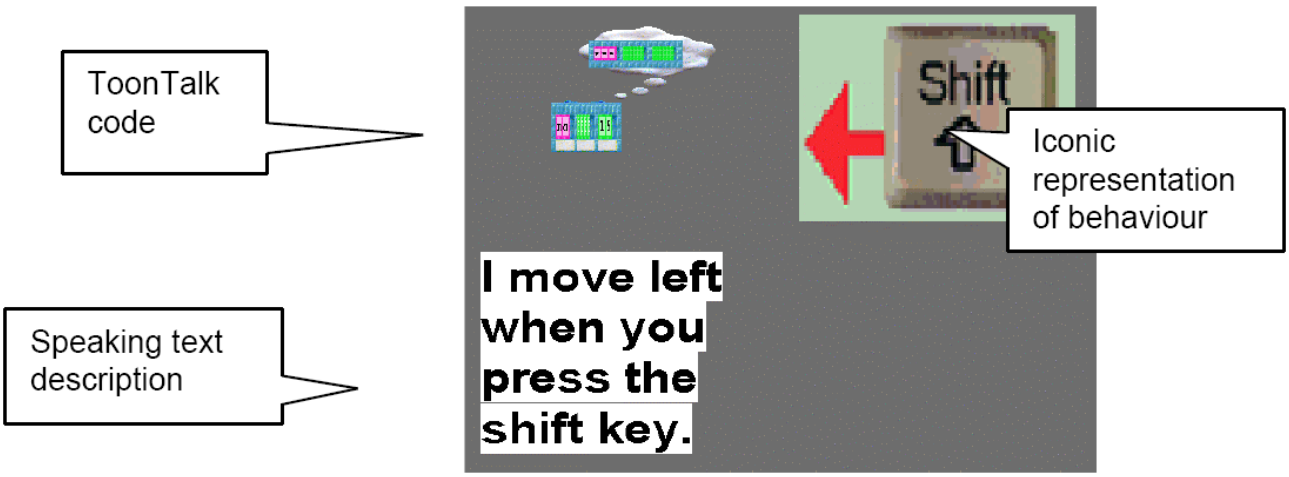


Figure 270 – ToonTalk behavior on the back of a picture, with two description alongside

From: Noss et al., 2002

As mentioned in section 3.3.5 (on p. 214), ToonTalk images are containers, which can be flipped around to place or remove contents from their back sides. These contents can also be subprograms, performing all sorts of activities and reacting to events affecting the images, which become "images with behaviors" that can be combined by simply dropping them on the back of each other. For instance, in Figure 271 the white square acquires vertical movement automatically, because a "I start moving up" behavior was dropped on its back. And it will bounce off the two smaller violet rectangles, because of the other behavior dropped on its back, "I bounce off things when I hit them moving up and down."

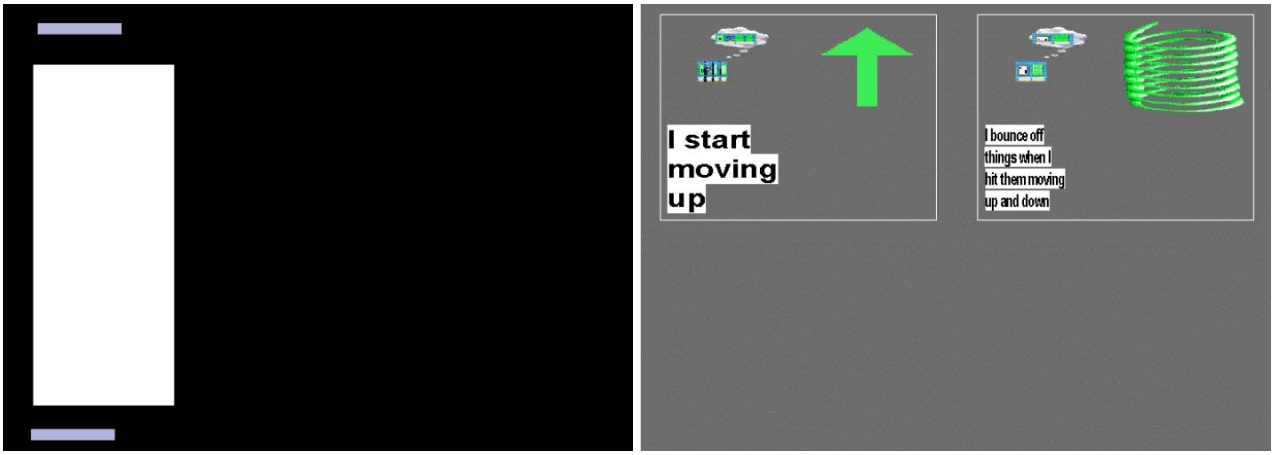


Figure 271 – The white square on the left picture, and two combined behaviors on its backside

From: Noss et al., 2002

In other programming systems, the approach presented in this typology is used by providing the children with programming elements that interact with other programming elements, responding to them or making them react.

Stagecast Creator, for instance, includes a tutorial which employs this style for starting-level activities in their systems: in the first section of the tutorial, children are expected to help a caveman character called Bungee find a Healing Root to cure someone in his village. To do so, the children have to interact with the environment while Bungee runs automatically. Children drag obstables out of the way, reposition Bungee, click on some objects to activate their special behaviors[320], in some cases use arrow keys to control Bungee, copy environment objects to build ramps, and so on. One of these situations is shown below in Figure 272: at this point in the first stage of the tutorial, children need to copy the available monkey twice; the monkeys possess a behavior that makes them move along the long tree branch with their tails and grab the bananas below, so that Bungee can move along.

Figure 272 – Stagecast Creator tutorial: the child must copy monkeys so they get the bananas out of the way

Another Stagecast Creator example is provided in Figure 273: the cat runs and the dog chases it. However, there is a rule (presented in the middle) that when the dog passes nearby a bush, it hides behind the bush. Children may try out the behavior of the dog, cat, and bush, and try out several combinations and settings, perhaps even coming up with a storyline, or "planting" more bushes to help the cat escape.

Figure 273 – Dog chasing a cat, rule telling the dog to hide behind a bush, and the result

[320] In the Stagecast Creator tutorial, at a certain point Bungee's path is blocked by a gigantic beehive, which when clicked upon lifts off and lets the bees out (Bungee then runs underneath it).

This overall approach is also easy to picture with tangible programming systems involving interacting objects. One clear case is the Electronic Blocks sytem (*vd.* p. 175). An example is shown in Figure 138, on p. 175 (reproduced below, for the convenience of the reader). As I described on that page: "*on the left, the touch sensor block is on top of the light action block, which therefore produces light whenever the sensor is touched. On the right, the movement action block has a seeing sensor block on top, which therefore moves whenever the sensor detects enough light. By placing the first set of blocks (remote control) near the second set (car), the car will move whenever the child put her/his hand in contact with the touch sensor.*"

Figure 274 – Remote control car with Electronic Blocks (identical to Figure 138)

From: Wyeth & Wyeth, 2001

7.3.4. Usage typology 4: instant programming = show how it is done

This level encompasses activities where all commands can have a direct, immediate consequence. In traditional text-based languages such as Logo (*vd.* p. 139), this would consist in issuing commands to the interpreter for immediate execution, or defining a single procedure or function and use it by itself, without further combination with other such subprograms.

In ToonTalk, the basic programming concept, detailed in section 3.3.5, is that this basic elements of programming are done by demonstrating to a robot how to follow the steps required to perform the intended action, starting from an initial set of values (a sample program is provided in Annex I, p. 441).

When involved in educational activities developed under this typology, children face the need to reason over the general sequence of activities that compose the classical definition of programming (Blackwell, 2002): defining requirements; perform a specification; design the technical features; perform the coding; and "*anticipate and account for departures from the intended behaviour (debugging)*" (*id.*, *ibid.*). I am not referring to a complex program of various interacting elements; rather, I am referring to the programming of a single procedure (in ToonTalk, a single robot).

For instance, in Figure 275 two girls had shown their robots how to type their names[321], following the style presented in Annex I (starting with an empty box, pick up a text pad, write the name and drop the name in the box). However, since there were no starting elements in the boxes of their robots to constrain the robots' operation, they kept on writing their names, over and over again. The "final" result pleased the children, but was also an opportunity to reason, by analyzing why this happened in this manner, in order to experiment with how to avoid it (or, in cases such as this, where the result is pleasant, how to make use of it).

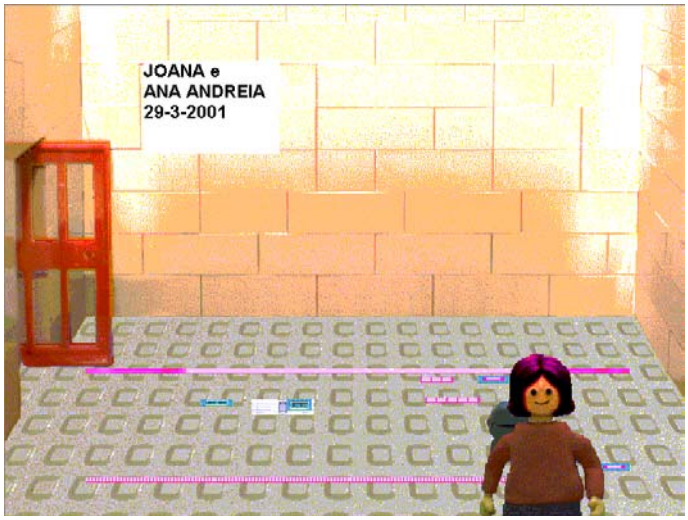


Figure 275 – Robots typed the children's names many times (on the floor).

Even adults find it difficult to reason under these parameters. This much is a recurrent theme in programming literature (*vd.* chapter 1), but I also had the opportunity to construct my personal first-hand knowledge and notions in several contexts, as described in section 6.2. One specific support item, developed as a conceptual support for both adults and children, was an inquiry-based strategy of facing the programming process (Morgado *et al.*, 2003a), presented in section 6.2 (Table 34, p. 353).

This typology maps onto virtually any other programming system, since it represents the typical entry point found on presentations of the syntax of most programming systems. Notable exceptions are Stagecast Creator and Squeak Etoys, whose tutorials propose simpler entry points, as

[321] This took place in a session during the second research setting, summarized in Annex I and detailed in Annex I.

described in sections 7.3.1 and 7.3.3. Even these two systems, however, provide minimal exposure to those other entry points, and soon[322] move on to examples that fall under this typology.

For instance, Figure 276 on the right, shows part of the third Squeak tutorial at the Squeakland site (anon., n.d.-2). In this tutorial, the child is invited to draw a car (this part of the tutorial fall under the first typology) and then try out one of its default scripts; finally, under the section "Make A Script", the child is invited to create a new script, rather than try out a pre-made one.

Similarly, Figure 277 shows a final section of the second Stagecast Creator tutorial, where the child is invited to make a rule determining how to make an alien move to the right.

Figure 277 – Final section of the "Making Your Own Rules" activity in the Stagecast tutorial

In the field of tangible-programming systems, examples of usage of this methodology are common. For instance, Valiant Technology Ltd., the company responsible for the Valiant Roamer (p. 172), published an "Activity Book" where right from the start children are encouraged to try out issuing commands and checking out their results (Valiant Technology, 2004).

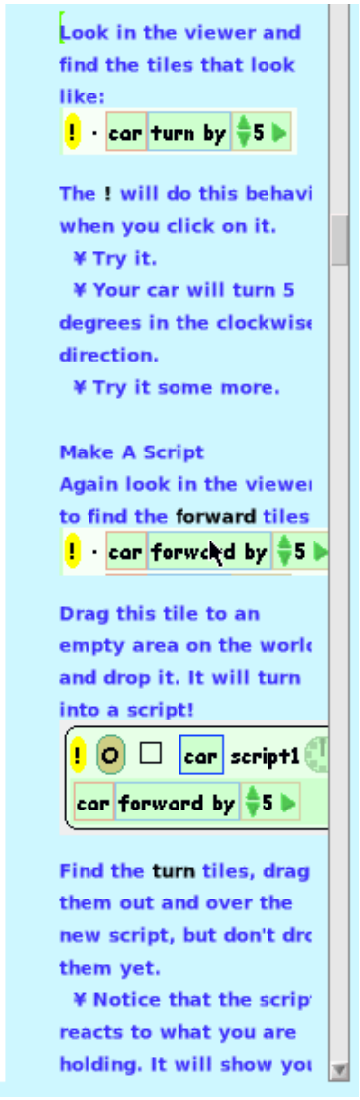


Figure 276 – Sample text from the "Make your own Car" Squeak tutorial
From: http://www.squeakland.org/project.jsp?/projects/etoys/NewCarTutorial.001.pr

[322] In the Stagecast Creator Tutorial, this is the second activity; in the list of Squeak Etoys tutorials at the Squeakland site (anon., n.d.-2), these activities are part of the third tutorial.

7.3.5. Usage typology 5: combining programs

This latter phase of activities represents the combination of simpler programming constructs, thus being virtually expanded to encompass all the complex activities commonly associated with the idea of programming.

At the level of preschool children, the entry event into this typology is when children start to combine several subprograms, creating chains of actions or behaviors. This can be seen as an application of two different concepts, presented earlier in the Cookbook of Computer Programming Topics (section 3.4): **compound procedures** (section 3.4.6) and **high-order functions** (section 3.4.9); respectively, the referral to various operations as a single unit, and the combination of the operation of various procedures or functions.

As described in those sections, in ToonTalk this is done either by combining the results of a behavior with another (or with a robot), or by chaining several robots (*i.e.*, program robots counting on the actions of other robots, by communicating with birds, being integrated in teams, or using some other form of coordination). Another method, which was not described in those sections, is to join pictures with specific behaviors, combining them to produce a more complex behavior or what might be considered traditionally as a "full program."

When describing the third typology, in section 7.3.3, the nature of such pictures with behaviors was explained (Figure 271 shows a picture with two behaviors on its back, and the caption describes the result), resorting to some behaviors developed by the Playground project (Playground Project, 2001). In the course of the Playground project, children aged between 6 to 8 combined behaviors such as "I blow up when I touch something", and "I follow the arrow keys", to create game characters and develop and modify computer games (*e.g.*, Noss *et al.*, 2002; Tholander *et al.*, 2002). At the preschool level, the children successfully understood the concept of robots working behind pictures, during the sessions of the second research setting (summaries and highlights in Annex VII, full reports in Annex VIII). This pointed towards the feasibility of employing Playground behaviors with preschool children, but my research did not explore this path further. Instead, my research approached the more finely grained case of chaining robots together. During the research sessions, two methods of coordination were experienced.

The most frequently witnessed coordination mechanism was the launch of a separate process (loading a truck with a robot). Typically, a child would teach a robot how to place her name on the roof, or a picture on a room wall, and then she would launch a truck and load it with that robot. This way, two events are coordinated. First, the truck launches a new process (build a new house). Second, the robot is put to work inside that process/house, placing the child's name on the roof or the child's picture on the room wall.

Figure 278 – R (boy, age 5) in his new house, where the robot placed its gift on the wall

A further evolution of this mechanism was witnessed, by employing another robot, to launch the trucks in place of the child. For instance, Figure 278 shows the result of one such activity: having programmed a robot to place a "gift" on the room wall, the child R programmed a different robot, to build houses (something he had prior experience doing). Finally, he combined these two robots, therefore managing to have a robot creating houses which automatically display "gifts" on the wall, as result of the operation of the robots sent into the new houses. Putting names one the roof instead, children realized whose robot made which house.

The other witnessed coordination method was the use of birds. This method was actually the first one to be tried out, as soon as the first research setting (summarized in section 5.3.2, detailed in Annex III).

Figure 279 – Robot giving to a bird the received ball, to send it back to the child

Both in that research setting and in later settings, a common activity would be for children to employ birds to send a ball to a robot. The robot would then reply by sending the ball back through another bird (as shown in Figure 279). This approach was successfuly employed with children aged four and five, and even with a three-year old child in the fifth research setting (section 5.3.6).

Regarding other programming systems besides ToonTalk, this situation regarding this typology is identical to the fourth one, described in section 7.3.4. Back then, I said: "*This typology maps onto virtually any other programming system, since it represents the typical entry point found on presentations of the syntax of most programming systems.*"

The only difference is that, obviously, this fifth typology is not the typical entry point, as was the case in the fourth typology. Nonetheless, it remains a common approach to programming, the variety being only in how soon the different systems propose that children get involved in combining procedures or functions. For instance, in Stagecast Creator, the tutorial presents the combination of independent "rules" as the third activity, and their combination, by organizing their order of priority, as the fourth activity, as shown in Figure 280. In Squeak Etoys, the third tutorial already incorporates activities in this level, since it ends by showing how to program two pictures – a car and a steering wheel – to act in concert (this was already described in section 3.3.4, p. 148).

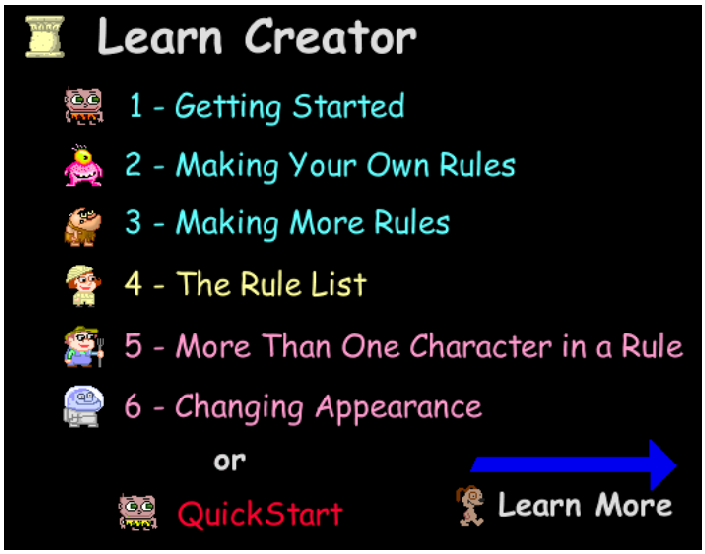


Figure 280 – List of activities from the Stagecast Creator tutorial

8. Final remarks

I welcome the reader to this final chapter in my thesis – hopefully, a read that may have proved so far not only informative, but also pleasant. Through it, the reader was presented with a cookbook of sample activities for preschoolers, each employing a different computer programming topic, which I hope educators can now find more edible (section 3.4). Then, after the presentation of the field activities (chapter 5), the reader further progressed along my analysis of children's hurdles and issues faced by their teachers – and some approaches to assist on both accounts (chapter 6). Finally, the reader has found in the previous chapter my framing of computer programming activities in view of planning or devising their integration in the everyday activities of preschools.

'What now?', other researchers may ask;

'What can I do on Monday?', early childhood teachers may ask.

Too often research has sought impacts or effects of computer-use as if it were an environment chemical, rather than an instrument. And an instrument can have its specific features, but a central issue is its use. How can computer programming be used in learning environments? There is the need for research to build at the very least a half-baked, half-acceptable answer. In my view, both researchers and practitioners must strive to come to this answer, each in his or her particular way.

The question itself has been amply debated and answered, but mostly at two extremes of its scope. At one end, there has been an adequate provision of 'big ideas', such as the philosophic analysis of its potential on learning; at the other end, there are various examples of programming projects developed with or by young children (and in this thesis I have also provided a few), albeit usually lacking adequate integration with other on-going activities in preschools. It is across this mid-ground of everyday integration within preschool practice that there is simply too little information. There has been ample debate on obstacles poised by current educational and organizational practices in formal schooling levels, but these settings bear little resemblance with the educative environment and practice at the preschool level. Even the experience and know-how developing at the Constructionist Learning Environment (Stager, 2005) is on older, literate students.

For researchers wishing specific targets, further in this chapter I will provide some topics, questions, and hypotheses that popped up during my field research and later reflections, and may serve as inspiration for further research. But first, I want to address the concerns of teachers.

On that mid-ground I mentioned above, dealing with actual field practices for education professionals should be, I feel, one of the most important lines of applied research. There is the need for the development of something in the line of a curriculum model for the immersion of computer programming in common practice. Thus, I would answer to the teachers that the problem isn't about Monday or Tuesday – it's quite easy to come up with two or three scattered programming-based activities. Ask me rather, 'how can I use this from September to July, without confining it to a specific time or a specific weekday?' Better still, ask me 'How can I use this in the same way I use ink, paint, wood, glue, sand, and stone? Mind-ink, mindpaint, mindwood, mindsand, mindstone?'

I believe to have partly contributed to building the answer, but still much applied research is required. And not just that which can be done within the confines of an educational laboratory or independent weekly sessions, such as my own. It is also necessary for research based on the trial and development of detailed models of practice, of detailed accounts over a long time period, so that teachers can build a personal relation between the large ideas, the accounts of others, and their own practice. Such contributions call for research on the teacher's practice in the field, possibly both from the teachers' own perspective as teachers-researchers, and from researchers not simultaneously acting as teachers.

This concern has been in the back of my mind throughout my research for this thesis. Even while focused on some other detail or on a different set of issues, I have always been concerned on how they might be used by people on the field, not just myself. Hence, I would like to disappropriate my framework: I'd like it to be seen as a frame-for-work, and frame-in-work, for people to turn it into their own.

As an example of a specific issue to which teachers' own research contribution would be central, one can consider the attention-span of teachers to each children. Teachers need to deal with more than a few children at a time, so the level of autonomy that children can have in the use of programming is an essential issue to tackle. In view of the cognitive development and effort involved, this is not something likely to ever be immediate or entirely intuitive, so preschool teachers need to know how to build it up without resorting to long sessions entirely devoted to a single child or pair of children. This is something that shouldn't be addressed just at the level of software or hardware design – even though much more can certainly be done at this level.

From their part, computer scientists, at the preschool level, have mainly been involved in research on technology development. But there is a need for more educational observations and interactions from these professionals. A background issue with educators using computer programming is a common personal misinterpretation of its core features and significance; and until such time as programming becomes a common feature of most individuals' educational development, I don't think it likely that such a background will change. By cooperating with education professionals, the particular perspective of researchers from a technical background can provide the educational researchers with new insights.

I sometimes imagine an approach similar to that of Reggio Emilia schools, but where children and teachers benefit not just from the support of an art educator, but also from that of a computerist, someone with an awareness for the potential of technology on the development of thinking, not just its fireworks. I don't know if or how this might work. After all, there is the danger that instead of having the computerist assist the development of efforts by the children and their teachers, he/she creates a separate line of activity – in effect subverting the whole purpose of his/her existence. The same might happen if teachers decide to assign computer-use to the computerist, instead of putting the computerist's know-how to everyone's use. It could also happen that political decision-takers or educational managers decide that a computerist can be anyone, from any background, if provided with a little "add-on" training on the use of computers and programming. I fear this would be disastrously foolish: the background is the key.

Technology, of course, must also evolve, not just the educational practice. I believe the development of ToonTalk was a breakthrough in the course of children's programming. Beyond all its particular features and technical developments, one essential aspect is that in its environment children are not just programming something else, they can actually be part of the program. They can interact with a program while it is running, they can manipulate objects in which various programs are working, and to which various other programs may respond.

Furthermore, children are not part of the program in an abstract way, they are moving around within a common environment: a city with houses, boxes, birds, trucks. And most significantly, they discover that there are a few objects in that city that would tipically be off-limits, such as the helicopter, and bombs. In fact, for young children even something as simple as a rooftops is usually off-limits, as is walking around a city alone. There is a sense of freedom in this city, and a sense of empowerment of the child user. No other language had before rendered this possibility so real.

These aspects can be further developed. ToonTalk's objects and environment have been developed in service of programming, and the most advanced examples of ToonTalk programming are programmed pictures – not programmed houses, trucks, birds, roofs or roads. An exciting development would be the inversion of these priorities, putting programming at the service of the environment, while retaining its power. Prior to ToonTalk, this sort of advanced-scripting approach would likely be deemed unviable in a children-oriented language. But ToonTalk has raised the possibility of a pathway between concrete manipulation of elements in a virtual world, their specialized programming, and general-purpose programming.

As a child, before I knew how to program, I dreamed of a virtual world. Not a world resembling reality, but more like outer space, somewhat of a "inner space" to reality, in which I might fly around in a special suit, placing cubes, balls, translucent tubes and wobbly gelly shapes

together – and in doing so, programming the operation of computer-game elements. Since then, ToonTalk is the closest I've ever been to such a dream.

And so finally, for the researchers that reached this point while still wishing for more specific new research objectives, here are my suggestions:

1. The hurdles described in section 6.1.2 were developed from the events on ToonTalk sessions. What changes would the use of other programming languages and environments suggest to them?
2. How much can children achieve, in terms of complexity of their programming, over the course of their entire preschooling (3 years), with the support of a researcher or computer-activities teacher, using programming once a week? And using programming several times per week, integrating it with the on-going events at the preschool?
3. Starting from the framework provided in this thesis, there is a need for the development of large numbers of case studies and activity trials, to refine it or ultimately replace it.
4. Children's use of abstraction using ToonTalk has been promising. Specific research can be done regarding the strategies for its development and understanding, beyond my "usefulness" approach. These could include small focused experiments, like the one suggested in section 5.2.4, on p. 330.
5. My research was based on common traits of children between the ages of 3 and 5. There is a need for research that looks into age and gender differences.
6. There are certainly connections between Piaget's limitations of preoperational thought and the hurdles I identified. Some of these seem almost obvious (*vd.* footnote 243, on p. 259), but only fine-tuned research can support such assumptions, disprove them, or shed some light on their interactions.
7. When I initiated the research activities of setting 4, with computer-activities teachers (*vd.* p. 335), I had proposed three different approaches to introduce programming to children, which were subsequently abandoned. But as I mentioned in chapter 7, my framework aims to assist the teachers in designing, planning, and customizing programming activities – it is not a hierarchical way to introduce children to programming. Research can revisit my original three approaches and also develop and try out others.
8. Why do most adults in the educational area, when shown ToonTalk, think it will not be successful with young children? As I said in section 6.1.1: "*it would be interesting for psychological and sociological studies to be conducted in order to determine factors influencing both this strong engagement by children, and this common skepticism by preschool education professionals*."
9. Do freedom and empowerment, as I mentioned earlier in this chapter, really play a role in children's enjoyment of ToonTalk? What are the features of that role?
10. Certainly, many more strategies can be devised to help children overcome the various hurdles, but it's definitely critical to research ways to overcome the one named "*Children find it hard to consider the possibility of being wrong and not knowing it*".
11. The cookbook activities can provide interesting case studies, at various levels: their actual feasibility, the main difficulties faced by children in developing them, and the issues teachers came across in their course, as well as the strategies employed.
12. Are teachers (or trainee teachers) more effective in developing their planning skills if they progress by devising activities in the sequence of complexity presented in section 7.2.1, in some other sequence, or is this not a relevant factor?

13. The training and management approach style of weekly meetings with computer-activities teachers, as in research setting 4, might itself be a focus of research, by comparing the events at such meetings with the evolution of the activities conducted autonomously by those teachers at the preschools. I am referring not only to working methods, but also (indeed, mainly) to teachers' refinement of their understanding of the bigger ideas behind the use of computer programming.
14. The use by preschool children of ToonTalk behaviors, such those developed by the Playground project (*vd.* p. 374), seems feasible, and an interesting area of research.
15. Another potentially important area of research is that dealing with the adaptation of programming terminology to children, as shown in Annex VI. Such an approach has also been mentioned in other research papers (Hildreth, 1975, pp. 6-10), but it would be useful to understand more about the actual benefits of such terminology adaptations, as well has knowledge about successful and failed terms.
16. The relative sizing of elements is an issue that could be scrutinized by research. Enlarging objects in ToonTalk or other computer environments may possible simplify their identification and selection by children. But wouldn't the consequent reduction in available free screen area complicate matters? What about the events of unintended collisions, unintended dropping of objects into boxes or notebooks, or other interactions between large elements? There is need for analysis of children's interactions with screen objects of various sizes, at different resolutions and different dimensions of monitors.

9. References

Abbagnano, Nicola; Visalberghi, Aldo (1959). *Linee di storia della pedagogia*, vol. III, G. B. Paravia e C., Turin, Italy. Referenced from the Portuguese translation by Glicínia Quartin (two volumes, 1981 and 1982), "História da Pedagogia III"/"História da Pedagogia IV", Livros Horizonte, Lisbon, Portugal.

Abelson, Harlod; Sussman, Gerald Jay; Sussman, Julie (1996). *Structure and Interpretation of Computer Programs*, 2nd edition, ISBN 0-262-51087-1, The MIT Press, Cambridge, MA, USA. Referenced from the fifth printing, 2000.

Abelson, Harold (1982). *Apple Logo*, ISBN 0070004250, Byte Books, Peterborough, NH, USA.

Adams, Joel (no date). *CS 214 Lab 4: Controlling Behavior: Repetition*, Web page at http://cs.calvin.edu/curriculum/cs/214/adams/labs/4/, Computer Science Department, Calvin College, Grand Rapids, MI, USA. Retrieved on September 7th, 2004.

AgentSheets, Inc. (2002). *Getting Started with AgentSheets*, AgentSheet manual, AgentSheets, Inc., Boulder, CO, USA. Referenced from the on-line version at http://agentsheets.com/Documentation/windows/Getting-Started.pdf, retrieved on November 9th, 2004.

Agha, Gul; Hewitt, Carl (1986). *Concurrent Programming Using Actors: Exploiting Large-Scale Parallelism*, A.I. Memo 865, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Referenced from the on-line version, retrieved on August 12th, 2004, from ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-865.pdf.

Alborzi, Houman; Druin, Allison; Montemayor, Jaime; Sherman, Lisa; Taxén, Gustav; Best, Jack; Hammer, Joe; Kruskal, Alex; Lal, Abby; Schwenn, Thomas Plaisant; Sumida, Lauren; Wagner, Rebecca; Hendler, Jim (2000). *Designing StoryRooms: Interactive Storytelling Spaces for Children*, in "Proceedings of the conference on Designing interactive systems: processes, practices, methods, and techniques", ISBN 1-58113-219-0, pp. 95-105, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/347642.347673 on June 11th, 2004.

Alexandrov, Kirill; Soprunov, Sergei; Yakovleva, Elena (1997). *Logo for the illiterate programmers*, in "Eurologo'97 Proceedings", on-line proceedings of the Sixth European Logo Conference, held in Budapest, Hungary, 20-23 August, 1997. Retrieved on January 28th, 2004, from http://eurologo.web.elte.hu/lectures/kirill.htm.

Alves-Mazzotti, Alda Judith; Gewandsznajder, Fernando (1998). *O Método nas Ciências Naturais e Sociais: Pesquisa Quantitativa e Qualitativa*, ISBN 85-221-0133-7, Editora Pioneira, São Paulo, Brazil.

Ambassade de France au Brésil (2002). *O acolhimento e a guarda na primeira infância*, information sheet, May 2002, Ambassade de France au Brésil, Brasília, DF, Brazil. Referenced from the on-line version, retrieved on May 5th, 2005, from http://www.ambafrance.org.br/abr/imagesdelafrance/Formato%20PDF/infancia.pdf.

Amory, Alan; Naicker, Kevin; Vincent, Jacky; Adams, Claudia (1999). *The use of Computer Games as an Educational Tool: 1. Identification of Appropriate Game Types and Game Elements*, in "British Journal of Educational Technology", ISSN 0007-1013, vol. 30, no. 4, pp. 311-322, British Educational Communications and Technology Agency, Coventry, UK. Referenced from the on-line version, retrieved on January 27th, 2004, from http://www.nu.ac.za/biology/staff/amory/bjet30.rtf.

Andrews, Gregory R.; Schneider, Fred B. (1983). *Concepts and Notations for Concurrent Programming*, in "ACM Computing Surveys", ISSN 0360-0300, vol. 15, no 1, pp. 3-43, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on February 13th, 2005, from http://doi.acm.org/10.1145/356901.356903.

Anonymous (2003). *¿Trabajas con LOGO? Base de datos*, Web page, retrieved on August 20th, 2005, from http://roble.pntic.mec.es/~apantoja/trabajas/bdatos.htm.

Anonymous (2004). *Computer Programmer*, in "The Odyssey2 Homepage". Electronic resource at http://www.classicgaming.com/o2home/db/cart.asp?masterid=12&cartid=74, retrieved on March 5th, 2004.

Anonymous (no date-1). *Active Essays*, Web page at the Squeakland Web site "www.squeakland.org", Viewpoints Research Institute, Inc., Glendale, CA, USA. Retrieved on September 18th, 2004, from http://www.squeakland.org/author/essays.html. Last accessed on August 9th, 2005 at http://www.squeakland.org/whatis/a_essays.html.

Anonymous (no date-2). Etoys, Tutorials & Other Goodies, Web page at the Squeakland Web site "www.squeakland.org", Viewpoints Research Institute, Inc., Glendale, CA, USA. Retrieved on July 2nd, 2004, from http://www.squeakland.org/author/etoys.html. Last accessed on August 9th, 2005 at http://www.squeakland.org/whatis/tutorials.html.

APA, American Psychoanalytic Association (no date). *About Psychoanalysis*, Web page at http://www.apsa.org/pubinfo/about.htm, American Psychoanalytic Association, New York, NY, USA. Retrieved on July 7th, 2005.

Arcà, Andrea (2002). *The EuroPreArt Database System*, in "TRACCE Online Rock Art Bulletin", October 2002, Footsteps of Man - Le Orme dell'Uomo, Cerveno, Italy. Retrieved from http://www.rupestre.net/tracce_php/modules.php?name=News&file=article&sid=1 on August 18th, 2005. Also available at http://www.europreart.net/structure6.htm.

Armstrong, Joe (1997). *The development of Erlang*, in "Proceedings of the second ACM SIGPLAN international conference on Functional programming", ISBN 0-89791-918-1, pp. 196-203, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/258948.258967.

Askildsen, Tormod; Chioccariello, Augusto; Manca, Stefania; Munch, Gaute; Sarti, Luigi (2001). *A cybernetic construction kit for young children*, in CNR-ITD, LEGO A/S, CRE, HLK (eds.), "Construction kits made of Atoms and Bits. Research findings & perspective", technical report RT/ITD 4/01, Istituto Per Le Tecnologie Didattiche – ITD, Genoa, Italy. Referenced from the on-line version at http://cab.itd.ge.cnr.it/cab/partnersonly/deliverables/del25/Del25.PDF, retrieved on August 10th, 2005.

Backus, John (1978). *Can programming be liberated from the von Neumann style? A functional style and its algebra of programs*, in "Communications of the ACM", ISSN 0001-0782, vol. 21, no. 8, pp. 613-641, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/359576.359579.

Barach, John (1999). *The Origin of the Automobile*, in "Automobile History", Web site at http://www.motorera.com/history/history.htm, last accessed on May 18th, 2005.

Baranauskas, Maria Cecília Calani (1993). *Procedimento, Função, Objecto ou Lógica? Linguagens de programação vistas pelos seus paradigmas*, in José Armando Valente (ed.), "Computadores e Conhecimento: Repensando a Educação", ch. 3, Gráfica Central da Universidade de Campinas, Campinas, SP, Brazil. Referenced from the on-line version, retrieved on June 10th, 2005, from http://www.nied.unicamp.br/publicacoes/separatas/Sep3.pdf.

Barber, Matt (2004). *ZX81 Home Page*. Electronic resource, retrieved from http://www.honneamise.u-net.com/zx81/ on March 5th, 2004.

Basu, Anindita (2003). *[Squeakland] Logo vs. Squeak*, e-mail message sent to the mailing list squeakland@squeakland.org on August 9th, 2003, at 12:55 PDT. Retrieved on September 19th, 2004, from the on-line archive at http://squeakland.org/pipermail/squeakland/2003-August/001750.html.

Battista, Michael T.; Clements, Douglas H. (1991). *Using spatial imagery in geometric reasoning*, in "The Arithmetic Teacher", ISSN 0004-136X, vol. 39, no. 3, pp. 18-21, National Council of Teachers of Mathematics, Reston, VA, USA. Referenced from the on-line version, retrieved on August 5th, 2005, from http://investigations.terc.edu/relevant/UsingSpatialImgery.html.

Bauer, Friedrich L. (2002). *My years with Rutishauser*, LATSIS Symposium 2002 - ETH Zurich, Switzerland. Referenced from the electronic version, retrieved on May 20th, 2004 from http://www.cs.umd.edu/users/oleary/cggg/bauer.pdf.

Bauer, Friedrich L. and Wössner, H. (1972). *The "Plankalkül" of Konrad Zuse: A Forerunner of Today's Programming Languages*, in "Communications of the ACM", ISSN 0001-0782, vol. 15, no. 7, ACM Press, New York, NY, USA. Referenced from the electronic version, retrieved on May 20th, 2004, from www.sv-edv.de/ak-swt/PLANK.pdf.

Bearden, Donna; Martin, Kathleen (1998). *My Make Believe Castle – An Epic Adventure in Problem Solving*, in "Learning and Leading with Technology", ISSN 0278-9175, vol. 25, no. 5 , pp. 21-25, International Society for Technology in Education, Eugene, OR, USA. Referenced from the on-line version at http://www.microworlds.com/company/news/bearden.pdf, retrieved on October 3rd, 2004.

Beatty, Barbara (1998). *From Infant Schools to Project Head Start: Doing Historical Research in Early Childhood Education*, in Bernard Spodek, Olivia N. Saracho, and Anthony D. Pellegrini (eds.), "Issues in Early Childhood Educational Research", ISBN 0-8077-3765-8, pp. 1-29, Teachers College Press, New York, NY, USA.

Bederson, Benjamin B.; Boltman, Angela (1998). *Does Animation Help Users Build Mental Maps of Spatial Information?*, in "Proceedings of the 1999 IEEE Symposium on Information Visualization", ISBN 0-7695-0431-0, pp. 28-35, IEEE Computer Society, Washington, DC, USA. Referenced from the on-line version at ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/98-11html/98-11.pdf, retrieved on February 5th, 2004.

Begel, Andrew (1996). *LogoBlocks: A Graphical Programming Language for Interacting with the World*, bachelor's thesis, Epistemology and Learning Group, MIT Media Laboratory, MIT, Cambridge, MA, USA. Referenced from the on-line version at http://web.media.mit.edu/~abegel/begelaup.pdf, retrieved on October 4th, 2004.

Belcher, Jeff (2003). *Glendale Kindergarten 2003/2004 – Robotics*, Web page at http://schools.cbe.ab.ca/b143/SwiftRunner/Robotics.html, Glendale Elementary School, Calgary, Alberta, Canada. Retrieved on August 1st, 2005.

Bennett-Levy, Michael (2001). *1935-1941 Timeline*, Web page found at the address http://www.tvhistory.tv/timeline1.htm, part of the Web site "Television History – The First 75 Years", located at http://www.tvhistory.tv/. Last accessed on May 18th, 2005.

Berg, Gary A. (2003). *The Knowledge Medium: Designing Effective Computer-Based Learning Environments*, ISBN 1-59140-103-8, Information Science Publishing, Idea Group Inc., Hershey, PA, USA.

Bergagna, Miguel (1998). *Uso de Logo a Edades Tempranas*, e-mail message sent on August 15th, 1998, at 15:51:33 -0300, as part of a discussion on the use of Logo with young children. Retrieved from http://mondragon.angeltowns.net/paradiso/EdadesTempranas.html on August 6th, 2005.

Bernhard, Judith K.; Siegel, Linda S. (1994). *Increasing internal locus of control for a disadvantaged group: A computer intervention*, in "Computers in the Schools", ISSN 0738-0569, vol. 11, no. 1, pp. 59-77, The Haworth Press, Inc., Binghamton, NY, USA. Available on-line, DOI 10.1300/J025v11n01_06.

Bers, Marina U.; Ponte, Iris; Juelich, Catherine; Viera, Alison; Schenker, Jonathan (2002). *Teachers as Designers: Integrating Robotics in Early Childhood Education*, in "Information Technology in Childhood Education Annual", ISSN 1522-8185, vol. 2002, no. 1, pp. 123-145, Association for the Advancement of Computers in Education, Norfolk, VA, USA. Referenced from the on-line version at http://www.aace.org/dl/files/ITCE/ITCE20021123.pdf, retrieved on August 1st, 2005.

Berthold, Frank (2004). *Re: Young Toon-Talkers*, e-mail message sent to Leonel Morgado on July 21st, 2004, at 01:23:17 -0400.

Beyer, Landon E. (1997). *William Heard Kilpatrick (1871-1965)*, in "PROSPECTS: quarterly review of comparative education", ISSN 0033-1538, vol. 27, no. 3, pp. 470-485, UNESCO International Bureau of Education, Geneva, Switzerland. Retrieved on July 20th, 2005, from http://www.ibe.unesco.org/International/Publications/Thinkers/ThinkersPdf/kilpatricke.PDF.

Black, Maurice Joseph (2002). *The art of code*, PhD thesis, ISBN 0-493-92866-9, University of Pennsylvania, Philadelphia, PA, USA. Referenced from the on-line preview, retrieved from http://wwwlib.umi.com/dissertations/preview_all/3072974 on August 18th, 2005.

Blackwell, Alan F. (2002). *What is Programming?*, in J. Kuljis, L. Baldwin & R. Scoble (eds.), "Proceedings of the Fourteenth Workshop of the Psychology of Programming Interest Group", pp. 204-218, Brunel University, London, UK. Referenced from the electronic version, retrieved from http://www.ppig.org/papers/14th-blackwell.pdf on July 4th, 2004.

Blackwell, Alan F.; Green, Thomas R. G. (1999). *Does Metaphor Increase Visual Language Usability?*, in "Proceedings of the IEEE Symposium on Visual Languages", ISBN 0-7695-0216-4, IEEE Computer Society, Washington, DC, USA. Referenced from the on-line version, last retrieved on September 2nd 2004, from http://www.cl.cam.ac.uk/~afb21/publications/VL99.pdf.

Blackwell, Alan F.; Hague, Rob (2001). *AutoHAN: An Architecture for Programming the Home*, in "Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)", ISBN 0-7695-0474-4, pp. 150-157, IEEE Computer Society, Washington, DC, USA. Referenced from the on-line version, retrieved on Novermber 11th, 2004, from http://www.cl.cam.ac.uk/users/afb21/publications/HCC01.pdf.

Bolstad, Rachel (2004). *The role and potential of ICT in early childhood education: A review of New Zealand and international literature*, ISBN 0-478-13236-0, Ministry of Education, Wellington, New Zealand. Referenced from the on-line version, ISBN 0-478-13237-9, retrieved from http://www.minedu.govt.nz/web/downloadable/dl10074_v1/ictinecefinal.pdf on July 27th, 2005.

Bonar, Jeffrey; Soloway, Elliot (1983). *Uncovering Principles of Novice Programming*, in "Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages", ISBN 0-89791-090-7, pp. 10-13, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on June 10th, 2005, from http://doi.acm.org/10.1145/567067.567069.

Bork, Alfred (2000). *Four fictional views of the future of learning*, in "The Internet and Higher Education", ISSN 1096-7516, vol. 3, no. 4, 2000-2001, Elsevier Science Publishers Ltd., Essex, UK. Referenced from the electronic version retrieved on March 30th, 2004, from http://www.ics.uci.edu/~bork/fiction4.

Briggs, Jean L. (1998). *Inuit Morality Play: The Emotional Education of a Three-Year-Old*, ISBN 0-300-08064-6, Yale University Press, New Haven, Connecticut, USA.

Bromley, Allan G. (1990). *Difference and Analytical Engines*, in "Computing Before Computers", ISBN 0-8138-0047-1, ch. 2, pp. 59-98, Iowa State University Press, Ames Iowa, USA, 1990. Referenced from the electronic version, retrieved from http://ed-thelen.org/comp-hist/CBC.html on April 29th, 2004.

Bronfenbrenner, Urie (1979). *Ecology of Human Development: Experiments by Nature and Design*, ISBN 0-674-22457-4, Harvard University Press, Cambridge, MA, USA. Referenced from the Brazilian Portuguese translation, "A Ecologia do Desenvolvimento Humano: Experimentos Naturais e Planejados" (1996), Maria Adriana Veríssimo (trad.), ISBN 85-7307-173-7, Editora Artes Médicas Sul, Porto Alegre, Brazil.

Brooks, Xan (2003). *We are all nerds now*, article in the newspaper "The Guardian" on Friday, December 12th, 2003. Guardian Newspapers Limited, London, UK. Referenced from the on-line version at http://film.guardian.co.uk/features/featurepages/0,4120,1104848,00.html, retrieved on August 7th, 2004.

Brookshear, J. Glenn (2003). *Computer Science – An overview*, 7th Edition, ISBN 0-321-18993-0, Pearson Education, Upper Saddle River, New Jersey, USA.

Brown, Curtis (2001). *Syntax, Semantics, and Pragmatics*, notes of the course "Philosophy of Language", PHIL 3333, Philosophy Department, Trinity University, San Antonio, Texas, USA. Available on-line at http://www.trinity.edu/cbrown/language/distinctions.html (last accessed on April 14th, 2005).

Bruckman, Amy Susan (1997). *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids*, doctoral dissertation, Program in Media Arts and Sciences, School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved on September 5th, 2004, from http://www.cc.gatech.edu/fac/Amy.Bruckman/thesis/moose-crossing-entire.pdf.

Bruner, Jerome Seymour (1957). *Going beyond the information given*, in J.S. Bruner, E, Brunswik, L. Festinger, F. Heider, K.F. Muenzinger, C.E. Osgood, & D. Rapaport (eds.), "Contemporary approaches to cognition", pp. 41-69, Harvard University Press, Cambridge, MA, USA.

Bruner, Jerome Seymour (1960). *The Process of Education*, ISBN 0-674-71001-0, Harvard University Press, Cambridge, MA, USA.

Bruner, Jerome Seymour (1966). *Toward a theory of instruction*, ISBN 0-674-89700-5, Harvard University Press, Cambridge, MA, USA.

Bruner, Jerome Seymour (1973). *The relevance of education*, ISBN 0-393-00690-5, W.W. Norton, New York, NY, USA.

Bruner, Jerome Seymour (1996). *The Culture of Education*, ISBN 0674179536, Harvard University Press, Cambridge, MA, USA.

Bruner, Jerome Seymour (2002). Joint interview of Jerome Bruner and Oliver Sacks, by Michael Cole, on March 2002. Retrieved on July 17th, 2005, from http://luria.ucsd.edu/Luria.mov, University of California – San Diego, La Jolla, CA, USA.

Brunner, Helmutt (1981). *L'éducation en ancienne Égypte*, in Gaston Mialaret & Jean Vial (eds.), "Histoire mondiale de l'éducation", ISBN 2-130-36776-3, vol. 1, pp. 65-86, Presses Universitaires de France, Paris, France. Referenced from the Portuguese translation by Evaristo Santos, *A Educação no Antigo Egipto*, in (n.d.) "História Mundial da Educação", ISBN 972-703-067-X, vol. 1, pp. 61-81, RÉS-Editora, Porto, Portugal.

Bryson, Steve; Bulterman, Dick; Catarci, Tiziana; Citrin, Wayne; Cruz, Isabel; Glinert, Ephraim; Grundin, Jonathan; Hollan, Jim; Ioannidis, Yannis; Jacob, Rob; John, Bonnie; Kurlander, David; Myers, Brad; Olsen, Dan; Pausch, Randy; Shieber, Stuart; Shneidermann, Ben; Stasko, John; Strong, Gary; Wittenburg, Kent (1996). *Strategic Directions in Human Computer Interaction*, in "ACM Computing Surveys", ISSN 0360-0300, vol. 28, no. 4, ACM Press. New York, NY, USA. Referenced from the electronic version, retrieved on January 19th, 2001, from http://www.cs.cmu.edu/~bam/nsfworkshop/hcireport.html.

Buescu, Jorge (2003). *Da Falsificação de Euros aos Pequenos Mundos: Novas Crónicas das Fronteiras da Ciência*, ISBN 972-662-898-9, Gradiva, Lisbon, Portugal.

Bulas-Cruz, José; Morgado, Leonel; Barbosa, Luís; Reis, Arsénio; Barroso, João; Melo-Pinto, Pedro; Henriques, Pedro (1998). *The specification of geometrical dynamic behaviour in Web page design*, in Nikos E. Mastorakis (ed.), "Recent Advances in Information Science and Technology", ISBN 981-02-3657-3, pp.129-135, World Scientific, Athens, Greece.

Bulas-Cruz, José; Morgado, Leonel; Melo-Pinto, Pedro; Abreu, Mila; Lobo, Helena; Guedes, Mário; Santos, Arlindo; Borges, Jorge; Bicho, Joel; Barroso, João; Reis, Arsénio; Proença, Alberto (1999). *Beyond traditional Web page designs - a communication language between designers and Web page developers*, in "New Techniques for Old times - CAA 98 - Computer Applications and Quantitative Methods in Archaeology - Proceedings of the 26th Conference", ISBN 0-86054-961-5, BAR International Series 757, pp. 367-368, Archaeopress, Oxford, England.

Burns, Alan; Davies, Geoff (1993). *Concurrent Programming*, ISBN 0-201-54417-2, Addison-Wesley Publishers Ltd., Pearson Education Ltd., Harlow, Essex, UK.

Burton-Wilcock, Donna; Owen, Martin; Facer, Keri (2003). *Tableaux*, on-line article at http://www.nestafuturelab.org/showcase/mediastage/Tableaux.pdf, NESTA Futurelab, Bristol, UK. Retrieved on November 15th, 2004.

Bush, Vannevar (1945). *As We May Think*, in The Atlantic Monthly, July 1945. Referenced from the electronic version at http://arti.vub.ac.be/cursus/2001-2002/ai2/material/bush.pdf, retrieved on March 24th, 2003.

Buysse, Virgina; Sparkman, Karen L.; Wesley, Patricia W. (2003). *Communities of Practice: Connecting What We Know With What We Do*, in "Exceptional Children", ISSN 0014-4029, vol. 69, no. 3, pp. 263-277, The Council for Exceptional Children, Arlington, VA, USA. Referenced from the on-line version, retrieved on July 7th, 2005, from http://journals.sped.org/EC/Archive_Articles/VOLUME69NUMBER3Spring2003_EC_Article-1.pdf.

Byron-King, Augusta Ada (1843). *Sketch of the Analytical Engine invented by Charles Babbage, Esq. By L. F. MENABREA, of Turin, Officer of the Military Engineers*, in Richard Taylor (ed.), "Scientific Memoirs", no. 3, pp. 666-731, translation from the French (Menabrea, 1842). Referenced from the electronic version in *Classics in the History of Psychology*, ISSN 1492-3173, York University, Toronto, Canada. Referenced from the on-line version, retrieved on April 28th, 2004, from http://psychclassics.yorku.ca/Lovelace/menabrea.htm.

Cabasino, Simone; Todesco, Gian Marco; Paolucci, Pier Stanislao (2001). *Zz: The Tao Engine*, Web-based manual at http://cvs.sourceforge.net/viewcvs.py/openzz/openzz/doc/zzdoc.html, retrieved on August 24th, 2004.

Callahan, Paul (2000). *What is the Game of Life?*, Web page at http://www.math.com/students/wonders/life/life.html, Math.com, LLC, Encore Software, Inc., Gardena, California, USA. Retrieved on August 17th, 2004.

Camus, William (1968). *Lorsque vinrent les Visages-Pâles* (record number at the Bibliothéque Natianale de France: FRBNF32939605), Presses de la Cité, Paris, France. Referenced from the Portuguese translation by Helena Ramos, "Vêm aí os rostos pálidos", Plátano Editora, Lisbon, Portugal, 1978.

Cartier, Michel (n.d.). *A Educação na China Antiga e Medieval*, in "História Mundial da Educação", ISBN 972-703-067-X, vol. 1, pp. 83-93, RÉS-Editora, Porto, Portugal. This volume is a translation by Evaristo Santos of Gaston Mialaret & Jean Vial (eds.), 1981, "Histoire mondiale de l'éducation", ISBN 2-130-36776-3, vol. 1, Presses Universitaires de France, Paris, France.

Cassidy, William (2002). *Pinball Construction Set*, online article at http://archive.gamespy.com/halloffame/september02/pcs/, gamespy.com, GameSpy Industries, IGN Entertainment, Inc., Palatine, IL, USA. Retrieved on October 1st, 2004.

Cassidy, William (2004). *The Odyssey² Homepage*. Electronic resource, retrieved from www.classicgaming.com/o2home/ on March 5th, 2004.

Castro-Caldas, Alexandre (2003). *Como encontrar áreas de interesse para estudar o cérebro analfabeto*, in Revista Psychologica 34, 2003, Faculdade de Psicologia e Ciências da Educação da Universidade de Coimbra, Coimbra, Portugal. Also referenced is version in English (Castro-Caldas, 2004), which includes updated research information.

Castro-Caldas, Alexandre (2004). *Targeting regions of interest for the study of the illiterate brain*, in International Journal of Psychology, February 2004, vol. 39, no. 1, pp. 5-17, ISSN 0020-7594, Psychology Press, Taylor & Francis Group, London, UK.

Cauthen, Lavetia (1990). *Mathematics/Computer Integrated Curriculum, K-1*, ED323113, ERIC – Education Resources Information Center, Computer Sciences Corporation, Lanham, MD, USA.

Cawley, Kevin (2004). *Latin Dictionary and Grammar Aid*, University of Notre Dame, Notre Dame, Indiana, USA. Electronic resource, retrieved on April 30th, 2004, from http://www.nd.edu/~archives/latgramm.htm.

Ceruzzi, Paul E. (1983). *Reckoners – The Prehistory Of The Digital Computer, From Relays To The Stored Program Concept, 1935-1945*, ISBN 0-313-23382-9, Greenwood Press, Westport, Connecticut, USA. Referenced from the electronic version, retrieved from http://ed-thelen.org/comp-hist/Reckoners.html on April 29th, 2004.

Ceruzzi, Paul E. (1990). *Relay Calculators*, in "Computing Before Computers", ch. 6, pp. 200-222, ISBN 0-8138-0047-1, Iowa State University Press, Ames, Iowa, USA. Referenced from the electronic version at http://ed-thelen.org/comp-hist/CBC.html, retrieved on April 29th, 2004.

Chakraborty, Anit; Graebner, Randy; Stocky, Tom (1999). *Logo – A project history*, final paper of student project for course 6.933J/STS.420J, "The Structure of Engineering Revolutions", lectured by David A. Mindell, MIT, Cambridge, MA, USA. Retrieved from http://web.mit.edu/6.933/www/LogoFinalPaper.pdf on September 9th, 2004.

Chandra, Shyamal Suhana; Chandra, Kailash (2005). *A comparison of Java and C#*, in "Journal of Computing Sciences in Colleges", vol. 20, no. 3, pp. 238-254, Consortium for Computing Sciences in Colleges, Shelbyville, IN, USA. Referenced from the on-line version, retrieved on August 17th, 2005, from http://delivery.acm.org/10.1145/1050000/1040228/p238-chandra.pdf.

Chemstations (no date). *ChemCAD Suite – Program Features*, on-line document, Chemstations, Inc., Houston, Texas, USA. Retrieved on August 18th, 2005, from http://www.chemstations.net/documents/ccsuite.pdf.

Cheng, Andrew Chun-Ho (1998). *A graphical programming interface for a children's constructionist learning environment*, thesis (M.Eng.), Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Chioccariello, Augusto (1997). *Multimedia Messages by Five-year-old Children*, in Masoud Yazdani & Linda Baggott (eds.), "CAL 97", ISBN 85068-188X, on-line proceedings at http://www.media.uwe.ac.uk/masoud/cal-97/%23proc%5e5f, Digital Media Research Centre, University of the West of England, Bristol, UK. Retrieved on August 10th, 2005, from http://www.media.uwe.ac.uk/masoud/cal-97/posters/chiccari.htm.

Chmiel, Ryan; Loui, Michael C. (2004). *Debugging: from novice to expert*, in "Proceedings of the 35th SIGCSE technical symposium on Computer science education", ISSN 0097-8418, pp.

17-21, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on June 11th, 2005, from http://doi.acm.org/10.1145/971300.971310.

Chomsky, Noam (1968). *Linguistic Contributions to the Study of Mind*, excerpted from Chomsky, Noam (1968), "Language and Mind", ISBN 0-15549-257-8, Harcourt Brace Jovanovich, San Diego, CA, USA. Referenced from the on-line version, retrieved on February 9th, 2005, from http://www.chomsky.info/books/mind01.htm.

CIELJ, Centre International d'Etudes en Littérature de Jeunesse (no date). *William Camus*, Web page at http://www.ricochet-jeunes.org/auteur.asp?name=Camus&surname=William, retrieved on September 3rd, 2004.

Clements, Douglas H. (1983). *Supporting Young Children's Logo Programming*, in "The Computing Teacher", ISSN 0278-9175, vol. 11, no. 5, pp. 24-30, International Society for Technology in Education, Eugene, OR, USA.

Clements, Douglas H. (1990). *Turtle soup: A beginning look at Logo research*, in Logo Exchange, ISSN 0888-6970, vol. 9, issue 2, pp. 32-35, ISTE, Eugene OR, USA.

Clements, Douglas H. (1994). *The uniqueness of the computer as a learning tool: Insights from research and practice*, in June L. Wright & Daniel D. Shade (eds.), "Young children: Active learners in a technological age", pp. 31-50, NAEYC, Washington, DC, USA.

Clements, Douglas H. (1999a). *'Concrete' Manipulatives, Concrete Ideas*, in "Contemporary Issues in Early Childhood", ISSN 1463-9491, vol. 1, no. 1, Symposium Journals, Oxford, UK. Referenced from the on-line version, retrieved on September 1st, 2004, from www.wwwords.co.uk/pdf/viewpdf.asp?j=ciec&vol=1&issue=1&year=2000&article=clements.

Clements, Douglas H. (1999b). *The Future of Educational Computing Research: The Case of Computer Programming*, in "Information Technology in Childhood Education Annual", ISSN 1522-8185, vol. 1999, no. 1, pp. 147-179, Association for the Advancement of Computers in Education, Norfolk, VA, USA. Referenced from the on-line version, retrieved on August 5th, 2005. from http://investigations.terc.edu/relevant/pdf/EducationalComputing.pdf.

Clements, Douglas H. (1999c). *Effective use of computers with young children*, in "Mathematics in the Early Years", J. V. Copley (ed.), ISBN 0-87353-469-7, pp. 119-128, National Council of Teachers of Mathematics, Reston, VA, USA. Referenced from the on-line version, retrieved on August 5th, 2005, from http://investigations.terc.edu/relevant/EffectiveUse.html.

Clements, Douglas H. (2002). *Computers in Early Childhood Mathematics*, in "Contemporary Issues in Early Childhood", vol. 3, no. 2, pp. 160-181. Referenced from the on-line version, retrieved from on June 28th, 2004.

Clements, Douglas H.; Battista, Michael T. (2001). *Logo and geometry*, ISBN 0-87353-509-X, National Council of Teachers of Mathematics, Reston, VA, USA.

Clements, Douglas H.; Meredith, Julie S. (1992). *Research on Logo: Effects and Efficacy*, Logo Foundation, New York, USA. Electronic resource, retrieved on March 19th, 2004, from http://el.media.mit.edu/logo-foundation/pubs/papers/research_logo.html.

Clements, Douglas H.; Nastasi, Bonnie K. (1992). *Computers and early childhood education*, in M. Gettinger, S.N. Elliott, & T.R. Kratochwill (eds.), "Advances in school psychology: Preschool and early childhood treatment directions", ISBN 0805807578, pp. 187-246, Lawrence Erlbaum, Hillsdale, NJ, USA.

Clements, Douglas H.; Nastasi, Bonnie K. (1999). *Metacognition, learning, and educational computer environments*, in "Information Technology in Childhood Education Annual", ISSN 1522-8185, vol. 1999, no. 1, pp. 3-36, Association for the Advancement of Computers in Education, Norfolk, VA, USA. Referenced from the on-line version, retrieved on August 6th, 2005, from http://investigations.terc.edu/relevant/Metacognition.cfm.

Clements, Douglas H.; Nastasi, Bonnie K.; Swaminathan, Sudha (1993). *Young Children and Computers: Crossroads and Directions From Research*. Young Children, 48 (2), pp. 56-64, ISSN 0044-0728, National Association for the Education of Young Children, Washington DC, USA.

Clements, Douglas H.; Sarama, Julie (2002). *The Role of Technology in Early Childhood Learning*, in "Teaching Children Mathematics", ISSN 1073-5836, no. 8, pp. 340-343, National Council of Teachers of Mathematics, Reston, VA, USA. Referenced from the on-line version at http://my.nctm.org/eresources/view_media.asp?article_id=1897, retrieved on February 27th, 2005.

Clements, Douglas H.; Sarama, Julie (2003). *Strip Mining for Gold: Research and Policy in Educational Technology—A Response to "Fool's Gold"*, in "Educational Technology Review", ISSN 1551-3696, vol. 11, issue 1, pp. 7-69, Association for the Advancement of Computing in Education, Norfolk, VA, USA. Referenced from the on-line version, retrieved on August 2nd, 2005, from http://dl.aace.org/12683

Cnotinfor (2005). *Floresta Mágica – A quem se destina*, Web page at the company Web site, Cnotinfor, Coimbra, Portugal. Retrieved from http://educacao.cnotinfor.pt/index.php?pag=123 on August 2nd, 2005.

Cockburn, Andy; Bryant, Andrew (1997). *Leogo: An Equal Opportunity User Interface for Programming*, in "Journal of Visual Languages and Computing", vol. 8, issue 5-6, pp. 601-619, ISSN 1045-926X, Academic Press, Cambridge, UK. Referenced from the on-line version, retrieved on October 25th, 2004, from http://usa.cosc.canterbury.ac.nz/andrew.cockburn/papers/jour_leo.pdf.

Cockburn, Andy; Bryant, Andrew (1998). *Cleogo: Collaborative and Multi-Metaphor Programming for Kids*, in "Proceedings of the Third Asian Pacific Computer and Human Interaction", ISBN 0-8186-8347-3, pp. 189-, IEEE Computer Society, Washington, DC, USA. Referenced from the on-line version, retrieved on October 24th, 2002, from http://www.cosc.canterbury.ac.nz/~andy/papers/cleogo.pdf.

CogniToy (2001). *MindRover – MindRover: The Europa Project*, Web page at http://www.mindrover.com/mindrover/mindrover.htm, maintained by CogniToy, LLC, Acton, MA, USA. Retrieved on November 15th, 2004.

Cole, George (2004). *A new stage of learning*, in "The Times Educational Supplement", edition of October 22nd, 2004, TSL Education Ltd., London, UK. Referenced from the on-line fac-simile version, retrieved from http://www.kar2ouche.com/mediastage/tesreview22-10-04.pdf on November 15th, 2004.

Colmerauer; Alain; Roussel, Philippe (1996). *The birth of Prolog*, in Thomas J. Bergin Jr. & Richard G. Gibson Jr. (eds.), "History of programming languages II", ISBN 0201895021, pp. 331-367, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/234286.1057820.

Coniam, David (1992). *Literacy for the next generation: Writing without handwriting*, in "Ejournal", ISSN 1054-1055, vol. 2, issue 2, Faculty of Communication and Culture, University of Calgary, Calgary, Alberta, Canada. Retrieved on August 9th, 2005, from http://www.ucalgary.ca/ejournal/archive/rachel/v2n2/article.html.

Conway, Matthew J. (1997). *Alice: Easy-to-Learn 3D Scripting for Novices*, PhD dissertation, School of Engineering and Applied Science, University of Virginia, Charlottesville, Virginia, USA. Referenced from the on-line version, retrieved on November 24th, 2004, from http://www.alice.org/advancedtutorial/ConwayDissertation.PDF.

Copeland, Jack (2000). *A Brief History of Computing*, Web page at the site "The Turing Archive for the History of Computing", The Turing Project, University of Canterbury, Christchurch, New Zealand. Retrieved on August 22nd, 2005, from http://www.alanturing.net/turing_archive/pages/Reference%20Articles/BriefHistofComp.html.

Correia, Rigoberto (1990). *A Informática ao Serviço da Aprendizagem*, in "BICA – Boletim informativo do centro de recursos – Interactividade, Comunicação, Aprendizagem", vol. 90/91, no. 1, pp. 11-15, Cnotinfor, Coimbra, Portugal.

Correia, Secundino; Andrade, Manuela (2001). *Floresta Mágica*, product manual, ISBN 972-8336-07-1, Cnotinfor, Coimbra, Portugal. Referenced from the on-line version at http://educacao.cnotinfor.pt/manuais/index.php?manual=floresta, retrieved on October 20th, 2004. At the URL address http://educacao.cnotinfor.pt/manuais/index.php?manual=floresta there is an English translation available.

Correia, Tiago (2005). *Floresta Mágica Reports*, personal e-mail message sent to Leonel Morgado on August 1st, 2005, at 16:30:39 +0100 (full text available on Annex II).

Coutinho, Clara Pereira (2005). *Os "Conteúdos" da Tecnologia Educativa nos Cursos de Formação de Professores em Portugal: Estudo Analítico em Instituições de Ensino Superior Público*, in "Challenges 2005" (conference proceedings on CD-ROM), Paulo Dias & Cândido Varela de Freitas (eds.), ISBN 972-8746-13-05, pp. 561-573, Centro de Competências Nónio Séc. XXI, Universidade do Minho, Braga, Portugal.

Crato, Nuno (2005). *Elogio da Divulgação*, in "Única" magazine, part of the weekly newspaper "Expresso", no. 1711, August 13th, 2005, Sojornal, Lisbon, Portugal.

Crijns, Harry (2003). *The communication between Home and Building applications on One-Single-Standard*, in "Journal", March 2003, no. 1, pp. 5-6, Konnex Association, Brussels, Belgium. Referenced from the on-line version at http://www.konnex.org/DOWNLOAD/18_JOURN.PDF, retrieved on August 18th, 2005.

Crouch, John (1991). *Bronson Alcott's Experiment In Practical Transcendentalism*, Web page at http://adams.patriot.net/~crouch/artj/bron.html, Crouch & Crouch, Arlington, Virgina, USA. Last accessed on May 5th, 2005.

Cruz, Frank da (2004). *The IBM 650 Magnetic Drum Calculator*, in the Web site "Columbia University Computing History", at http://www.columbia.edu/acis/history/650.html, retrieved on July 2nd, 2004.

Cruz, Maria Gabriel Moreno Bulas; Cristóvão, Rosa; Ferreira, Paula; Lopes, Fernando; Costa, Gisela; Claro, Ana Rita; Soares, Graça; Águeda, Paula; Moreira, Idalina (in the press). *Informática em Contextos de Educação de Infância – ICEI. Funcionamento e práticas ao nível da introdução das TIC em Jardins de Infância da Região de Trás-os-Montes e Alto Douro*, paper presented at the 3rd International Education Congress, "O Mundo da Criança", at UTAD, Vila Real, Portugal, 2004.

CSTA, Computer Science Teachers Association (2003). *A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee*, ISBN 1-58113-837-7, Computer Science Teachers Association, New York, NY, USA. Referenced from the on-line version, retrieved from http://www.acm.org/education/k12/k12final1022.pdf on June 24th, 2005.

Cunningham, Rich (1997). *Caution!!! Mad Scientist At Work! – A Review of Widget Workshop: The Mad Scientist's Laboratory*, on-line review retrieved on November 15th, 2004, from http://www.worldvillage.com/wv/school/html/reviews/widget.htm, InfoMedia, Inc., Birmingham, AL, USA.

Cypher, Allen (1993). *Bringing Programming to End Users*, in Allen Cypher (ed.), "Watch What I Do: Programming by Demonstration", ISBN 0-262-03213-9, pp. 1-11, MIT Press, Cambridge, Massachusetts, USA. (Cited from the second printing, 1994.)

Daffron, Susan C. (1999). *Speed Up with Styles*, in Computor Companion, Septermber/October 1999 issue, Logical Expressions, Inc. Referenced from the on-line version at http://www.computorcompanion.com/LPMArticle.asp?ID=104, retrieved on June 28th, 2004.

DAM, Digital Art Museum (no date). *Roman Verostko: Homage to Norbert Wiener – A Decision Machine Suite, Interactive Electronic Sculpture, 1982-1995*, Web page at the Web site of the DAM, Berlin, Germany. Retrieved from http://www.dam.org/verostko/wiener.htm on August 18th, 2005.

Davis, Bernadette Caruso & Shade, Daniel D. (1994). *Integrate, Don't Isolate! — Computers in the Early Childhood Curriculum*, ERIC Digest no. EDO-PS-94-17, U.S. Department of Education, Washington DC, USA. Referenced from the on-line version, retrieved on February 27th, 2005, from http://ceep.crc.uiuc.edu/eecearchive/digests/1994/shade94.html.

Dear, Brian L. (2004). *What's New at PLATOPEOPLE.COM*, Web page retrieved from http://www.platopeople.com/whatsnew.html on July 30th, 2004.

DeBonte, Austina M. (1998). *Pet Park: A Virtual Learning World for Kids*, Master Thesis, Department of Electrical Engineering and Computer Science, Massachussets Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version at http://llk.media.mit.edu/papers/archive/deBonte-MEng/, retrieved on October 6th, 2004.

Degelman, Douglas; Brokaw, Ellen J.; Free, John U. (1986). *Effects of Logo Experience and Grade on Concept Learning and Creativity*, on-line report at www.vanguard.edu/faculty/ddegelman /logo.pdf, Vanguard University of Southern California, Costa Mesa, CA, USA. Retrieved on March 7th, 2003.

Degelman, Douglas; Free, John U.; Scarlato, Michelle; Blackburn, Janice M.; Golden, Thomas (1986). *Concept learning in preschool children: Effects of a short-term LOGO experience*, in "Journal of Educational Computing Research", vol. 2, no. 2, ISSN 0735-6331, Baywood Publishing Company, Amityville, New York, USA.

Dewey, John (1895). *Plan of organization of the university primary school*, in "Early Works of John Dewey" (1972), vol. 5, pp. 224-243, Southern Illinois University Press, Carbondale, IL, USA.

Dewey, John (1897). *My Pedagogic Creed*, in "The School Journal", Volume LIV, Number 3 (January 16th, 1897), pp. 77-80, New York, NY, USA. Referenced from the on-line version, retrieved from http://www.ittheory.com/dewey2.htm on May 25th, 2005, and from the Portuguese translation by Maria Adelaide Pinto Correia (1989), "O meu Credo Pedagógico", in "Cadernos de Educação de Infância", no. 9, Associação dos Profissionais de Educação de Infância, Lisbon, Portugal.

Dick, Bob (1999). *What is action research?*, Web page available on-line at http://www.scu.edu.au/schools/gcm/ar/whatisar.html, part of "Action Research Web Site", Graduate College of Management, Southern Cross University, Lismore, NSW, Australia. Retrieved on June 2nd, 2005.

Dijkstra, Edsger W. (1965). *Cooperating sequential processes*, Report EWD 123, Technische Universiteit, Eindhoven, The Netherlands. Referenced from "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", pp. 65-138, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Dijkstra, Edsger W. (1968). *Go To Statement Considered Harmful*, Communications of the ACM, ISSN 0001-0782, volume 11, Issue 3 (March 1968), pp. 147-148, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/362929.362947, retrieved on September 9th, 2004.

Dijkstra, Edsger W. (1975). *A synthesis emerging?*, in E. W. Dijkstra, "Selected Writings on Computing: A Personal Perspective" (1982), ISBN 0387906525, pp. 147-160, Springer-Verlag, New York, NY, USA. Referenced from "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", pp. 65-138, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Dijkstra, Edsger W. (1989). *On the Cruelty of Really Teaching Computing Science*, in "A Debate on Teaching Computer Science", Communications of the ACM, ISSN 0001-0782, Volume 32, no. 12, December 1989, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 27th, 2004, from http://doi.acm.org/10.1145/76380.76381.

Diot, Christophe (1994). *Reliability in Multicast Services and Protocols ; A Survey.*, in "Local and Metropolitan Communication Systems: Proceedings of the Third International Conference on Local and Metropolitan Communication Systems", Kyoto, Japan, December 1994, Toshiharu Hasegawa, Guy Pujolle, Hideaki Takagi, Yutaka Takahashi, eds., ISBN 0412711702, Chapman & Hall, London, UK. Referenced from the on-line version, retrieved on February 2nd, 2004 from http://cambridgeweb.cambridge.intel-research.net/people/pub/cdiot/xcast-kyoto94.ps.gz.

diSessa, Andrea A. (2000). *Changing Minds: computers, learning and literacy*, ISBN 0-262-04180-4, MIT Press, Cambridge, Massachusetts, USA.

diSessa, Andrea A.; Abelson, Harold (1986). *Boxer: a reconstructible computational medium*, in "Communications of the ACM", vol. 29, issue 9, September 1986, pp. 859-868, ISSN 0001-0782, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on October 20th, 2004, from http://doi.acm.org/10.1145/6592.6595.

Donnachie, Ian (2003). *Education in Robert Owen's New Society: The New Lanark Institute and Schools*, in "the encyclopedia of informal education", Web page retrieved on May 4th, 2005, from www.infed.org/thinkers/et-owen.htm, last updated on January 30th, 2005.

Dozen, Patty (1998). *PLATO/NOVANET History*, e-mail message sent from the e-mail address pdozen@sunny.vcccd.cc.ca.us, on March 12th, 1998, at 12:01:15 -0800 to the mailing list "Open Forum for Learning Assistance Professionals", at e-mail address <lrnasst@listserv.arizona.edu>. Referenced from the on-line archived version, retrieved from http://www.lists.ufl.edu/cgi-bin/wa?A2=ind9803&L=lrnasst-l&F=&S=&P=16965 on July 30th, 2004.

Druin, Allison (1996). *CD-ROM Edutainment*, in "Designing Multimedia Environments for Children", ch. 2, pp. 62-93 Allison Druin, Cynthia Solomon, eds., ISBN 0-471-11688-2, John Wiley & Sons, Inc., New York, NY, USA.

Druin, Allison (1999a). *Cooperative Inquiry: Developing New Technologies for Children with Children*, in "Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit", ISBN 0-201-48559-1, pp. 592-599, ACM Press, New York, NY, USA. Referenced from the online version, retrieved from http://doi.acm.org/10.1145/302979.303166 on June 11th, 2004.

Druin, Allison (1999b). *The Role of Children in the Design of New Technology*, HCIL Technical Report no. HCIL-99-23, also published in "Behaviour and Information Technology", ISSN 0144-929X, vol. 21, no. 1 (2002), pp. 1-25, Taylor & Francis Group Ltd., Abingdon, Oxfordshire, UK. Retrieved on June 11th, 2004, from ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/99-23html/99-23.pdf.

Dunhio, Fergus (1991). *The Mind/Body Problem and its Solution*, (unpublished thesis). Rensselaer Polytechnic Institute, Troy, New York, USA. Referenced from the electronic version, retrieved on April 7th, 2004, from http://www.ling.rochester.edu/~duniho/MS-Thesis/Contents.html.

Dwyer, Thomas A. (1971), *On the importance of complexity in supportive systems for educational computing*, Interface v. 5, no. 3: 99-105. Referenced from the on-line version of ACM SIGCUE Outlook, ISSN 0163-5735, ACM Press, New York, NY, USA, retrieved from http://doi.acm.org/10.1145/965836.965838 on August 4th, 2004.

Early Connections (no date). *Research Review of Literature about the Appropriate uses of Computers and Technology with Young Children*, Web page (last accessed on June 1st, 2005) at http://www.netc.org/earlyconnections/research.html, part of the site "Early Connections:

Technology in Early Childhood Education", edited by the Northwest Regional Educational Laboratory (Portland, OR, USA) and the Northwest Educational Technology Consortium (Portland, OR, USA).

Edmark Corporation (1995). *Thinking Things Collection 3*, ISBN 1569262047, Edmark Corporation, Riverdeep Interactive Learning Limited, San Francisco, CA, USA.

Edwards, Carolyn Pope (2002). *Three Approaches from Europe: Waldorf, Montessori, and Reggio Emilia*, in "Early Childhood Research & Practice", ISSN 1524-5039, vol. 4, no. 1, Early Childhood and Parenting (ECAP) Collaborative, Children's Research Center, University of Illinois at Urbana-Champaign, Champaign, IL, USA. Referenced from the on-line version, retrieved on July 23rd, 2005, from http://ecrp.uiuc.edu/v4n1/edwards.html.

Eljaala, Eeva (2002). *Programming without coding: Children as programmers*, Master's thesis, LiTH-IDA-Ex-02/72, June 7th, 2002, Department of Computer and Information Science, University of Linköping, Linköping, Sweden.

Epstein, Ann S. (2003). *All About High/Scope*, in "High/Scope ReSource, A Magazine for Educators", ISSN 0897-2007, Spring 2003, pp. 5-7, High/Scope Press, Ypsilanti, MI, USA. Referenced from the version on-line, retrieved on July 22nd, 2005 from http://www.highscope.org/NewsandInformation/AllAbout.pdf.

Erikson, Erik H. (1959). *Identity and the life cycle; selected papers, with a historical introduction by David Rapaport*, International Universities Press, New York, NY, USA. Referenced from the paperback reissue edition "Identity and the Life Cycle" (1994), ISBN 0393311325, W. W. Norton & Company, New York, NY, USA.

Erwin, Ben; Cyr, Martha; Rogers, Chris (1999). *LEGO Engineer and RoboLab: Teaching Engineering with LabVIEW from Kindergarten to Graduate School*, in "International Journal of Engineering Education", ISSN 0949-149X, vol. 16, no. 3, Dublin Institute of Technology, Dublin, Ireland. Referenced from the on-line version at www.ni.com/pdf/academic/us/journals/ijee_05.pdf, retrieved on August 1st, 2005.

Estes, Richard Despard (1991). *The Behavior Guide to African Mammals: Including Hoofed Mammals, Carnivores, Primates*. Referenced from the 1992 paperback edition, p. 551, ISBN 0520080858, University of California Press, Berkeley, California, USA.

Estevez, Puni (1998). *Uso de Logo a Edades Tempranas*, e-mail message sent on March 10th, 1998, at 22:19:01 -0400, as part of a discussion on the use of Logo with young children. Retrieved from http://mondragon.angeltowns.net/paradiso/EdadesTempranas.html on August 6th, 2005.

Estrela, Albano; Ferreira, Júlia (eds.) (2001). *Investigação em Educação: Métodos e Técnicas*, ISBN 972-8036-30-2, EDUCA, Faculdade de Psicologia e Ciências da Educação, University of Lisbon, Portugal.

Farley, T. (2005). *The first automatic radiotelephone service*, in "Mobiles Phones – A History", Web page at http://www.galaxyphones.co.uk/mobile_phones_history06.asp, Galaxy Phones, Wigan, Lancashire, UK. Last accessed on May 18th, 2005.

Feinberg, David S. (2004). *Broadcast-Based Communication in a Programming Environment for Novices*, Master's thesis, MIT, Cambridge, Massachusetts, USA. Referenced from the on-line version, retrieved on August 5th, 2005, from http://llk.media.mit.edu/papers/archive/davefthesis.pdf.

Felice, Jorge (1998). *Uso de Logo a Edades Tempranas*, e-mail message sent on November 6th, 1998, at 09:56:34 -0300, as part of a discussion on the use of Logo with young children. Retrieved from http://mondragon.angeltowns.net/paradiso/EdadesTempranas.html on August 6th, 2005.

Fenton, Jay; Beck, Kent (1989). *Playground: an object-oriented simulation system with agent rules for children of all ages*, in "Conference proceedings on Object-oriented programming

systems, languages and applications", ISSN 0362-1340, pp. 123-137, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/74877.74891 on October 19th, 2004.

Fernaeus, Ylva; Tholander, Jakob (2003). *Collaborative Computation on the Floor*, in "Designing for Change in Networked Learning Environments: Proceedings of the International Conference on Computer Support for Collaborative Learning 2003", ISBN 1-4020-1383-3, edited by Wasson, Barbara; Ludvigsen, Sten; and Hoppe, Ulrich, Kluwer Academic Publishers, Springer, Dordrecht, The Netherlands. Referenced from the on-line version, retrieved on October 7th, 2003, from http://www.dsv.su.se/research/kids/pdf/floor%20programmingFernaeusTholander.pdf.

Feurzeig, Wallace (1984). *The Logo Lineage*, in "Digital Deli – The Comprehensive, User-Lovable Menu of Computer Lore, Culture, Lifestyles and Fancy by The Lunch Group & Guests", Steve Ditlea, ed., ISBN 0894805916, Workman Publishing Company, Inc., New York, NY, USA. Referenced from the on-line version, retrieved from http://www.atariarchives.org/deli/logo.php on August 31st, 2004.

Feurzeig, Wallace (1996). *Function Machines*, in "Communications of the ACM", vol. 39, issue 8, ISSN 0001-0782, pp. 88-90, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on October 20th, 2004, from http://doi.acm.org/10.1145/232014.232038.

Feurzeig, Wallace (1999). *Visualizing Algorithms*, in "Proceedings of the Seventh European Logo Conference Eurologo '99, Sofia, Bulgaria, 22-25 August 1999", edited by Roumen Nikolov, Evgenia Sendova, Iliana Nikolova and Ivan Derzhanski, pp. 40-49, ISBN 954-9582-03-5, Virtech Ltd., Sofia, Bulgaria.

Figueiredo, António Dias de (2003). *Métodos de Investigação Científica, Acetatos*, lecture notes available at https://www.dei.uc.pt/weboncampus/class/getmaterial.do?idclass=66&idyear=1, Department of Informatics Engineering, Faculty of Sciences and Technology, University of Coimbra, Coimbra, Portugal. Retrieved on January 21st, 2004.

FIMEM, Fédération Internationale des Mouvements d'Ecole Moderne (no date). *Mouvements Freinet dans le monde*, Web page at http://freinet.org/fimem/mvements.htm, retrieved on July 21st, 2005.

Firaxis Games (no date). *The Civilization III Series*, Web page, Firaxis Games, Hunt Valley, MD, USA. Retrieved on July 23rd, 2004 from http://www.firaxis.com/games_civ3.cfm.

Fishwick, Paul A. (2002). *Aesthetic Programming*, in "Leonardo", ISSN 0024-094X, vol. 35, no. 4, pp. 383-390, Leonardo/ISAST, San Francisco, CA, USA. Referenced from the on-line version, retrieved on August 18th, 2005, from http://www.cise.ufl.edu/%7Efishwick/tr/00/leo.pdf.

Fodor, Jerry A. (1981). *The Mind-Body Problem*, Scientific American, ISSN 0036-8733, vol. 244, issue 1, pp. 124-132, Scientific American, Inc., New York, NY, USA.

Fonseca, Clotilde (1999). *The Computer in Costa Rica: A New Door to Educational and Social Opportunities*, in "Logo Philosophy and Implementation", ISBN 2-89371-494-3, pp. 2-21, Logo Computer Systems Inc., Highgate Springs, Vermont, USA. Electronic book retrieved from http://www.microworlds.com/company/philosophy.pdf on March 30th, 2004.

Foote, Thomas (no date). *What is Robot Odyssey?*, Web page retrieved from http://mywebpages.comcast.net/tomfoote3/DQ/id16.htm on November 15th, 2004.

Formosinho, Júlia (1998), *A Contextualização do Modelo Curricular High-Scope no Âmbito do Projecto Infância*, in "Modelos Curriculares para a Educação de Infância", Júlia Oliveira Formosinho (ed.), 2nd edition (1998), ISBN 972-0-34451-2, pp. 51-92, Porto Editora, Oporto, Portugal.

Formosinho, Júlia (2001). *Os modelos pedagógicos para a educação de infância e o desafio da diversidade pedagófica*, in "Pensar o currículo em educação de infância: Livro de Actas / VII Encontro Nacional da APEI", APEI, Lisbon, Portugal.

Fosnot, Catherine Twomey (1996). *Constructivism: A Psychological Theory of Learning*, in "Constructivism – Theory, Perspectives and Practice", pp. 8-33, ISBN 0807734888, Teachers College Press, New York, USA. Referenced from the Portuguese translation by Maria João Batalha Reis, *Construtivismo: Uma Teoria Psicológica da Aprendizagem*, in "Construtivismo e Educação – Teoria, Perspectivas e Prática", pp. 23-58, ISBN 972-771-098-0, Instituto Piaget, Lisbon, Portugal, 1999.

Frei, Phil; Su, Victor; Mikhak, Bakhtiar; Ishii, Hiroshi (2000). *curlybot: Designing a New Class of Computational Toys*, in "Proceedings of the SIGCHI conference on Human factors in computing systems", ISBN 1-58113-216-6, pp. 129-136, ACM Press, New York, NY, USA.

Freire, Paulo (1997). *Pedagogia da Autonomia*, ISBN 85-219-0243-3, Paz e Terra, São Paulo, Brazil (referenced from the 21st edition, 2002). There is an English-language version: *Pedagogy of Freedom: Ethics, Democracy, and Civic Courage*, ISBN 0847690474, Rowman & Littlefield Publishing, Lanham, Maryland, USA, 2000.

Fröbel, Friedrich (1826). *Die Menschenerziehung*, Wienbrach, Keilhau/Leipzig, Principality of Schwarzburg-Rudolstadt (presently part of Thuringia, Germany). English version: *On the Education of Man*, translated by J. Jarvis, A. Lovell and Co., New York, NY, USA, 1885. Referenced from the on-line excerpts at http://www.froebelweb.org/web7000.html.

Gal, Roger (1966). *Significado Histórico da Educação Nova*, in "Educação Nova e Mundo Moderno", Gaston Mialaret (ed.), chapter II, pp. 35-64, Editora Arcádia, Lisboa, Portugal, 1971. This is a translation of "Éducation nouvelle et monde moderne", chapter II, Presses Universitaires de France, Paris, France, 1966.

Ganichev, A. A.; Soprunov, Sergei F. (2001). *Logo for the preliterate programmers (ver. 2)*, in Gerald Futschek (ed.) "Eurologo 2001: A Turtle Odyssey - Proceedings 8th European Logo Conference", ISBN 3-85403-156-4, Oesterreichische Computer Gesellschaft (OCG), Wien, Austria. Referenced from http://www.ocg.at/activities/books/volumes/band%20156/D12%20Soprunov.rtf, retrieved on May 14th, 2002.

Garg, Vijay Kumar (1988). *Specification and analysis of concurrent systems using STOCS model*, in "Proceedings of the Computer Networking Symposium", pp. 192-200, ISBN 0-8186-083, IEEE Computer Society Press, Washington, DC, USA. Referenced from the on-line version at http://intl.ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=4996, retrieved on February 2nd, 2005.

Gargaro, Anthony (1993). *A Brief Introduction to Ada*, in "The second ACM SIGPLAN conference on History of programming languages", ISBN 0-89791-570-4, pp. 343-344, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/154766.155380.

Garro, Yamilette Herrera (1992). *Experiencia didactica con ninos preescolares utilizando el lenguaje logo como herramienta de aprendizaje: creacion de un modelo*, Practica dirigida, Licenciatura en Educacion Preescolar, Teacher Training School, Faculty of Education, University of Costa Rica, San José, Costa Rica.

Gee, James Paul (2003). *What Video Games Have To Teach Us about Learning and Literacy*, ISBN 1-4039-6538-2, Palgrave MacMillan, New York, NY, USA. Referenced from the paperback edition, edited in May 2004.

Geiger, Christian; Mueller, Wolfgang; Rosenbach, Waldemar (1998). *SAM – An Animated 3D Programming Language*, in "Proceedings of the IEEE Symposium on Visual Languages", p. 228-235, ISBN 0-8186-8712-6, IEEE Computer Society, Washington, DC, USA. Referenced from the on-line version retrieved on October 24th, 2002, from http://www.c-lab.de/~wolfgang/vl98a.pdf.

General Motors (2003). *Sales & Market Analysis Media Briefing*, presentation given by Paul Ballew, Executive Director of Market & Industry Analysis, in December 15th, 2003. Referenced from http://media.corporate-ir.net/media_files/IROL/84/84530/presentations/GM_121503_PB.pdf, on-line document retrieved on August 4th, 2004.

Gesel Institute (2004). *Gesel Institute – Program Philosophy*, Web page retrieved on July 7th, 2005, from http://www.uwsp.edu/HPHD/Gesell/, Gesell Institute, School of Health Promotion and Human Development, University of Wisconsin-Stevens Point, Stevens Point, WI, USA.

Gillespie, Catherine Wilson (2004). *Seymour Papert's Vision for Early Childhood Education? A Descriptive Study of Head Start and Kindergarten Students in Discovery-based, Logo-rich Classrooms*, in Early Childhood Research & Practice, ISSN 1524-5039, vol. 6, no. 1, Early Childhood and Parenting (ECAP) Collaborative, Children's Research Center, University of Illinois at Urbana-Champaign, Champaign, IL, USA. Referenced from the on-line version, retrieved on February 24th, 2005, from http://ecrp.uiuc.edu/v6n1/gillespie.html.

Gillespie, Catherine Wilson; Beisser, Sally (2001). *Developmentally Appropriate LOGO Computer Programming with Young Children*, in "Information Technology in Childhood Education Annual", ISSN 1522-8185, vol. 2001, no. 1, pp. 229-245, Association for the Advancement of Computers in Education, Norfolk, VA, USA. Referenced from the on-line version, retrieved on August 10th, 2005, from http://www.aace.org/dl/files/ITCE/ITCE2001-229.pdf.

Gimbert, Belinda G.; Cristol, Dean S. (2003). *Technological Tools: Enhancing All Young Children's Cognitive and Social-Emotional Learning*, on-line article associated with a workshop at the meeting "3er Encuentro Internacional de Educación Inicial y Preescolar", Centros de Desarrollo Infantil del Frente Popular "Tierra y Libertad", Monterrey, Nuevo León, Mexico. Retrieved from www.cendi.org/interiores/encuentro2003/talleres/t_07.htm on February 24th, 2005.

Goldenberg, Paul (1974). *FASTR – A Simple Turtle Runner*, Logo Working Paper 30, MIT AI Lab, Massachussets Institute of Technology, Cambridge, MA, USA.

Goldenberg, Paul (2005). *Re: FASTR system - 1970s*, personal e-mail sent to Leonel Morgado (leonelm@utad.pt) on July 31st, 2005, at 00:00:24 -0400 (full text available on Annex II).

Gomes, Joaquim Ferreira (1977). *A educação infantil em Portugal: achegas para a sua história*, Almedina, Coimbra, Portugal.

Gordon, Peter (1994). *Robert Owen (1771-1858)*, in "PROSPECTS: quarterly review of comparative education", ISSN 0033-1538, vol. 24, no. 1/2, pp. 279-296, UNESCO International Bureau of Education, Geneva, Switzerland. Referenced from the on-line version, retrieved from http://www.ibe.unesco.org/International/Publications/Thinkers/ThinkersPdf/owene.PDF on May 4th, 2005.

Gorn, Saul (1957). *Standardized Programming Methods and Universal Coding*, in "Journal of the ACM", ISSN 0004-5411, vol. 4, no. 3, pp. 254-273, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/320881.320883 on August 16th, 2005.

Graham, Paul (2002). *The Roots of Lisp*, on-line article, retrieved on August 16th, 2005, from http://www.paulgraham.com/lib/paulgraham/jmc.ps.

Granada Learning (2005). *Did You Know? SEMERC*, in "Outlook", issue 4, May 2005, Granada Learning, London, UK. Referenced from the on-line version, retrieved from http://www.granada-learning.com/press/group_newsletter_downloads/outlookmay05.pdf on August 5th, 2005.

Green, Christopher D. (2000). *Introduction to Ada Lovelace's Translation of, and Notes to, Luigi F. Menabrea's "Sketch of the analytical engine invented by Charles Babbage, Esq." (1842/1843)*, in "Classics in the History of Psychology", ISSN 1492-3173, electronic resource,

York University, Toronto, Canada. Retrieved from http://psychclassics.yorku.ca/Lovelace/intro.htm on April 28th, 2004.

Gregg, Lee W. (1976). *Spatial concepts, spatial names. and the development of exocentric representations*, in "Children's Thinking: What Develops?", ISBN 0470265205, Robert S. Siegler (ed.), pp. 275-290, Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Grezlik, Amy (1999). *G. Stanley Hall*, Web page retrieved on July 7th, 2005, from http://www.muskingum.edu/~psych/psycweb/history/hall.htm, part of the site "History Of Psychology Archives", maintained by the Department of Psychology, Muskingum College, New Concord, OH, USA.

Grimshaw, Andrew S. (1993). *The Mentat Computation Model Data-Driven Support for Object-Oriented Parallel Processing*, Technical Report No. CS-93-30, May 28, 1993, University of Virginia – Computer Science, Charlottesville, VA, USA. Referenced from the on-line version, retrieved from http://www.cs.virginia.edu/~techrep/CS-93-30.ps.Z on May 18th, 2005.

Gruau, Frédéric; Lhuillier, Yves; Reitz, Philippe; Temam, Olivier (2004). *BLOB computing*, in "Proceedings of the 1st conference on Computing frontiers", ISBN 1-58113-741-9, pp. 125-139, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 17th, 2005, from http://doi.acm.org/10.1145/977091.977111.

Gugerty, Leo; Olson, Gary M. (1986). *Debugging by Skilled and Novice Programmers*, in "Proceedings of the SIGCHI conference on Human factors in computing systems", ISSN 0736-6906, pp. 171-174, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/22627.22367 on June 11th, 2005.

Hamilton, Linda (2004). *Intelligent Train, Village of Barboursville Elementary, January 6, 2004*, Web page at Marshall University, Huntington, WV, USA. Retrieved on August 2nd, 2005, from http://www.marshall.edu/LEGO/VOB0405/06Jan05VOBtrain/06Jan05Train.html.

Hansen, Per Brinch (1969). *RC 4000 Software: Multiprogramming System*, Part I General Description, pp. 13-52, Regnecentralen, Copenhagen, Denmark. Referenced from "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", pp. 153-197, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Hansen, Per Brinch (1978). *Distributed processes: A concurrent programming concept*, in "Communications of the ACM", ISSN 0001-0782, vol. 21, no. 11 (November 1978), pp. 934-941, ACM Press, New York, NY, USA. Referenced from "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", pp. 153-197, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Hansen, Per Brinch (2001). *The Invention of Concurrent Programming*, in "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Harel, Idit (1991). *Children Designers: Interdisciplinary Constructions for Learning and Knowing Mathematics in a Computer-Rich School*, ISBN 0-893-91788-5, Ablex Publishing, Norwood, New Jersey, USA.

Harper, Robert (2005). *Programming in Standard ML*, on-line reference book, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Retrieved on August 17th, 2005, from http://www.cs.cmu.edu/~rwh/smlbook/online.pdf.

Harrison, John Scott, as "Scott H." (1992). *On the Analogy Between Mind/Brain and Software/Hardware*. Electronic resource, retrieved on April 5th, 2004, from http://members.aol.com/ScottH9999/essays/mindsoft.htm.

Harry, Huge (1995). *A Computational Perspective on Twenty-First Century Music*, in "Contemporary Music Review", ISSN 0749-4467, vol. 14, no. 3, pp. 153-159, Routledge, Taylor &

Francis Group, New York, NY, USA. Referenced from the on-line version, retrieved on August 18th, 2005, from http://iaaa.nl/hh/brettonh.html.

Hartnell, Tim; Jones, Dilwyn (1982). *Programming Your ZX Spectrum*, ISBN 0907563198, Interface Publications, London, UK. Referenced from the Portuguese translation by Conceição Jardim and Lúcio Nogueira, *Como Programar o Seu ZX Spectrum*, 2nd edition, Editorial Presença, Lisbon, Portugal, 1984.

Harvey, Brian (1997). *Computer Science Logo Style*, vol. 1, "Symbolic computing", ISBN 0–262–58148–5, MIT Press, Cambridge, Massachussets, USA. Referenced from the on-line version at http://www.cs.berkeley.edu/~bh/v1-toc2.html, retrieved on August 31st, 2004.

Hauben, Jay Robert (1995). *John G. Kemeny: BASIC and DTSS: Everyone a Programmer*, in "Computer Pioneers", J. A. N. Lee, ed., IEEE Computer Society Press, Los Alamitos, California, USA. Referenced from the on-line version, retrieved on August 3rd, 2004, from http://dtss.dartmouth.edu/people2.php.

Hauben, Michael (1997). *Behind the Net: The Untold Story of the ARPANET and Computer Science*, in "Netizens – On the History and Impact of the Net", Michael Hauben & Ronda Hauben (eds.), ISBN 0-8186-7706-6, ch. 7, pp. 96-114, IEEE Computer Society Press, Los Alamitos, CA, USA. Referenced from the online version at http://www.columbia.edu/~rh120/ch106.x07, retrieved on May 18th, 2005.

Haugland, Susan W. (2000). *Computers and young children*, ERIC Digest no. ED438926, U.S. Department of Education, Washington DC, USA. Referenced from the on-line version, retrieved from http://ceep.crc.uiuc.edu/eecearchive/digests/2000/haugland00.pdf on February 27th, 2005.

Hesprich, David "DarkGrue" (1998). QuakeC Reference Manual, on-line document, retrieved from http://www.gue-tech.org/quake/qcrm/qcrm.pdf on August 18th, 2005.

Hewitt, Carl (1976). *Viewing control structures as patterns of passing messages*, A.I. Memo 410, December 1976, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Referenced from the on-line version, retrieved on August 12th, 2004, from http://www.cypherpunks.to/erights/history/actors/AIM-410.pdf.

High/Scope Educational Research Foundation (2005). *About High/Scope*, Web page retrieved from http://www.highscope.org/About/homepage.htm on July 22nd, 2005, owned by the High/Scope Educational Research Foundation, Ypsilanti, MI, USA.

Hildreth, Ellen C. (1975). *Logo for First and Second graders: A Teacher's Helper*, Logo Working Paper no. 40, MIT AI Lab, Massachussets Institute of Technology, Cambridge, MA, USA.

Hoare, Charles Anthony Richard (1971). *Towards a Theory of Parallel Programming*, in "Operating Systems Techniques", proceedings of a seminar at Queen's University, Belfast, Northern Ireland, UK, August-September 1971. Charles A. R. Hoare and Ronald H. Perrott, eds. ISBN 0123506506, Academic Press, New York, NY, USA, 1973. Referenced from "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", pp. 231-244, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Hoare, Charles Anthony Richard (1974). *Monitors: an operating system structuring concept*, Communications of the ACM 17, 10 (October 1974), 549-557, Association for Computing Machinery, New York, NY, USA. Referenced from "The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls", pp. 272-294, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Hoare, Charles Anthony Richard (1978). *Communicating Sequential Processes*, Communications of the ACM 21, 8 (August 1978), 666-677, Association for Computing Machinery, New York, NY, USA. Referenced from "The Origin of Concurrent Programming: From

Semaphores to Remote Procedure Calls", pp. 413-443, Per Brinch Hansen, ed., ISBN 0-387-95401-5, Springer-Verlag, New York, NY, USA.

Hodgins, Jessica K.; O'Brien, James F.; Bodenheimer Jr., Robert E. (1999). *Computer Animation*, in John G. Webster (ed.), "Wiley Encyclopedia of Electrical and Electronics Engineering", ISBN 0471139416, vol. 3, pp. 686-690, John Wiley & Sons, Inc., Hoboken, NJ, USA. Referenced from the on-line version, retrieved on August 18th, 2005, from http://www.cc.gatech.edu/gvu/animation/papers/ency.pdf.

Hoffman, James (2001). *Introduction to Structured Query Language*, on-line tutorial, retrieved from http://www.highcroft.com/highcroft/sql_intro.pdf on August 17th, 2005.

Hoić-Božić, Nataša (1997). *Hypermedia Supported Education*, M.Sc. Thesis, Fakulteta za računalništvo in informatiko (Faculty of Computer and Information Science), Univerza v Ljubljani (University of Ljubljana), Ljubljana, Slovenia. Referenced from the electronic version, retrieved from http://top.pefri.hr/mr/, on June 28th, 2004.

Holland, Owen (no date). *The Grey Walter Online Archive - Background information*, Web page at http://www.ias.uwe.ac.uk/Robots/gwonline/gwonline.html, University of the West of England, Bristol, UK. Retrieved on September 9th, 2004.

Holmboe, Christian; McIver, Linda; George; Carlisle (2001). *Research Agenda for Computer Science Education*, in "Proceedings of the 13th Annual Workshop of the Psychology of Programming Interest Group", G. Kadoda, ed., ISBN 1-85899-122-6, pp. 207-223, Sheffield Hallam University, Sheffield, UK. Referenced from the on-line version, retrieved on June 10th, 2005, from http://www.ppig.org/papers/13th-holmboe.pdf.

Home of the Underdogs (no date). *RobotWar*, Web page article, at http://www.the-underdogs.org/game.php?id=2507. Retrieved on October 4th, 2004.

Hopper, Mary E. (2001). *Logo World Worlds: A Cooperative Purdue-Headstart-Apple Project*, Web page at http://world.std.com/~mehopper/Pro/LWW.htm, retrieved on August 6th, 2005.

Hopper, Mary E.; Lawler, Robert W. (1997). *A progress report for the Headstart-Apple Logo Project*, in Lawler, Robert W. (ed.), "Learning with Computers", ISBN 1871576579, pp. 36-40, Intellect Books, Bristol, UK. Referenced from the on-line version, retrieved on August 6th, 2005, from http://world.std.com/~mehopper/R/Hopper-L_91.htm.

Hourcade, Juan Pablo; Bederson, Benjamin B.; Druin, Allison; Guimbretière, François (2004). *Differences in pointing task performance between preschool children and adults using mice*, in "ACM Transactions on Computer-Human Interaction (TOCHI)", ISSN 1073-0516, vol. 11, issue 4, pp. 357-386, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/1035575.1035577 on June 27th, 2005.

Hoyles, Celia and Noss, Richard (1999). *Playing with (and without) words*, "Proceedings of the Seventh European Logo Conference Eurologo '99, Sofia, Bulgaria, 22-25 August 1999", edited by Nikolov, Roumen; Sendova, Evgenia; Nikolova, Iliana and Derzhanski, Ivan. ISBN 954-9582-03-5, Virtech Ltd., Sofia, Bulgaria. An on-line version is available (last checked on August 9th, 2004) at http://www.ioe.ac.uk/playground/RESEARCH/papers/playword.htm.

Hudak, Paul; Fasel, Joseph H. (1992). *A gentle introduction to Haskell*, in "ACM SIGPLAN Notices", ISSN 0362-1340, vol. 27, no. 5, pp. 1-52, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/130697.130698 on August 17th, 2005.

Hudson, Kate (1998). *Using Roamer with the under 5s*, Valiant Technology, London, UK. Referenced from the on-line version, retrieved on August 2nd, 2005, from http://www.valiant-technology.com/freebies/under5s/under5s.pdf.

Hunt, Andrew (2001). *The Art in Computer Programming*, paper presented at the conference JAOO 2001, September 10-14, 2001, Scandinavian Congress Center, Århus, Denmark. Referenced from the on-line version at http://www.pragmaticprogrammer.com/articles/ArtInProgramming.pdf, retrieved on August 18th, 2005.

Hutchinson, Jamie (2003). *Don Bitzer and Gene Slottow: A PLATO-nic relationship thrived in ECE*, in "Ingenuity", Vol. 8, Number 1, May 2003, a newsletter for Electrical and Computer Engineering's student, staff, and faculty at the University of Illinois at Urbana-Champaign. Referenced from the on-line version at http://www.ece.uiuc.edu/ingenuity/503/PLATO.html, retrieved on July 30th, 2004.

Id Software (no date). *Quake – Overview*, Web page available at the URL http://www.idsoftware.com/games/quake/quake/, Id Software, Inc., Mesquite, TX, USA. Retrieved on August 18th, 2005.

Iglesias, Rosa Maria Iglesias (1999). *La tecnología como instrumento para el juego y el aprendizaje*, in "Quaderns Digitals", no. 14, ISSN 1575-9393, Valencia, Spain, 1999. Electronic article retrieved on April 28th, 2004, from http://www.quadernsdigitals.net/index.php?accionMenu=hemeroteca.VisualizaArticuloIU.visualiza&articulo_id=49.

Immersive Education Limited (2004). *MediaStage*, Web site based at http://www.kar2ouche.com/mediastage/, maintained by Immersive Education Limited, Oxford, UK. Retrieved on November 15th, 2004.

INE (2002). Inquérito à Utilização de Tecnologias da Informação e da Comunicação pelas Famílias (2001 e 2002), Instituto Nacional de Estatística, Lisboa, Portugal. Referenced from the on-line version, retrieved from http://www.ine.pt/prodserv/quadros/163/179/005/xls/01000000.xls on August 4th, 2004.

Ingalls, Dan; Kaehler, Ted; Maloney, John; Wallace, Scott; Kay, Alan (1997). *Back to the future: The story of Squeak, A practical Smalltalk written in itself*, in "Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications", pp. 318-326, ISSN 0362-1340, ACM Press, New York, NY, USA. Retrieved on September 28th, 2004, from http://www.cosc.canterbury.ac.nz/~wolfgang/cosc205/squeak.html.

INRA (2000). *Measuring Information Society 2000*, European Commission, Directorate General "Information Society", Brussels, Belgium. Referenced from the on-line version, retrieved from http://europa.eu.int/ISPO/basics/measuring/eurobaro/eurobaro53/docs/mis2000_report.doc on August 4th, 2004.

INT (2005). *ПервоЛого* (PervoLogo), article on the Web page http://www.int-edu.ru/logo/products.html, INT, Moscow, Russia. Retrieved on August 11th, 2005.

ITiS (2003). *ITiS*, Web page at http://www.itis.gov.se/, The Swedish Agency for School Improvement, Stockholm, Sweden. Last accessed on July 27th, 2005.

Jamieson, Stuart (1985). *MAC MAN*, Your Spectrum no. 12, March 1985, ISSN 0269-6983, Dennis Publishing, London, UK. Code extracted from the electronic version, on-line at http://www.users.globalnet.co.uk/~jg27paw4/pourri/mac_man.zip, retrieved on August 10th, 2004.

Jenkins, Tony (2002). *On the Difficulty of Learning to Program*, in "Proceedings of 3rd Annual Conference of the Learning and Teaching Support Network for Information and Computing Science", ISBN 0-9541927-1-0, pp. 53-58, LTSN-ICS, Loughborough University, Loughborough, UK. Referenced from the on-line version at http://www.ics.ltsn.ac.uk/pub/conf2002/tjenkins.pdf, retrieved on June 10th, 2005.

João-Monteiro, Maria; Cristóvão-Morgado, Rosa; Bulas-Cruz, Maria; Morgado, Leonel (2003). *A Robot in Kindergarten*, in "Eurologo'2003 Proceedings – Re-inventing technology on education", ISBN 972-8336-16-0, pp. 382-387, Cnotinfor, Coimbra, Portugal, 2003. At the URL

address http://home.utad.pt/~leonelm/papers/RobotinKindergarten/RobotinKindergarten.html there is an on-line version available (last checked on October 26th, 2004).

Johanson, Joyce (1998). *Teaching and Learning with Technology*, in ACTTive Technology vol. 13, issue 1, McComb Projects, College of Education and Human Services, Western Illinois University, Macomb, IL, USA. Referenced from the on-line version, retrieved on February 27th, 2005, from http://www.wiu.edu/thecenter/articles/teachlearn.html.

Johnston, Wesley M.; Paul Hanna, J. R.; Millar, Richard J. (2004). *Advances in Dataflow Programming Languages*, ACM Computing Surveys, ISSN 0360-0300, vol. 36, no. 1, pp. 1–34, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/1013208.1013209, retrieved on August 27th, 2004.

Jonassen, David H. (2000). *Computers as Mindtools for Schools*, 2nd ed., Prentice-Hall, Inc., Pearson Education, New Jersey, NJ, USA.

Jones, Simon Peyton; Blackwell, Alan; Burnett, Margaret (2003). *A User-Centred Approach to Functions in Excel*, in "Proceedings of the eighth ACM SIGPLAN international conference on Functional programming", ISBN 1-58113-756-7, pp. 165-176, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on June 28th, 2004, from http://www.research. microsoft.com/~simonpj/papers/excel/excel.pdf.

Kaasbøll, Jens J. (1999). *Exploring didactic models for programming*, in "Proceedings Norsk Informatikkonferanse - NIK'98", ISBN 82-519-1336-5, edited by Storøy, Sverre; Bjørnestad, Solveig; Hadjerrouit, Said; Krogdahl, Stein; Maus, Arne; Karlsen, Randi, pp. 195–203, Tapir Publishers, Trondheim, Norway. Referenced from the on-line version, retrieved on June 10th, 2005, from heim.ifi.uio.no/~jensj/NIK98.pdf.

Kafai, Yasmin B. (1995). *Minds in Play: Computer Game Design as a Context for Children's Learning*, ISBN 0-805-81513-9, Lawrence Erlbaum, Mahwah, New Jersey, USA.

Kafai, Yasmin B. and Resnick, Mitchel (1996). *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*, ISBN 0-8058-1985-1, Lawrence Erlbaum Associates Inc., Mahwah, NJ, USA. Referenced from the on-line version, retrieved from http://www.gseis.ucla.edu/faculty/kafai/faculty/Book_CIP_Intro.html on August 9th, 2004.

Kahn, Kenneth; Tribble, Eric Dean; Miller, Mark S.; Bobrow, Daniel G. (1986). *Objects in Concurrent Logic Programming Languages*, in "Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming", ISBN 0-89791-205-5, pp. 29-38, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/323779.323739.

Kahn, Ken (1995). *ToonTalk™—An Animated Programming Environment for Children*, in Diana Harris (ed.) & Regenia Bailey (assistant ed.), "NECC '95 proceedings: Baltimore, Maryland, June 17-19, 1995", ISBN 1564840808, Towson University, Towson, MD, USA. Referenced from the expanded version (Kahn, 1996a).

Kahn, Ken (1996a). *ToonTalk™—An Animated Programming Environment for Children*, in "Journal of Visual Languages & Computing", ISSN 1045-926X , vol. 7, issue 2, pp. 197-217, Academic Press, Cambridge, UK. Referenced from the on-line version, retrieved from http://www.toontalk.com/Papers/jvlc96.pdf on November 23rd, 1999.

Kahn, Ken (1996b). *Drawing on napkins, video game animation, and other ways to program computers*, Communications of the ACM, ISSN 0001-0782, vol. 39, issue 8, pp. 49 – 59, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on November 23rd, 1999, from http://www.toontalk.com/Papers/cacm.pdf.

Kahn, Ken (2001a). *ToonTalk and Logo – Is ToonTalk a colleague, competitor, successor, sibling, or child of Logo?*, in "Proceedings of EuroLogo 2001", Paedagogische Akademie der

Dioezese Linz, Linz, Austria. Referenced from the on-line version, retrieved on August 6th, 2004, from http://www.ocg.at/activities/books/volumes/band%20156/K11Kahn.doc.

Kahn, Ken (2001b). *Generalizing by Removing Detail: How Any Program Can Be Created by Working with Examples*, in "Your Wish Is My Command: Programming By Example", Henry Lieberman (ed.), ISBN 1-55860-688-2, pp. 21-43, Morgan Kaufmann Publishers, San Francisco, California, USA.

Kahn, Ken (2003). *[Squeakland] Logo vs. Squeak*, e-mail message sent to the mailing list squeakland@squeakland.org, on August 9th, 2003, at 14:08 PDT. Retrieved on September 19th, 2004, from the on-line archive at http://squeakland.org/pipermail/squeakland/2003-August/001748.html.

Kahn, Ken (2004). *The child-engineering of arithmetic in ToonTalk*, in "Proceeding of the 2004 conference on Interaction design and children: building a community", ISBN 1-58113-791-5, pp. 141-142, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/1017833.1017860 on December 12th, 2004, and from an expanded version available on-line at http://www.weblabs.eu.com/about/idc04.pdf, retrieved on December 12th, 2004.

Kahn, Ken (no date). *Thinking Skills*, research paper from the Playground project, on-line at http://www.ioe.ac.uk/playground/research/papers/thinking.htm, retrieved on November 23rd, 1999. It is also on-line at http://www.toontalk.com/English/skills.htm (last checked on August 9th, 2004).

Kahn, M. Kenneth and Saraswat, Vijay A. (1990). *Complete visualizations of concurrent programs and their executions*, in "Proceedings of the 1990 IEEE Workshop on Visual Languages: October 4-6,1990, Skokie, Illinois, USA", ISBN 0818620900, IEEE Computer Society Press, Los Alamitos, CA, USA. Referenced from the on-line version, retrieved on August 27th, 2004, from http://ieeexplore.ieee.org/iel2/316/3596/00128375.pdf?isNumber=3596&arnumber=128375&prod=CNF&arSt=7&ared=15&arAuthor=Kahn%2C+K.M.%3B+Saraswat%2C+V.A.

Kato, Hiroshi; Ide, Akiko (1995). *Using a game for social setting in a learning environment: AlgoArena – a tool for learning software design*, in "The first international conference on Computer support for collaborative learning", ISBN 0-8058-2243-7, pp. 195-199, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Kato, Hiroshi; Yamazaki, Keiichi; Suzuki, Hideyuki; Kuzuoka, Hideaki; Miki, Hiroyuki; Yamazaki, Akiko (1998). *Designing a video-mediated collaboration system based on a body metaphor*, in "Proceedings of CSCL '97: The Second International Conference On Computer Support For Collaboratiave Learning", ISBN 0-8058-3262-9, pp. 142-149, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA. Referenced from the on-line version, retrieved on May 19th, 2005, from http://www.oise.utoronto.ca/cscl/papers/kato.pdf.

Katz, Donald L. (1960, ed.). *Conference report on the use of computers in engineering classroom instruction*, in "Communications of the ACM", ISSN 0001-0782, vol. 3, no. 10, pp. 522-527, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 31st, 2004, from http://doi.acm.org/10.1145/367415.993453.

Kay, Alan (1984). *Computer Software*, in "Scientific American", ISSN 0036-8733, vol. 251, no. 3, pp. 41-47, Scientific American, Inc., New York, NY, USA.

Kay, Alan (1993a). *Foreword*, in Allen Cypher (ed.), "Watch What I Do: Programming by Demonstration", ISBN 0-262-03213-9, pp. xi-xv, MIT Press, Cambridge, Massachusetts, USA. (Cited from the second printing, 1994.)

Kay, Alan (1993b). *The Early History of Smalltalk*, in "The second ACM SIGPLAN conference on History of programming languages", ISSN 0362-1340, pp. 69-95, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on September 28th, 2004, from http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html.

Kay, Alan (2003a). *[Squeakland] Logo vs. Squeak*, e-mail message sent on August 8th, 2003, 9:29 PDT, to the mailing list squeakland@squeakland.org. Retrieved on September 14th, 2004, from the on-line archive at http://squeakland.org/pipermail/squeakland/2003-August/001745.html.

Kay, Alan (2003b). *[Squeakland] Logo vs. Squeak*, e-mail message sent on August 9th, 2003, 12:09 PDT, to the mailing list squeakland@squeakland.org. Retrieved on September 19th, 2004, from the on-line archive at http://squeakland.org/pipermail/squeakland/2003-August/001752.html.

Kay, Alan (2005). *Re: [Squeakland] Squeak Etoys and preschoolers*, personal e-mail message, sent on August 4th, 2005. Full text is available in Annex II.

Kay, Alan (no date). "*Etoys and SimStories in Squeak*", on-line article in applet, from http://www.squeakland.org/project.jsp?/projects/etoys/EtoysSimstories.004.pr, retrieved on August 6th, 2004.

Kearsley, Greg (2004). *Explorations in Learning & Instruction: The Theory Into Practice Database*, Web site retrieved from http://tip.psychology.org/ on May 18th, 2004.

Kelleher, Caitlin Lara; Pausch, Randy (2003). *Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers*, Technical Report CMU-CS-03-137, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Referenced from the on-line version at http://reports-archive.adm.cs.cmu.edu/anon/2003/CMU-CS-03-137.pdf, retrieved on August 20th, 2004.

Kelly, Kevin (1998). *The Third Culture*, Science, vol. 279, issue 5353, pp. 992-993, February 13th, 1998, DOI: 10.1126, ISSN 1095-9203, American Association for the Advancement of Science, Washington, D.C., USA. Referenced from the on-line version, retrieved on August 7th, 2004, from http://www.sciencemag.org/cgi/content/full/279/5353/992.

Kemeny, John G.; Kurtz, Thomas E. (1984). *Bringing Up BASIC*, in "Digital Deli – The Comprehensive, User-Lovable Menu of Computer Lore, Culture, Lifestyles and Fancy by The Lunch Group & Guests", Steve Ditlea, ed., ISBN 0894805916, Workman Publishing Company, Inc., New York, NY, USA. Referenced from the on-line version, retrieved on August 3rd, 2004 from http://www.atariarchives.org/deli/basic.php.

Kent NGfL, National Grid for Learning (no date). *Early ICT – A collection of themed ICT activities and resources to support learning in the Foundation Stage*, Web site at http://www.naturegrid.org.uk/infant/earlyict/, last accessed on July 27th, 2005. Kent National Grid for Learning, Kent County Council, Maidstone, UK.

KidSmart (no date). *KidSmart Guide to Early Learning & Technology – for Home and School*, Web site at http://www.kidsmartearlylearning.org/, United Way of America (Alexandria, VA, USA) and IBM Corporation (White Plains, NY, USA). Last accessed on July 27th, 2005.

Kikin-Gil, Erez (2005). *Eco-Pods – Tangible user interfaces for learning systems thinking*, Master's thesis, Interaction Design Institute Ivrea, Ivrea, Italy. Referenced from the on-line version, retrieved from http://www.tiltool.com/images/Erez_kikin_gil_Eco_Pods.pdf on August 2nd, 2005.

Kilpatrick, William Heard (1951). *Philosophy of education*, The Macmillan Company, New York, NY, USA.

Kim, Eugene Eric and Toole, Betty Alexandra (1999). *Ada and the First Computer*, Scientific American, May 1999, pp. 76-81, ISSN 0036-8733, Scientific American, Inc., New York, NY, USA.

Kimura, Takayuki D.; Choi, Julie W.; Mack, Jane M. (1990). *Show and Tell: A Visual Language*, in "Visual Programming Environments: Paradigms and Systems", E. P. Glinert (ed.), ISBN 0818689730, pp. 397-404, IEEE Computer Society Press, Los Alamitos, CA, USA

Kindborg, Mikael (2003). *Concurrent Comics – programming of social agents by children*, Linköping Studies in Science and Technology, Dissertation No. 821, Department of Computer and Information Science, Linköpings universitet, Linköping, Sweden.

Kinderet (2002a). *Continuing Training of Early Childhood Nurseries: Practices and Models – The Portuguese, Spanish, English, Swedish and Bulgarian Contexts*, Escola Superior de Educação de Beja, Beja, Portugal. Referenced from the on-line version, retrieved on May 18th, 2005, from http ://www.eseb.ipbeja.pt/kinderet/Documentos/1%20Characterisation_Models%20and%20Practices_3 .rtf.

Kinderet (2002b). *Diagnóstico Inicial de necessidades de formação contínua de educadores em Tecnologia Educativa*, Escola Superior de Educação de Beja, Beja, Portugal. Referenced from the on-line version, retrieved on May 18th, 2005, from http://www.eseb.ipbeja.pt/kinderet/Document os/resultados_an%E1lise%20de%20necessidades2A(Portugu%EAs).rtf.

Kinnersley, William Morris (1991). *The Language List*, Web page at http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm, retrieved on August 24th, 2004.

Klein, Pnina S.; Nir-Gal, Ofra; Darom, Efraim (2000). *The Use of Computers in Kindergarten, With or Without Adult Mediation; Effects on Children's Cognitive Performance and Behavior*, in "Computers in Human Behavior", ISSN 0747-5632, vol. 16, issue 6 (November 1st 2000), pp. 591-608, Elsevier Science, Essex, UK. Referenced from the on-line version, last retrieved on March 27th, 2003, from http://web.macam98.ac.il/~nirgalo/a-publish&study/artical-klein&nir-gal&darom.htm.

Knoll, Michael (1997). *The Project Method: Its Vocational Education Origin and International Development*, in "Journal of Industrial Teacher Education", ISSN 0022-1864, vol. 34, no. 3, pp. 59-80, National Association of Industrial and Technical Teacher Educators, Portage, MI, USA. Referenced from the on-line version, retrieved on July 21st, 2005, from http://scholar.lib.vt.edu/ejournals/JITE/v34n3/Knoll.html.

Komenský, Jan Ámos (1633). *Informatorium Školy Mateřski*, 1st ed., Leszno, Poland. Referenced from the 1896 English translation, "Comenius' School Of Infancy: An essay on the education of youth during the first six years", Will S. Monroe (ed.), D.C. Heath, Boston, MA, USA. Available on-line, retrieved from http://wordsworth.roehampton.ac.uk/digital/froarc/comsch/ind.asp on May 5th, 2005.

Komlos, Maxine (1998). *Talent Education: The Suzuki Method*, Web article at http://www.musicstaff.com/lounge/article7.asp, MusicStaff.com, Somerset, NJ, USA. Retrieved on September 2nd, 2004.

Kopplin, John (2002). *An Illustrated History of Computers*, computersciencelab.com, Boise, ID, USA. Web page at http://www.computersciencelab.com/ComputerHistory/History.htm, retrieved on August 13th, 2004.

Kreutzer, Wolfgang (1998). *Basic Aspects of Squeak and the Smalltalk-80 Programming Language*, Web tutorial at http://www.cosc.canterbury.ac.nz/~wolfgang/cosc205/smalltalk1.html, retrieved on September 18th, 2004.

Kushan, Barbara (1994). *Preparing programming teachers*, in "Proceedings of the twenty-fifth SIGCSE symposium on Computer science education", ISBN 0-89791-646-8, pp. 248-252, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on January 21st, 2001, from http://doi.acm.org/10.1145/191029.191134.

Kuttner, Henry and Moore, C.L., as by Padgett, Lewis (1943). *Mimsy Were The Borogoves*, in Astounding, February 1943. Referenced from "The Golden Years of Science Fiction – Third Series", Isaac Asimov and Martin H. Greenberg eds., ISBN 0-517-435225, Bonanza Books, New York, USA, 1984.

Kypros-Net (no date). *ΛΕΞΙΚΟ - LEXICON: Greek-English-Greek dictionary*, on-line resource at http://www.kypros.org/cgi-bin/lexicon, Kypros-Net Inc., Cambridge, Massachusetts, USA. Accessed on August 5th, 2004.

Labrie, John (2002). Personal statement, during the documentaries included in the DVD "The Lord of the Rings – The Fellowship of the Ring – Extended Edition", New Line Cinema, New York, NY, USA. Referenced from the DVD, but cited from transcription retrieved on August 16th, 2005, from http://www.warofthering.net/ahobbitstale/extras/extras_makingof.htm.

Lakoff, George (1992). *The Contemporary Theory of Metaphor*, in "Metaphor and Thought", 2nd edition, pp. 202-251, ISBN 0521405610, Cambridge University Press, Cambridge, UK. Referenced from the on-line version, retrieved on September 3rd, 2004, from http://www.ac.wwu.edu/~market/semiotic/lkof_met.html.

Lakoff, George; Johnson, Mark (1980). *Metaphors We Live By*, ASIN 0226468003, University of Chicago Press, Chicago, IL, 1980. Referenced from the on-line excerpts at http://theliterarylink.com/metaphors.html, retrieved on September 3rd, 2004.

Lam, Ho Cheong (2003). *Misconceptions of Students about Programming Constructs*, computer presentation for the course "Psychological Issues in the Teaching and Learning of Computing", MITE 6218-A&B, part of the studies for Master of Science in Information Technology in Education, University of Hong Kong, Hong Kong, China. Referenced from the on-line version, retrieved from http://www.cmi.hku.hk/hclam/mite6218/error.pdf on August 12th, 2004.

Lascarides, V. Celia; Hinitz, Blythe F. (2000). *History of Early Childhood Education*, ISBN 0-815-31794-8, Falmer Press, Taylor & Francis Group, New York, NY, USA.

LCSI, Logo Computer Systems Inc. (2004). *MicroWorldsJR – MicroWorldsJR Resource Book with Extended Reference Guide*, ISBN 2-89371-546-X, LCSI, Highgate Springs, Vermont, USA. Electronic book at http://www.lcsi.ca/pdf/microworldsjr/mwjr-resource-extended.pdf, retrieved on April 4th, 2005.

Learning and Teaching Scotland (2003a). *Early Learning, Forward Thinking: The Policy Framework for ICT in Early Years*, ISBN 1-85955-795-3, Learning and Teaching Scotland, Glasgow, Scotland, UK. Referenced from the on-line version, retrieved on July 27th, 2005, from http://www.ltscotland.org.uk/earlyyears/files/ict_framework.pdf.

Learning and Teaching Scotland (2003b). *Come back in two years!*, ISBN 1-85955-779-1, Learning and Teaching Scotland, Glasgow, Scotland, UK. Referenced from the on-line version, retrieved from http://www.ltscotland.org.uk/earlyyears/files/comebackintwoyears.pdf on July 27th, 2005.

LEGO (no date-1). *RCX Code*, Web page retrieved on October 25th, 2004, from http://mindstorms.lego.com/eng/products/ris/rissoft.asp.

LEGO (no date-2). *Intelligent Locomotive*, Web page retrieved on October 27th, 2004, from http://shop.lego.com/product.asp?p=10052&cn=211&d=33&t=10.

Legrand, Louis (1993). *Célestin Freinet (1896-1966)*, in "PROSPECTS: quarterly review of comparative education", ISSN 0033-1538, vol. 23, no. 1/2, pp. 403-418, UNESCO International Bureau of Education, Geneva, Switzerland. Referenced from the on-line version, retrieved from http://www.ibe.unesco.org/International/Publications/Thinkers/ThinkersPdf/freinete.PDF on July 20th, 2005.

Lehrer, Richard; deBernard, Ann (1987). *Language of learning and language of computing: The perceptual-language model*, in "Journal of Educational Psychology", ISSN 0022-0663, vol. 79, no. 1, pp. 41-48, American Psychological Association, Washington, DC, USA.

Lehrer, Richard; Guckenberg, Thomas; Sancilio, Leonard. (1988). *Influences of Logo on children's intellectual development*, in Richard E. Mayer (ed.), "Teaching and learning computer

programming: Multiple research perspectives", ISBN 0805800735, pp. 75-110, Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Leung, Wai Man (2003). *The Shift From a Traditional to a Digital Classroom: Hong Kong Kindergartens*, in "Childhood Education: Infancy Through Early Adolescence", vol. 80, no. 1, pp. 12-17, Association for Childhood Education International, Olney, MA, USA.

Lewis, Clayton; Rader, Cyndi; Brand, Cathy; Carlone, Heidi (1997). *Models Children Build: Content, Logic and Educational Impact*, paper presented at the Annual Meeting of the National Association for Research in Science Teaching (NARST) in April 1997, in Chicago, IL, USA. Referenced from the on-line version, at http://www.cs.colorado.edu/~crader/NARST97/ModAnl-Narst97.html, retrieved on July 10th, 2001.

Lieberman, Henry (2001). *Introduction*, in "Your Wish Is My Command: Programming By Example", Henry Lieberman, ed., ISBN 1-55860-688-2, Morgan Kaufmann Publishers, San Francisco, California, USA.

Lino, Dalila (1998). *O Modelo Curricular para a Educação de Infância de Reggio Emilia: Uma Apresentação*, in "Modelos Curriculares para a Educação de Infância", Júlia Oliveira Formosinho (ed.), 2nd edition (1998), ISBN 972-0-34451-2, pp. 93-136, Porto Editora, Oporto, Portugal.

Lionet, François; Lamoureux, Yves (1996). *The Games Factory*, Europress Software Ltd, Macclesfield UK. Referenced from the manual update of February 6th 2000, available on-line. Retrieved from http://www.americaconnects.net/research/SouthendFIG/TGFmanual.PDF on October 1st, 2004.

Littlefield, J.; Delclos, V. R.; Lever, S.; Clayton, K. N.; Bradsford, J. D.; Franks, J. J. (1988). *Learning Logo: Method of teaching, transfer of general skills, and attitudes toward school and computers*, in "Teaching and learning computer programming: Multiple research perspectives", R. E. Mayer (ed.), ISBN 0805800735, pp. 111-135, Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Lock, Andrew J. (2003). *Lecture 14: Bruner*, Web-based lecture notes for the course "Evolution, Learning and Culture", School of Psychology, Massey University, Palmerston North, New Zealand. Retrieved from http://evolution.massey.ac.nz/lect14/lect1400.htm on July 17th, 2005.

Logiblocks (no date). *Understanding Logiblocs*, Web page retrieved on May 18th, 2005, from http://www.logiblocs.com/understanding_logiblocs.htm, Logiblocs, St. Albans, UK.

Logo Foundation (2000). *What is Logo?*, Web page at http://el.media.mit.edu/logo-foundation/logo/index.html, Logo Foundation, Epistemology and Learning group, The Media Laboratory, Cambridge, MA, USA. Retrieved on August 5th, 2004.

Lucas, Christopher (n.d.). *A Educação dos Escribas e a Instrução na Mesopotâmia*, in "História Mundial da Educação", ISBN 972-703-067-X, vol. 1, pp. 37-60, RÉS-Editora, Porto, Portugal. This volume is a translation by Evaristo Santos of Gaston Mialaret & Jean Vial (eds.), 1981, "Histoire mondiale de l'éducation", ISBN 2-130-36776-3, vol. 1, Presses Universitaires de France, Paris, France.

Lynch, Sharon A.; Warner, Laverne (2004). *Computer Use in Preschools: Directors' Reports of the State of the Practice*, in "Early Childhood Research & Practice", ISSN 1524-5039, vol. 6, no. 2, Early Childhood and Parenting (ECAP) Collaborative, Children's Research Center, University of Illinois at Urbana-Champaign, Champaign, IL, USA. Referenced from the on-line version, retrieved on February 24th, 2005, from http://ecrp.uiuc.edu/v6n2/lynch.html.

Maas, Benjamin (2004), under pseudonym "Zimond". *Point & Click Development Kit*, Web site http://pacdk.gamezworld.de/english.html, retrieved on October 1st, 2004.

Macchiusi, Lina (1997). *Children, robotics and problem solving: Technology in the early childhood classroom*, in "Australian Educational Computing", ISSN 0816-9020, vol. 12, no. 2, Australian Council for Computers in Education, Belconnen, Australian Capital Territory, Australia. Referenced from the on-line version at http://www.acce.edu.au/journal/journals/vol12_2.pdf, retrieved on May 20th, 2002.

Macromedia (2004a). *Macromedia Flash MX 2004 – Getting Started with Flash*, Macromedia Inc., San Francisco, CA, USA. Referenced from the on-line version, retrieved on August 18th, 2005, from http://download.macromedia.com/pub/documentation/en/flash/mx2004/fl_getting_started.pdf.

Macromedia (2004b). *Macromedia Flash MX 2004 – Using ActionScript in Flash,* Macromedia Inc., San Francisco, CA, USA. Referenced from the on-line version, retrieved from http://download.macromedia.com/pub/documentation/en/flash/mx2004/fl_usingas_in_flash.pdf on August 18th, 2005.

Magical Ears (no date). *Disney Animation & Computers*, Web page retrieved on August 18th, 2005, at http://www.magicalears.com/films/Disney%20Animation%20&%20Computers/index.php.

Malaguzzi, Loris (1998). *The Story As Told By Loris Malaguzzi In An Interview With Lella Gandini*, in "The Hundred Languages of Children: The Reggio Emilia Approach – Advanced Reflections", ISBN 156750311X, ch. 3, Carolyn Edwards, Lella Gandini, George Forman (eds.), Ablex Publishing, Norwood, New Jersey, USA. Referenced from the on-line version, retrieved from http://www.reggioinspired.com/reggiohistory1.htm on July 23rd, 2005.

Maloney, John; Burd, Leo; Kafai, Yasmin; Rusk, Natalie; Silverman, Brian; Resnick, Mitchel (2004). *Scratch: A Sneak Preview*, on pp. 104-109 of the proceedings of the "Second International Conference on Creating, Connecting, and Collaborating through Computing", held in 2004 in Kyoto, Japan. Referenced from the on-line version, retrieved on August 5th, 2005, from http://llk.media.mit.edu/projects/scratch/ScratchSneakPreview.pdf.

Maqsood, Zaeem Max (2001). *The Potential for Modelling and Learning in 'The Sims'™*, student paper for the module "ICT.03C2 – Learning with Virtual Worlds: models, data and control", part of the Master's course "Information and Communications Technology in Education", Institute of Education, University of London, London, UK. Referenced from the on-line version, retrieved from http://www.bloglinker.com/papers/TheSims(TM).pdf on September 1st, 2004.

Marsham, Monica (2004). *Roamer Robots in the preschool and primary setting*, abstract of paper presented at the Australian Computers in Education Conference 2004, Adelaide, Australia, July 5th-8th 2004, Australian Council for Computers in Education, Belconnen, Australian Capital Territory, Australia. Referenced from the on-line version, retrieved on August 2nd, 2005, from http://www.acec2004.info/confpapers/paperdetails.asp?pid=7211&uid=&docid=235.

Martin, F.; Colobong, Genee Lyn; Resnick, M. (1999). *Tangible Programming with Trains*. Web page reported by Kelleher & Pausch (2003) at el.www.media.mit.edu/projects/trains (no longer available).

Maulsby, David; Turransky, Alan (1993). "*A Programming by Demonstration Chronology: 23 Years of Examples*", in "Watch What I Do: Programming by Demonstration", appendix A, pp. 529-538, Allen Cypher, ed., ISBN 0-262-03213-9, MIT Press, Cambridge, Massachusetts, USA. (Cited from the second printing, 1994.)

Mayer, Richard E., Dyck, Jennifer L.; Vilberg, William (1986). *Learning to program and learning to think: What's the connection?*, in "Communications of the ACM", ISSN 0001-0782, vol. 29, no. 7, pp. 605-610, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on January 21st, 2001, from http://doi.acm.org/10.1145/6138.6142.

McCarthy, John (1960). *Recursive functions symbolic expressions and their computation by machine, Part I*, in "Communications of the ACM", ISSN 0001-0782, vol. 3, issue 4 (April 1960),

pp. 184-195, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/367177.367199 on February 9th, 2004.

McCarthy, John (1979). *History of Lisp*, in "History of Programming Languages", ISBN 0127450408, Academic Press, Inc., Orlando, FL, USA (1981). Referenced from the on-line version, retrieved on September 9th, 2004, from http://www-formal.stanford.edu/jmc/history/lisp.ps.

McDonald, Jason K.; Yanchar, Stephen C.; Osguthorpe, Russel T. (2005). *Learning from Programmed Instruction: Examining Implications for Modern Instructional Technology*, Educational Technology Research and Development, ISSN 1042-1629, vol. 53, issue 2, pp. 94-89, Association for Educational Communications and Technology, Bloomington, IN, USA. Referenced from the electronic version, retrieved from http://cid.byu.edu/mcdonald/pi.pdf on July 2nd, 2004.

McIver, Linda Kathryn (2000). *The Effect of Programming Language on Error Rates of Novice Programmers*, in "Collected Papers of the 12th Annual Workshop of the Psychology of Programming Interest Group (PPIG-12), Corigliano Calabro, Cosenza, Italy", Blackwell, Alan F. and Bilotta, Eleonora (eds.), ISBN 8887373213, pp. 181-192, Edizioni Memoria, Corigliano Calabro, Italy. Referenced from the on-line version at http://www.ppig.org/papers/12th-mciver.pdf, retrieved on June 10th, 2005.

McIver, Linda Kathryn (2001). *Syntactic and Semantic Issues in Introductory Programming Education*, PhD Dissertation, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia. Referenced from the on-line version, retrieved on September 9th, 2004, from http://www.csse.monash.edu.au/~lindap/papers/LindaMcIverThesis.pdf.

McNeil, Sara (no date). *A hypertext history of instructional design*, Web site hosted at the College of Education of the University of Houston, retrieved on July 1st, 2004, from http://www.coe.uh.edu/courses/cuin6373/idhistory/index.html.

McNerney, Timothy Scott (2000). *Tangible Programming Bricks: An approach to making programming accessible to everyone*, Master thesis, Program in Media Arts and Sciences, School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved from www.media.mit.edu/people/mc/mcnerney-sm-thesis.pdf on October 6th, 2004.

McNerney, Timothy Scott (2003). *From turtles to Tangible Programming Bricks: explorations in physical language design*, in "Personal and Ubiquitous Computing", volume 8, number 5, September 2004, pp. 326 - 337, ISSN 1617-4909, DOI 10.1007/s00779-004-0295-6, Springer-Verlag, London, UK. Referenced from the on-line version, retrieved from http://www.springerlink.com/media/PF9DMMYWYL0RT16PXGFY/Contributions/K/B/8/7/KB873HV41HCHTQVK.pdf on October 27th, 2004.

Meisels, Samuel J. (1988). *Developmental Screening in Early Childhood: The Interaction of Research and Social Policy*, in "Annual Review of Public Health", ISSN 0163-7525, vol. 9, issue 1, pp. 527-550, Annual Reviews, Palo Alto, CA, USA. Referenced from the electronic version, retrieved from http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.pu.09.050188.002523 on July 7th, 2005.

Menabrea, Luigi Federico (1842). *Notions sur la machine analytique de M. Charles Babbage*, Bibliothèque Universelle de Genève, 41, 352-76, Geneva, Switzerland. Referenced from an electronic version of the English translation (Byron-King, 1843).

Meyer, Alexander (2004). *Lemmings Games*, fan Web site, at the URL address http://www.kallex.de/lemmings/. Retrieved on October 7th, 2004.

Microsoft (2004a). *Excel home page*, electronic resource available on-line at the URL address http://www.microsoft.com/excel/ (accessed on June 4th, 2004).

Microsoft (2004b). *Word home page*, electronic resource available on-line at the URL address http://www.microsoft.com/word/ (accessed on June 8th, 2004).

Microsoft (2005). *PowerPoint home page*, electronic resource available on-line at the URL address http://www.microsoft.com/powerpoint/ (accessed on August 17th, 2005).

Miller, George A. (1956). *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*, The Psychological Review, 1956, vol. 63, pp. 81-97, ISSN 0033-295X, American Psychological Association, Washington, DC, USA. Referenced from the on-line version at http://www.well.com/user/smalin/miller.html, retrieved on May 18th, 2004.

Miller, Gloria E.; Emihovich, Catherine (1986). *The effects of mediated program instruction on preschool children's self-monitoring*, in "Journal of Educational Computing Research", ISSN 0735-6331, vol. 2, no. 3, pp. 283-297, Baywood Publishing Co., Amityville, NY, USA.

Ministério da Educação (1997). *Orientações Curriculares para a Educação Pré-Escolar*, ISBN 972-742-087-7, Department of Basic Education, Ministry of Education, Lisbon, Portugal. Available on-line at http://www.deb.min-edu.pt/fichdown/pre_escolar/Orientacoes_curriculares.pdf. An English translation was included in the 1998 title by the Portuguese Ministry of Education, "Early Childhood Education in Portugal": ISBN 972-742-094-X, Ministério da Educação, Departamento da Educação Básica, Lisbon, Portugal.

Minsky, Marvin (1970). *Form and Content in Computer Science*, 1970 ACM Turing Lecture, in "Journal of the Association for Computing Machinery", ISSN 0004-5411, vol. 17, no. 2, pp. 197-215, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/321574.321575 on June 28th, 2004.

Minsky, Marvin (1988). *The Society of Mind*, ISBN 0671657135, Simon & Schuster, New York, NY, USA.

Miranda, Guilhermina Lobato (1989). *A linguagem LOGO no pré-escolar: avaliação de alguns efeitos cognitivos decorrentes da actividade de programação*, Master's thesis, Faculty of Psychology and Education Sciences, University of Lisbon, Lisbon, Portugal.

Miranda, Guilhermina Lobato (1990). *Crianças do pré-escolar programam em LOGO - Análise dos efeitos cognitivos de um ano de experiência*, in "Análise Psicológica", ISSN 0870-8231, vol. VIII, no. 1, pp. 47-60, Instituto Superior de Psicologia Aplicada, Lisbon, Portugal.

Misra, Atmakand (n.d.). *A Educação na Índica, na Antiguidade e na Idade Média*, in "História Mundial da Educação", ISBN 972-703-067-X, vol. 1, pp. 95-116, RÉS-Editora, Porto, Portugal. This volume is a translation by Evaristo Santos of Gaston Mialaret & Jean Vial (eds.), 1981, "Histoire mondiale de l'éducation", ISBN 2-130-36776-3, vol. 1, Presses Universitaires de France, Paris, France.

Mitchell, John C. (2003). *Concepts in Programming Languages*, ISBN 0-521-78098-5, Cambridge University Press, Cambridge, UK.

Monroe, Paul (1907). *A Brief Course in the History of Education*, The MacMillan Company, New York, NY, USA. Referenced from the Brazilian Portuguese translation of the 1949 edition, translated and annotated by Idel Becker in 1988: *História da Educação*, Companhia Editora Nacional, São Paulo, Brazil.

Montemayor, Jaime (2001). *Physical Programming: Technology Tools for Kindergarten Children to Program Physical Interactive Environments*, extended abstract of a presentation at the Doctoral Consortium of the CHI 2001 Conference on Human Factors in Computing, held between March 31st and April 5th 2001, in Seattle, WA, USA. Referenced from the on-line version at http://www2-data.informatik.unibw-muenchen.de/HCC01/ChildProg/Montemayor-abstract.pdf, last accessed on August 1st, 2005.

Montemayor, Jaime; Druin, Allison; Chipman, Gene; Farber, Allison; Leigh Guga, Mona (2004). *Tools for children to create physical interactive storyrooms*, in "Computers in Entertainment (CIE)", vol. 2, issue 1, January 2004, pp. 12-36, ISSN 1544-3574, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/973801.973821, on September 30th, 2004.

Montemayor, Jaime; Druin, Allison; Farber, Allison; Simms, Sante; Churaman, Wayne; D'Amour, Allison (2002). *Physical programming: designing tools for children to create physical interactive environments*, in "Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves", pp. 299-306, ISBN 1-58113-453-3, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/503376.503430, last retrieved on November 8th, 2004.

Montessori, Maria (1946). *Education for a New World*, Kalakshetra Publications Press, Adyar, Madras, India. Referenced from the online version of the 1995 edition by Clio Press, Oxford, UK, retrieved on May 26th, 2005, from http://www.moteaco.com/clio/world.html.

Montessori, Maria (1955). *Formazione dell'uomo: pregiudizi e nebule: analfabetismo mondiale*, Garzanti, Milan, Italy. Referenced from the online version of the English-language translation by A. M. Joosten (1989), *The Formation of Man*, ISBN 1-85109-097-5, Clio Press, Oxford, UK, retrieved from http://www.moteaco.com/clio/form.html on May 26th, 2005.

Morejón, Julián Betancourt; Sierra, Maria de los Dolores Valadez (1998). *Jerome Bruner: uno de los precursores de los estudios sobre estrategias cognitivas*, in "Educar", no. 6, July/September 1998, Secretaria de Educación, Gobierno de Jalisco, Guadalajara, Jalisco, Mexico. Referenced from the on-line version, retrieved on July 17th, 2005, from http://educacion.jalisco.gob.mx/consulta/educar/06/6betan.html.

Morgado, Leonel; Cruz, Maria Gabriel Bulas; Kahn, Ken (2001). *Working in ToonTalk with 4- and 5-year olds*, paper presented at the seminar "Playground - novos ambientes de aprendizagem", at Casa de Vilar, Porto, Portugal, 2001, organized by Cnotinfor, Coimbra, Portugal. An edited version, *Using ToonTalk in Kindergartens*, was published in "Proceedings of the IADIS International Conference e-Society 2003", vol. II, ISBN 972-98947-0-1, pp. 988-994, IADIS, Lisbon, Portugal, 2003. This edited version is also available on-line at the following address (last checked on May 31st, 2005): http://home.utad.pt/~leonelm/TTon4-5.htm.

Morgado, Leonel; Cruz, Maria Gabriel; Kahn, Ken (2003b). *ToonTalk in Kindergartens: Field Notes*, in Mendez-Vilas, Antonio; Mesa González, José António; Solo de Zaldívar Maldonado, Inés (eds.), "Information Society and Education - Proceedings of the International Conference on Information and Communication Technologies in Education (ICTE2002)", Journal of Digital Contents, ISSN 1696-313X, ISBN 84-607-8369-3, vol. 1, no 1, Formatex, Badajoz, Spain. There is an on-line version at http://home.utad.pt/~leonelm/papers/TTICTE2002/TTknf.html, last accessed on July 27th, 2005.

Morgado, Leonel; Cruz, Maria; Kahn, Ken (2003a), *Taking Programming into Kindergartens: Exploratory Research Activities Using ToonTalk*, in "Eurologo'2003 Proceedings – Re-inventing technology on education", ISBN 972-8336-16-0, pp. 178-189, Cnotinfor, Coimbra, Portugal. There is an version on-line, last accessed on July 27th, 2005, at http://home.utad.pt/~leonelm/papers/eurologo2003/EuroLogo2003.htm.

Morgado, Leonel; Morgado, Rosa; Cruz, Maria Gabriel; Kahn, Ken (2005). *Embedding Computer Activities into the Context of Preschools*, in "Challenges 2005" (conference proceedings on CD-ROM), Paulo Dias & Cândido Varela de Freitas (eds.), ISBN 972-8746-13-05, pp. 471-478, Centro de Competências Nónio Séc. XXI, Universidade do Minho, Braga, Portugal.

Mosconi, Mauro and Porta, Marco (2000). *Iteration constructs in data-flow visual programming languages*, Computer Languages, ISSN 0096-0551, Vol. 26, Issues 2-4, July 2000,

pp. 67-104, Elsevier Science Publishers Ltd., Essex, UK. Referenced from the on-line version, retrieved from http://dx.doi.org/10.1016/S0096-0551(01)00009-1 on August 27th, 2004.

Mota, Carlos A. M. G. (2003). *Breve História da Educação no Ocidente*, ISBN 972-8346-11-5, Cadernos do Caos, Porto, Portugal.

Mota, Carlos Alberto M. G.; Cruz, Maria Gabriel M. Bulas (2001). *História da Educação (com referência à Educação de Infância): Textos de Apoio*, ISBN 972-669-456-6, UTAD, Vila Real, Portugal.

Moura, Leonel; Pereira, Henrique Garcia (2004). *Man + Robots : Symbiotic Art*, ISBN 2905985674, Institut d'Art Contemporain, Lyon/Villeurbanne, France. A summary of these author's ideas can be found in Leonel Moura & Henrique Garcia Pereira (2004), "Symbiotic Art Manifesto: Making the Artists that make the Art", Web page at http://www.lxxl.pt/artsbot/index.html, retrieved on August 18th, 2005.

Mukherjee, Amitabha; Sharma, Gaurav; Prakash, Manu (2002). *Programming using Stackable Bricks*, in "Development by Design 2002 – Proceedings of the 2nd International Conference on Open Collaborative Design for Sustainable Innovation - December 1-2, 2002, Bangalore, India", under publication at ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://www.geocities.com/gashar1982/dyd_paper.pdf on October 6th, 2004.

Muller, Jim (1998). *The Great Logo Adventure: Discovering Logo on and Off the Computer*, ISBN 0965193462, Doone Publications, Madison, Alabama, USA. Referenced from the electronic version, retrieved from http://www.softronix.com/download/tgla.zip on February 9th, 2005.

Munro-Mavrias, Sandra (1983). *Computer Programming by Kindergarten Children Using LOGO*, paper presented at the Association for Media and Technology in Education in Canada/ADATE Confluence '83 (Montreal, Canada, June 21, l983), ED237066, ERIC – Education Resources Information Center, Computer Sciences Corporation, Lanham, MD, USA.

Myers, Brad A. (1989). *Taxonomies of Visual Programming and Program Visualization*, Journal of Visual Languages and Computing, vol. 1, no. 1, pp. 97-123, Mar. 1990, ISSN 1045-926X, Academic Press, Cambridge, UK. Referenced from the on-line version, dated September 20, 1989, retrieved from http://www-2.cs.cmu.edu/~bam/papers/vltax2.pdf on August 25th, 2004.

Myers, Brad A. (1993). *Demonstrational Interfaces: A Step Beyond Direct Manipulation*, in Allen Cypher (ed.), "Watch What I Do: Programming by Demonstration", ISBN 0-262-03213-9, ch. 26, pp. 485-512, MIT Press, Cambridge, Massachusetts, USA. (Cited from the second printing, 1994.)

NAEYC (1996). *Technology and Young Children—Ages 3 through 8: A position statement of the National Association for the Education of Young Children*, position statement, National Association for the Education of Young Children, Washington, D.C., USA. Referenced from the on-line version at http://www.naeyc.org/about/positions/pdf/PSTECH98.PDF, retrieved on January 26th, 2005.

Napper, Brian (1999). *Alan M. Turing (1912 - 1954)*, Web page at the Web site "*Computer 50: The University of Manchester Celebrates the Birth of the Modern Computer*", University of Manchester, Manchester, UK. Retrieved from http://www.computer50.org/mark1/turing.html on August 22nd, 2005.

Naur, Peter (ed.); Backus, J. W.; Wegstein, J. H.; van Wijngaarden, A.; Woodger, M.; Bauer, F. L.; Green, J; Katz, C.; McCarthy, J.; Perlis, A. J.; Rutishauser, H.; Samelson, K.; Vauquois, B. (1960). *Report on the algorithmic language ALGOL 60*, in "Communications of the ACM", ISSN 0001-0782, vol. 3, no. 5, pp. 299-314, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/367236.367262 on August 16th, 2005.

NCTM (2000). *Principles and Standards for School Mathematics*, ISBN 0-87353-480-8, National Council of Teachers of Mathematics, Reston, VA, USA. Referenced from the on-line version, retrieved from http://www.blaine.k12.wa.us/EStandards/Standards/document/index.htm on February 9th, 2005.

Negroponte, Nicholas; Resnick, Mitchel; Cassell, Justine (1997). *Creating a Learning Revolution*, Opinion Article 8, MIT, Cambridge, Massachusetts, USA. Referenced from the on-line version, retrieved from http://llk.media.mit.edu/papers/1997/opinion8.htm, on March 8th, 2001.

Nelson, Andrew (1983). *Creating Adventure Programs on Your Computer*, ISBN 0907563368, Interface Publications, London, UK. Referenced from the Portuguese translation by Conceição Jardim and Eduardo Nogueira, *Programar Aventuras no Seu Computador*, Editorial Presença, Lisbon, Portugal, 1984.

NETS (1999). *Performance Indicators for Technology-Literate Students: Grades PreK–2*, in "National Educational Technology Standards for Students – Connecting Curriculum & Technology", ISBN 1564841502, section 2, pp. 18-19, ISTE – International Society for Technology in Education, Washington, D.C., USA. Referenced from the on-line version, retrieved on August 10th, 2005, from http://cnets.iste.org/students/pdf/profilek2.pdf.

New Zealand Ministry of Education (2005). *Supporting Learning In Early Childhood Education Through Information And Communication Technologies: A Framework For Development*, ISBN 0-478-13260-3, Ministry of Education, Wellington, New Zealand. Referenced from the electronic version, which was retrieved on July 9th, 2005, from the URL address http://www.minedu.govt.nz/web/downloadable/dl10417_v1/foundationsfordiscoveryextendedversion.pdf.

New, Rebecca S. (2000). *Reggio Emilia: Catalyst for Change and Conversation*, ERIC Digest EDO-PS-00-15, December 2000, ERIC Clearinghouse on Elementary and Early Childhood Education, University of Illinois, Champaing, IL, USA. Referenced from the on-line version, retrieved from http://ceep.crc.uiuc.edu/eecearchive/digests/2000/new00.pdf on January 25th, 2005.

Nickerson, Jeffrey Vernon (1994). *Visual Programming*, Ph.D. Dissertation, New York University, (UMI# 9514409), New York, NY, USA. Referenced from the on-line version at http://www.stevens-tech.edu/jnickerson/, retrieved on August 6th, 2004.

Niza, Sérgio (1998). *O Modelo Curricular de Educação Pré-Escolar da Escola Moderna Portuguesa*, in "Modelos Curriculares para a Educação de Infância", Júlia Oliveira Formosinho (ed.), 2nd edition (1998), ISBN 972-0-34451-2, pp. 137-159, Porto Editora, Oporto, Portugal.

NMAH-BC (2002). *Slates, Slide Rules, and Software – Teaching Math in America*, an exhibit at the National Museum of American History, Behring Center, in Washington, USA. Cited from the exhibition's Web site at http://www.americanhistory.si.edu/teachingmath/index.htm, retrieved on July 2nd 2004.

Noll, Steven; Trent Jr., James W. (2004). *Introduction*, in "Mental Retardation in America: A Historical Reader", ISBN 0-814-78247-7, Steven Noll & James W. Trent Jr. (eds.), pp. 1-16, New York University Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://www.nyupress.org/webchapters/0814782477intro.pdf, on May 25th, 2005.

Norman, Donald A. *Cognitive engineering*, in D. A. Norman & S. W. Draper (eds.), "User centered systems design", pp. 31 – 61, ISBN 0898598729, Lawrence Erlbaum Associates Inc., Mahwah, NJ, USA.

Noss, Richard; Hoyles, Celia; Gurtner, Jean-Luc; Adamson, Ross; Lowe, Sarah (2002). *Face-to-face and online collaboration: appreciating rules and adding complexity*, in International Journal of Continuing Engineering Education and Lifelong Learning, vol. 12, no. 5/6, pp. 521-540, ISSN 1560-4624, Inderscience Publishers, Olney, Buckshire, England, UK. Referenced from the on-line

version at http://www.ioe.ac.uk/playground/RESEARCH/papers/online_collaboration.pdf, retrieved on July 2nd, 2004.

Nygaard, Kristen; Dahl, Ole-Johan (1978). *The Development of the SIMULA Languages*, in "The first ACM SIGPLAN conference on History of programming languages", ISSN 0362-1340, pp. 245-272, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/800025.808378.

Nyquist, John R. (2005). *Stagecast Creator and preschoolers*, personal e-mail message sent on August 4th, 2005, at 6:37 PM. Full text is provided in Annex II.

Oberlin, Steven D. (2004). *Johann Friedrich Oberlin*, Web page retrieved on May 3rd, 2005, from http://www.oberlins.com/ob_john_f.htm.

Office of Technology Assessment (1995). *Teachers and technology: Making the connection*, OTA-HER-616, Office of Technology Assessment, U.S. Congress, U.S. Government Printing Office, Washington, D.C., USA. Referenced from the on-line version, retrieved on July 27th, 2005, from http://www.wws.princeton.edu/cgi-bin/byteserv.prl/~ota/disk1/1995/9541/9541.PDF.

Oliveira, Martha Kohl de (1997). *Vygotsky: Aprendizado e desenvolvimento: Um processo sócio-histórico*, ISBN 85-262-1936-7, Editora Scipione, São Paulo, Brazil.

Ong, James; Ramachandran, Sowmya (2003). *Intelligent Tutoring Systems: Using AI to Improve Training Performance and ROI*, Technical Document, Stottler Henke Associates, San Mateo, CA, USA. Referenced from the on-line version, retrieved on August 18th, 2005, from http:// www.stottlerhenke.com/papers/ITS_using_AI_to_improve_training_performance_and_ROI.pdf.

Oppenheimer, Todd (2003). *Meet The Characters*, Web page of the companion site to the book "The Flickering Mind: The False Promise of Technology in the Classroom and How Learning Can Be Saved", retrieved from http://www.booknoise.net/flickeringmind/characters/ on July 27th, 2004.

Overmars, Mark (2004). *Designing Games with Game Maker*, software documentation available on-line at http://www.gamemaker.nl/doc/gmaker.pdf. Retrieved on September 30th, 2004.

Overmars, Mark (no date). *Drawing Programming Environment*, Web page retrieved from http://www.cs.uu.nl/people/markov/kids/drape.html, on October 4th, 2004, Instituut voor Informatica/Informatiekunde, Universiteit Utrecht, The Netherlands.

Owen, Robert (1812). *Essays on the Formation of the Human Character*, 4 essays published in 9 weekly installments, W. Strange, London, UK. Republished in 1813/1814 as "A new view of society, or essays on the principle of the human character and the application of the principle of practice", Cadell & Davies, London, UK. Referenced from the on-line version, retrieved on May 4th, 2005, from http://www.eco.utexas.edu/facstaff/Cleaver/368owennewviewtable.pdf.

Ozzie, Ray (2002), *Speaking Mind to Mind*, article in "The New York Times", Business section, December 1st, 2002, written with Glenn Rifkin. Referenced from the on-line version at http://www.nytimes.com/2002/12/01/business/yourmoney/01BOSS.html?ex=1091332800&en=155 bc2ae7014f7c7&ei=5070, retrieved on July 30th, 2004.

Pane, John F. (2002), *A Programming System for Children that is Designed for Usability*, Ph.D. Thesis, Carnegie Mellon University, Computer Science Department, CMU-CS-02-127, Pittsburgh, PA, USA. Referenced from the on-line version, retrieved on October 21st, 2004, from http://www-2.cs.cmu.edu/~pane/thesis/.

Papalia, Diane E.; Olds, Sally Wendkos; Feldman, Ruth Duskin (1999). *A Child's World: Infancy Through Adolescence*, eight edition, ISBN 0-07-048785-5, McGraw-Hill, New York, NY, USA. Referenced from the Portuguese translation, "O Mundo da Criança", ISBN 972-773-069-8, translated by Isabel Soares, Alice Bastos, Carla Martins, Inês Jongenelen, Orlanda Cruz, and Teresa Gonçalves, published in 2001 by McGraw-Hill de Portugal, Lisbon, Portugal.

Papert, Seymour (1977). MIT Logo Project meeting notes, referenced by Ken Kahn (2001).

Papert, Seymour (1980). *Mindstorms: Children, Computers, and Powerful Ideas*, ISBN 0-465-04629-0, Basic Books, New York, USA. Referenced from the second edition, ISBN 0-465-04674-6, Basic Books, New York, USA, 1993.

Papert, Seymour (1993). *The Children's Machine: rethinking school in the age of the computer*, ISBN 0-465-01063-6, Basic Books, New York, USA, 1993.

Papert, Seymour (1998). *Technology in Schools: To Support the System or Render it Obsolete*, on-line article published by the Milken Exchange on Education Technology, Milken Family Foundation, Santa Monica, California, USA. Retrieved on March 19th, 2004, from http://www.mff.org/edtech/article.taf?_function=detail&Content_uid1=106.

Papert, Seymour (1999). *Introduction: What is Logo? And Who Needs It?*, in "Logo Philosophy and Implementation", ISBN 2-89371-494-3, LCSI, Highgate Springs, Vermont, USA. Electronic book retrieved from http://www.microworlds.com/company/philosophy.pdf on March 30th, 2004.

Paper, Seymour (1999). *Jean Piaget - He found the secrets of human learning and knowledge hidden behind the cute and seemingly illogical notions of children*, in "Time", ISSN 0040-781X, vol. 153, no. 12, Time Inc., New York, NY, USA. Referenced from the on-line version, retrieved from http://www.time.com/time/time100/scientist/profile/piaget.html on August 20th, 2005.

Papert, Seymour; Harel, Idit (1991). *Situating Constructionism*, in "Constructionism", ISBN 0893917869 , Ablex Publishing Corporation, Norwood, NJ, USA. Referenced from the on-line version, retrieved from http://www.papert.org/articles/SituatingConstructionism.html on August 6th, 2004.

Papert, Seymour; Solomon, Cynthia (1971). *Twenty Things To Do With A Computer*, Logo Memo no. 3, Artificial Intelligence Memo no. 248, MIT AI Lab, Massachussets Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved on August 4th, 2005, from http://hdl.handle.net/1721.1/5836.

Paz, Alexandra Maria Serpa Melo da (2004). *Software Educativo Multimédia no Jardim de Infância – Actividades Preferidas pelas Crianças dos 3 aos 5 Anos*, Master's dissertation, Instituto de Educação e Psicologia, Universidade do Minho, Braga, Portugal. Referenced from the on-line version, retrieved from http://hdl.handle.net/1822/921 on May 11th, 2005.

PBS, Public Broadcasting System (1998). *People and Discoveries – Ivan Pavlov, 1849-1936*, Web page retrieved from http://www.pbs.org/wgbh/aso/databank/entries/bhpavl.html on July 7th, 2005, Public Broadcasting Service, Alexandria, VA, USA.

Pea, Roy D. (1986). *Language-independent conceptual "bugs" in novice programming*, in "Journal of Educational Computing Research", ISSN 0735-6331, vol. 2, no. 1, pp. 25–36., Baywood Publishing Co., Amityville, NY, USA.

Peixoto, Daniel; Carvalho, José Luís (1990). *O Teclado de Conceitos na Educação*, in "BICA – Boletim informativo do centro de recursos – Interactividade, Comunicação, Aprendizagem", vol. 90/91, no. 1, pp. 23-26, Cnotinfor, Coimbra, Portugal.

Perlis, Alan J.; Samelson, Klaus (1958). *Preliminary report: international algebraic language*, in "Communications of the ACM", ISSN 0001-0782, vol. 1, no. 12, pp. 8-12, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/377924.594925.

Perlman, Radia (1974). *TORTIS – Toddler's Own Recursive Turtle Interpreter System*, MIT AI Memo 311, Logo Memo 9, Massachusetts Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved on October 25th, 2004 from ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-311.pdf.

Perlman, Radia (1976). *Using computer technology to provide a creative learning environment for preschool children*, MIT AI Lab Memo 360, Logo Memo 24, Massachusetts Institute of Technology, Cambridge, MA, USA. There is an on-line version available at ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-360.pdf (last checked on October 25th, 2004).

Perlman, Radia (2005). *Re: Computer Programming in Preschool*, personal e-mail message sent to Leonel Morgado (leonelm@utad.pt) on May 26th, 2005, at 18:22. Full text is provided in Annex II.

Perlmutter, Marion; Behrend, Stephanie Danell; Kuo, Frances; Muller, Alexandra (1989). *Social influence on children's problem solving*, in "Developmental Psychology", ISSN 0012-1649, vol. 25, no. 5, pp. 744-754, American Psychological Association, Washington, DC, USA.

Perry, Bruce D. (2002). *Childhood Experience and the Expression of Genetic Potential: What Childhood Neglect Tells Us About Nature and Nurture*, in "Brain and Mind", ISSN 1389-1987, vol. 3, no. 1, pp. 79-100, Springer Science+Business Media B.V., Berlin, Germany. Referenced from the on-line version, retrieved from http://www.childtrauma.org/ctamaterials/MindBrain.pdf on August 13th, 2005.

Perry, Bruce D. (2001). *The Neuroarcheology of Childhood Maltreatment: The Neurodevelopmental Costs of Adverse Childhood Events*, in K. Franey, R. Geffner, R. Falconer (eds.), "The Cost of Child Maltreatment: Who Pays? We All Do", ISBN 1882948122, Family Violence & Sexual Assault Institute, San Diego, CA, USA. Referenced from the on-line version, retrieved from http://www.childtrauma.org/ctamaterials/Neuroarcheology.asp on August 13th, 2005.

Perry, Bruce D.; Pollard, Ronnie (1997). *Altered brain development following global neglect in early childhood*, in "Proceedings from Annual Meeting, New Orleans, 1997", Society For Neuroscience, Washington DC, USA. Referenced from the on-line version, retrieved on August 13th, 2005, from http://www.childtrauma.org/CTAMATERIALS/neuros~1.asp.

Piaget, Jean (1957). *The Significance of John Amos Comenius at the Present Time*, in "John Amos Comenius 1592-1670. Selections.", pp. 11-31, UNESCO, Paris, France. Referenced from the 1993 version, *Jan Amos Comenius*, in "PROSPECTS: quarterly review of comparative education", ISSN 0033-1538, vol. 23, no. 1/2, pp. 173-196, UNESCO International Bureau of Education, Geneva, Switzerland. Last accessed on-line on May 4th, 2005, at the address http://www.ibe.unesco.org/International/Publications/Thinkers/ThinkersPdf/comeniuse.PDF.

Piaget, Jean (1962). *Commentaire sur les remarques critiques de Vygotski concernant 'Le langage et la pensée chez l'enfant' et 'Le jugement et le raisonnement chez l'enfant'*, manuscript translated into English by Leslie Smith and published as "Commentary on Vygotsky's criticisms of *Language and thought of the child* and *Judgement and reasoning in the child*", in L.S. Vygotsky, "Thought and Language", MIT Press, Cambridge, MA, USA. Referenced from the version in the magazine "New Ideas in Psychology", ISSN 0732-118X, vol. 18, no. 2-3, pp. 241-259, Elsevier B.V., Amsterdam, Netherlands, 2000, retrieved on-line on July 16th, 2005, from http://www.ssc.uwo.ca/psychology/undergraduate/psych532b/piaget_2000.pdf.

Piaget, Jean (1971). *The science of education and the psychology of the child*, ISBN 0-670-62172-2, The Viking Press, New York, NY, USA.

Piaget, Jean (1974). *To understand is to invent*, ISBN 0-670-00577-0, The Viking Press, Penguin, New York, NY, USA.

Piaget, Jean (1975). *L'équilibration des structures cognitives: problème central du développement*, Presses Universitaires de France, Paris. English version: "The equilibration of cognitive structures: the central problem of intellectual development", ASIN 0226667812, University of Chicago Press, Chicago, IL, USA, 1985. Portuguese version: "O Desenvolvimento do Pensamento – Equilibração das Estruturas Cognitivas", Dom Quixote, Lisbon, 1977.

Pierce, Patsy L. (1994). *Technology Integration into Early Childhood Curricula: Where We've Been, Where We Are, Where We Should Go*, in "Research Synthesis on Early Intervention Practices", ch. 3, on-line technical report, National Center to Improve the Tools of Educators, College of Education, University of Oregon, Eugene, OR, USA. Retrieved on January 26th, 2005, from http://idea.uoregon.edu/~ncite/documents/techrep/tech11-3.html.

Pietri, Charles (1981). *Les origines de la 'pédagogie': Grèce et Rome*, in Gaston Mialaret & Jean Vial (eds.), "Histoire mondiale de l'éducation", ISBN 2-130-36776-3, vol. 1, Presses Universitaires de France, Paris, France. Referenced from the Portuguese translation by Evaristo Santos, *As Origens da «Pedagogia»: Grécia e Roma*, in (n.d.) "História Mundial da Educação", ISBN 972-703-067-X, vol. 1, pp. 129-200, RÉS-Editora, Porto, Portugal.

Piper, Gil (2000). *The use of roamer as a constructivist tool in early learning*, ICT Resources, Postgraduate Certificate of Education (PGCE), School of Education, University of Southampton, Southampton, UK. Referenced from the on-line version, retrieved on August 9th, 2005, from http://www.pgce.soton.ac.uk/ict/roamer/.

Plann, Susan (1997). *A Silent Minority: Deaf Education in Spain, 1550-1835*, ISBN 0-520-20471-9, University of California Press, Berkeley, CA, USA. Referenced from the on-line edition, retrieved from http://ark.cdlib.org/ark:/13030/ft338nb1x6/ on May 26th, 2005.

Plato Learning, Inc. (2004). *3 Decoders*, promotional video in digital format describing the software application on-line at http://www.plato.com/downloads/movies/3_DECODERS_300.mov, retrieved on July 30th, 2004.

Plato Learning, Inc. (no date). *History*, Web page retrieved on July 30th, 2004, from http://www.plato.com/aboutus/company_history.asp.

Playground Project (2001). *3rd Annual Report – September 2001*, on-line collection of documents at www.ioe.ac.uk/playground/RESEARCH/reports/finalreport/, retrieved on July 2nd, 2004.

Plowman, Lydia; Stephen, Christine (2003). *A 'benign addition'? Research on ICT and pre-school children*, in "Journal of Computer Assisted Learning", ISSN 0266-4909, vol. 19, no. 2, pp. 149-164, Blackwell's, Oxford, UK. Referenced from the on-line version, retrieved on March 5th, 2004, from http://www.ioe.stir.ac.uk/CACHET/Docs/JCAL_benign_addition.PDF. An expanded version was previously published in 2002 as "*ICT in Pre-School: A 'Benign Addition'? A review of the literature on ICT in pre-school settings*", ISBN 1-85955-772-4, Learning and Teaching Scotland, Glasgow, Scotland, UK. Referenced from the on-line version, retrieved on July 27th, 2005, from http://www.ltscotland.org.uk/earlyyears/files/benignaddition.pdf.

Plowman, Lydia; Stephen, Christine (2005). *Children, play, and computers in pre-school education*, in "British Journal of Educational Technology", ISSN 0007-1013, vol. 36, no. 2, Blackwell Publishing, Oxford, UK.

Plucker, Jonathan (2003). *Jean-Marc Gaspard Itard*, Web page at the site "Human Intelligence", Indiana University, Bloomington, IN, USA. Retrieved on May 25th, 2005, from http://www.indiana.edu/~intell/itard.shtml.

Poole, Steven (2000). *Trigger Happy: Videogames and the Entertainment Revolution*, ISBN 1559705396, Arcade Publishing, New York, NY, USA.

Portsmore, M., Cyr, M.; Rogers, C. (2001), *Integrating the Internet, LabVIEW, and Lego Bricks Into Modular Data Acquisition and Analysis Software for K-College*, in "Computers in Education Journal", vol. 11, no. 2, Computers in Education Division, American Society for Engineering Education, Port Royal, VA, USA. Referenced from the on-line version, retrieved from http://www.asee.org/acPapers/20334.pdf on August 1st, 2005.

Potter, Richard (1993). *Just-in-Time Programming*, in Allen Cypher (ed.), "Watch What I Do: Programming by Demonstration", ISBN 0-262-03213-9, pp. 514-515, MIT Press, Cambridge, Massachusetts, USA. (Cited from the second printing, 1994.)

PPIG (2004). *State of the PPIG*, Web page at http://www.ppig.org/ppigstate/, part of teh site "Psychology of Programming Interest Group", administered by Judith Segal, The Open University, Milton Keynes, UK. Retrieved on June 10th, 2005.

Pressey, Sidney L. (1926). *A simple apparatus which gives tests and scores – and teaches*. School and Society, 23 (586), 373-376.

Quivy, Raymond; Campenhoudt, Luc Van (1995). *Manuel de recherche en sciences sociales*, 2nd edition, ISBN 2100026569, Dunod, Paris, France. Referenced from the Portuguese translation by João Minhoto Marques, Maria Amália Mendes & Maria Carvalho (1998), "Manual de Investigação em Ciências Sociais", ISBN 972-662-275-1, Gradiva, Lisbon, Portugal.

Rader, Cindy; Brand, Cathy; Lewis, Clayton (1997). *Degrees of Comprehension: Children's Understanding of a Visual Programming Environment*, in "Proceedings of the SIGCHI conference on Human factors in computing systems", ISBN 0-89791-802-9, pp. 351-358, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on January 21st, 2001, from http://doi.acm.org/10.1145/258549.258793.

Raffle, Hayes Solos (2004). *Topobo: A 3-D Constructive Assembly System with Kinetic Memory*, Master's dissertation, School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved from http://web.media.mit.edu/~hayes/topobo/Raffle_MS_Thesis_small.pdf on August 2nd, 2005.

Raffle, Hayes Solos; Garcia, Cristóbal (2003). *topobo for Tangible Learning*, on-line article at http://web.media.mit.edu/~hayes/topobo/tangible-topobo.pdf, Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA. Retrieved on August 2nd, 2005.

Raffle, Hayes Solos; Parkes, Amanda J.; Ishii, Hiroshi (2004). *Topobo: a constructive assembly system with kinetic memory*, in "Proceedings of the 2004 conference on Human factors in computing systems", pp. 647-654, ISBN 1-58113-702-8, ACM Press, New York, NY, USA. Referenced from the on-line version at http://web.media.mit.edu/~hayes/topobo/Topobo-CHI.pdf, retrieved on November 8th, 2004.

Ramirez, Vilma Contreras; Quirós, Ingrid Morales; Hernández, Ana Polanco; Moya, Ingrid Trejos; Cordero, Marlene Zúñiga (1989). *El Uso de la microcomputadora en la educacion de ninos preescolares de cinco anos seis meses a seis anos seis meses, en instituciones publicas y privadas de la provincia de San Jose*, Seminario de graduacion, Licenciatura en Educacion Preescolar, Faculty of Education, Teacher Training School, University of Costa Rica, San José, Costa Rica.

Reilly, Rob (2002). *What Are We Aiming At--What Do We Really Want To Aim At?*, in "Teachers.Net Gazette" (January 2002), vol. 3, number 1, WWW-based magazine, Teachers.Net, San Diego, CA, USA. Retrieved from http://www.teachers.net/gazette/JAN02/reilly.html on February 9th, 2005.

Reimer, G. (1985). *Effects of a Logo computer programming experience on readiness for first grade, creativity, and self concept: A pilot study in kindergarten*, in "The Monitor", ISSN 0886-3695, vol. 23, no. 7 & 8, pp. 8-12, The Association for Educational Data Systems, Washington, D.C., USA.

Reiser, Robert A. (2001).*A History of Instructional Design and Technology: Part II: A History of Instructional Design*, in Educational Technology Research and Development, Vol. 49, No. 2, 2001, pp. 57–67, ISSN 1042–1629. Referenced from the electronic version, retrieved from http://www.aect.org/pdf/etr&d/4902/4902-04.pdf on June 30th, 2004.

Repenning, Alexander; Perrone, Corrina (2001). *Programming by Analogous Examples*, in Henry Lieberman (ed.), "Your Wish Is My Command: Programming By Example", ISBN 1-55860-688-2, Morgan Kaufmann Publishers, San Francisco, California, USA.

Resnick, Mitchel (1990). *MultiLogo: A Study of Children and Concurrent Programming*, Interactive Learning Environments, vol. 1, no. 3, ISSN 1049-4820, Swets & Zeitlinger Publishers, Lisse, The Netherlands. On-line version retrieved on January 12th, 2001, from http://el.www.media.mit.edu/groups/el/Papers/mres/MultiLogo/MultiLogo.html.

Resnick, Mitchel (1997). *Turtles, Termites, and Traffic Jams*, ISBN 0-262-68093-9, MIT Press, Cambridge, Massachusetts, USA.

Resnick, Mitchel (1998). *Technologies for Lifelong Kindergarten*, Educational Technology Research and Development, vol. 46, no. 4, ISSN 1042-1629, Association for Educational Communications and Technology, Bloomington, Indiana, USA.

Resnick, Mitchel; Silverman, Brian (1997). *Turtles*, in "Going in Circles", active essay on-line at http://lcs.www.media.mit.edu/groups/el/projects/circles/, Epistemology and Learning Group, MIT Media Laboratory, MIT, Cambridge, MA, USA. Retrieved on September 9th, 2004, from http://lcs.www.media.mit.edu/groups/el/projects/circles/turtles.html.

Reyes, Martin (2000). *Return of the Incredible Machine: Contraptions – Sierra's patented recipe for toast: A hamster, two rubber bands, a pulley, and multicolored lasers*, on-line review at http://pc.ign.com/articles/156/156754p1.html, ign.com, IGN Entertainment Incorporated, Palatine, IL, USA. Retrieved on November 15th, 2004.

Rheta DeVries (2000). *Vygotsky, Piaget, and Education: A Reciprocal Assimilation of Theories and Educational Practices*, in "New Ideas in Psychology", ISSN 0732-118X, vol. 18, no. 2-3, pp. 187-213, Elsevier B.V., Amsterdam, Netherlands, 2000. Referenced from the on-line version at http://www.uni.edu/coe/regentsctr/Publications/Vygotsky%20Piaget%20and%20Edu.pdf, retrieved on July 16th, 2005.

Rice, Alex (2004). *Re: Exocet dreams; not the missile but...*, message sent to the "use-revolution" mailing list on February 12th, 2004, at 17:25:32 EST, available on-line at http://lists.runrev.com/pipermail/use-revolution/2004-February/031430.html. Retrieved on August 1st, 2005.

Richardson, Jeff (no date). *Programming Projects by young children*, Web page at http://users.bigpond.net.au/deasey/kinder.html, retrieved on July 2nd, 2004.

Ritchie, Dennis M. (1993). *The Development of the C Language*, in "The second ACM SIGPLAN conference on History of programming languages", ISBN 0-89791-570-4, pp. 201-208, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/154766.155580.

Riverdeep (no date). *Critical Thinking – Thinkin' Things Collection 3*, online product brochure at http://www.riverdeep.net/products/critical_thinking/ttc3.jhtml, Riverdeep Interactive Learning Limited, San Francisco, CA, USA. Retrieved on October 3rd, 2004.

Robertson, Andrew (2002). *Pac-Man*, Web site at http://www.pacman.i-p.com (alternative address: http://www.red-gallery.com/pacman/). Retrieved on August 11th, 2004.

Robinett, Warren (no date). *Rocky's Boots – for the Apple II and PC*, Web page at http://www.warrenrobinett.com/rockysboots/. Retrieved on November 15th, 2004.

Rogers, Chris (2003). *Engineering Education with LEGO Bricks: Kindergarten meets College*, on-line abstract of a colloquium held on March 13th, 2003, at the Department of Computer Science, School of Engineering, University of Virginia, Charlottesville, Virginia, USA. Retrieved from http://www.cs.virginia.edu/colloquia/event281.html on August 1st, 2005.

Rogoff, Barbara; Turkanis, Carolyn Goodman; Bartlett, Leslee (2001). *Learning Together: Children and Adults in a School Community*, ISBN 0-19-509753-X, Oxford University Press, New York, NY, USA.

Röhrs, Hermann (1994). *Maria Montessori (1870-1952)* in "PROSPECTS: quarterly review of comparative education", ISSN 0033-1538, vol. 24, no. 1/2, pp. 89-90, UNESCO International Bureau of Education, Geneva, Switzerland. Referenced from the on-line version, retrieved from http://www.ibe.unesco.org/International/Publications/Thinkers/ThinkersPdf/owene.PDF on May 25th, 2005.

Rose, Kim (2005). *Re: [Squeakland] Squeak Etoys and preschoolers*, personal e-mail message sent on August 4th, 2005, at 4:44 PM. Full text is available in Annex II.

Ross, Douglas T. (1978). *Origins of the APT language for automatically programmed tools*, in "The first ACM SIGPLAN conference on History of programming languages", ISSN 0362-1340, pp. 61-99, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/800025.808378.

Rube Goldberg Incorporated (no date). *Rube Goldberg Biography*, Web page at http://www.rube-goldberg.com/html/bio.htm, maintained by Rube Goldberg Inc., Herndon, VA, USA. Retrieved on November 15th, 2004.

Sadiku, Matthew N. O.; Obiozor, Clarence N. (1997). *Evolution of Computer Systems*, in "Technology-Based Re-Engineering Engineering Education: Proceedings of Frontiers in Education Fie '96 26th Annual Conference: November 6-9, 1996 Salt Lake City, Utah", Fie'96 Program Co-Chairs and Organizing Committee (ed.), ISBN 0780333489, IEEE, Piscataway, New Jersey, USA. Referenced from the on-line version, retrieved from http://fie.engrng.pitt.edu/fie96/papers/434.pdf on April 29th, 2004.

Sagan, Carl (1983). *Cosmos*, ISBN 0-394-71596-9, Random House, New York, NY, USA.

Sammet, Jean E. (1972). *Programming languages: history and future*, in "Communications of the ACM", ISSN 0001-0782, vol. 15, no. 7, pp. 601-610, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/361454.361485 on August 16th, 2005.

Sammet, Jean E. (1978). *The early history of COBOL*, in "The first ACM SIGPLAN conference on History of programming languages", ISSN 0362-1340, pp. 121-161, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/800025.808378.

Santomé, Jurjo Torres (1991). *A Educação Infantil*, Editora Movimento de Renovação Pedagógica Associaçâo Sócio-Pedagógica Galaico-Portuguesa, Ourense, Spain.

Santos, Arquimedes (1999). *Breve NOTA acerca de Psicologia e Jogo*, in "Cadernos de educação de infância", no. 49, Associação dos Profissionais de Educação de Infância, Lisbon, Portugal.

Saraswat, Vijay A. (1993). *Concurrent Constraint Programming*, ISBN 0-262-19297-7, MIT Press, Cambridge, Massachusetts, USA.

Schement, Jorge Reina (2002). *From Castle to Node: The Changing Information Environment of the American Home*, presentation given on June 6th, 2002, Institute for Information Policy of the Penn State University. Referenced from the on-line version, retrieved on August 4th, 2004, from http://www.webuse.umd.edu/abstracts2002/20th_Cent._&_Households.pdf.

Schwartz, Sydney L. (1985). *Kindergartners Having Microcomputer Experiences: A Descriptive Study*, paper presented at the Annual Meeting of the American Educational Research Association (Chicago, IL, March 31-April 4, 1985), ED277456, ERIC – Education Resources Information Center, Computer Sciences Corporation, Lanham, MD, USA.

Science Museum (2004). *babbage – Analytical Engine*, Science Museum, London, UK. Electronic article retrieved from http://www.sciencemuseum.org.uk/on-line/babbage/page5.asp on April 28th, 2004.

Scrivener, Alan B. (1994). *The Impact of Visual Programming in Medical Research*, in "Medicine Meets Virtual Reality II: Interactive Technology & Healthcare: Visionary Applications for Simulation, Visualization, and Robotics", p. 8, IOS Press, Amsterdam, The Netherlands. Referenced from the on-line version at http://www.well.com/~abs/impact_vis_med.html, retrieved on August 25th, 2004.

Sedighian, Kamran; Sedighian, Andishe (1997). *Aesthetic Response: Children's Reactions to Color and Graphics in Educational Software*. In "Proceedings of ED-MEDIA/ED-TELECOM'97", ISBN 1-880094-26-6, AACE – Association for the Advancement of Computing in Education, Norfolk, VA, USA.

Seger, Eliane; Verhoeven, Ludo (2002). *Multimedia support of early literacy learning*, in "Computers & Education", ISSN , vol. 39, issue 3, pp. 207-211, Elsevier Science, Essex, UK.

Shapiro, Ehud (1989). *The Family of Concurrent Logic Programming Languages*, ACM Computing Surveys, vol. 21, no. 3, September, 1989, pp. 413-510, ISSN 0360-0300, Association for Computing Machinery, New York, NY, USA. Referenced from the electronic version, retrieved from http://doi.acm.org/10.1145/72551.72555 on January 21st, 2001.

Shaw, Peter (1985). *The Generation Game*, in "Your Spectrum", issue 17, August 1985, ISSN 0269-6983, Dennis Publishing, London, UK. An on-line version is available at http://www.users.globalnet.co.uk/~jg27paw4/yr17/yr17_24.htm (confirmed on September 30th, 2004).

Sheehan, Robert (2003). *Children's perception of computer programming as an aid to designing programming environments*, in "Proceeding of the 2003 conference on Interaction design and children", ISBN 1-58113-732-X, pp. 75-83, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/953536.953548, retrieved on September 30th, 2004.

Sheehan, Robert (2004). *The Icicle programming environment*, in "*Interaction Design And Children – Proceeding of the 2004 conference on Interaction design and children: building a community*", ISBN 1-58113-791-5, pp. 147-148, ACM Press, New York, NY, USA. Referenced from the online version, retrieved from http://doi.acm.org/10.1145/1017833.1017863 on May 18th, 2005.

Shneidermann, Ben (2001). *Foreword*, in Henry Lieberman (ed.), "Your Wish Is My Command: Programming By Example", ISBN 1-55860-688-2, Morgan Kaufmann Publishers, San Francisco, California, USA.

Shoham, Yoav (1993). *Agent-oriented programming*, Artificial Intelligence, vol. 60, issue 1, pp. 51-92, ISSN 0004-3702, Elsevier Science Publishers Ltd., Essex, UK.

Shoji, Masako (n.d.) *A Educação no Japão Antigo e Medieval*, in "História Mundial da Educação", ISBN 972-703-067-X, vol. 1, pp. 117-126, RÉS-Editora, Porto, Portugal. This volume is a translation by Evaristo Santos of Gaston Mialaret & Jean Vial (eds.), 1981, "Histoire mondiale de l'éducation", ISBN 2-130-36776-3, vol. 1, Presses Universitaires de France, Paris, France.

Silva, Maria Isabel R. Lopes da (1996). *Práticas educativas e construção de saberes: Metodologias da investigação-acção*, ISBN 972-9380-85-6, Instituto de Inovação Educacional, Ministry of Education, Lisbon, Portugal.

Siraj-Blatchford, John; Siraj-Blatchford, Iram (2004). *IBM KidSmart Early Learning programme European Evaluation: France, Germany, Italy, Portugal, Spain and UK – Final Report June 2004*, IBM Corporation, White Plains, NY, USA. Referenced from the on-line version,

retrieved from http://www.ibm.com/ibm/ibmgives/downloads/kidsmart_eval_full_rep_English.pdf on July 29th, 2005.

Skeele, Rosemary; Stefankiewicz, Gretchen (2002). *Blackbox in the Sandbox: The Decision to Use Technology with Young Children With Annotated Bibliography of Internet Resources for Teachers of Young Children*, in "Educational Technology Review", ISSN 1065-6901, vol. 10, no. 2, pp. 79-95, Association for the Advancement of Computing in Education, Norfolk, VA, USA. Referenced from the on-line version, retrieved from http://www.aace.org/pubs/etr/issue3/skeele.pdf on January 28th, 2004.

Skinner, B. F. (1968). *The Technology of Teaching*, Prentice-Hall, Englewood Cliffs, NJ, USA. Referenced from the on-line version, retrieved on August 22nd, 2004, from http://www.cedu.niu.edu/tutortechlab/history/machine.html.

Sleeman, D.; Putnam, Ralph T.; Baxter, Juliet; Kuspa, Laiani (1986). *Pascal and high school students: a study of errors*, in "Journal of Educational Computing Research", ISSN 0735-6331, vol. 2, no. 1, pp. 57-73, Baywood Publishing Co., Amityville, NY, USA. Referenced from the version found on-line at http://www.cmi.hku.hk/hclam/mite6218/sleeman.pdf, retrieved on August 12th, 2004.

Smarthome (2005). *Smarthome Web site*, SMARTHOME Inc., Irvine, CA, USA. Last accessed at http://www.smarthome.com, on August 18th, 2005.

Smith, David Canfield; Cypher, Allen; Schmucker (1996). *Making programming easier for children*, in "Interactions", ISSN 1072-5520, vol. 3, no. 5, pp. 58 – 67, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/234757.234764, retrieved on September 20th, 2004.

Smith, David Canfield; Cypher, Allen; Spohrer (1994). *KidSim: programming agents without a programming language*, in "Communications of the ACM", ISSN 0001-0782, vol. 37, no. 7, pp. 54-67, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on January 21st, 2001, from http://doi.acm.org/10.1145/176789.176795.

Smith, David Canfield; Cypher, Allen; Tesler, Larry (2001). *Novice Programming Comes of Age*, in Henry Lieberman (ed.), "Your Wish Is My Command: Programming By Example", ISBN 1-55860-688-2, ch. 1, pp. 7-19, Morgan Kaufmann Publishers, San Francisco, California, USA.

Smith, Mark K. (1997). *fredrich froebel (fröbel)*, in "the encyclopedia of informal education", Web page retrieved on May 24th, 2005, from http://www.infed.org/thinkers/et-froeb.htm, last updated on January 28th, 2005.

Smith, Mark K. (1999). *learning theory*, in "the encyclopedia of informal education", Web page retrieved on July 6th, 2005, from http://www.infed.org/biblio/b-learn.htm, last updated on January 30th, 2005.

Smith, Mark K. (2002). *Jerome Bruner and the process of education*, in "the encyclopedia of informal education", Web page at http://www.infed.org/thinkers/bruner.htm (last updated on January 28th, 2005), retrieved on July 17th, 2005.

Smith, Mark K. (2003). *communities of practice*, in "the encyclopedia of informal education", Web page at http://www.infed.org/biblio/communities_of_practice.htm, last updated on January 30th, 2005, retrieved on July 9th, 2005.

Smith, Randall B. (1986). *Experiences with the alternate reality kit: an example of the tension between literalism and magic*, in "Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface", ISSN 0736-6906, pp. 61-67, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on November 15th, 2004, from http://doi.acm.org/10.1145/29933.30861.

Solomon, Cynthia (1986). *Computers environments for children*, ISBN 0-262-19249-7, MIT Press, Cambridge, Massachusetts, USA.

Solomon, Cynthia (1996). *Origins of Educational Multimedia Environments*, in Allison Druin & Cynthia Solomon (eds.), "Designing Multimedia Environments for Children", ISBN 0-471-11688-2, ch. 1, John Wiley & Sons, Inc., New York, NY, USA.

Solomon, Cynthia; Papert, Seymour (1975). *Teach: A Step Toward More Interactive Programming*, Logo Working Paper 43, MIT AI Lab, Massachussets Institute of Technology, Cambridge, MA, USA.

Soprunov, Sergei F. (1996). *ПервоЛого: Пособие для учителей* ("PervoLogo: Benefits for the teachers"), INT, Moscow, Russia.

Spaggiari, Sergio; Rinaldi, Carla (2000). *Catalogue of the exhibit The Hundred Languages of Children*, ISBN 88-87960-08-9, Reggio Children, Reggio Emilia, Italy.

Specht, Jacqueline; Wood, Eileen; Willoughby, Teena (2002). *What Early Childhood Educators Need to Know About Computers in Order to Enhance the Learning Environment*, in "Canadian Journal of Learning and Technology", ISSN 1499-6677, vol. 28, no. 1, Faculty of Education, University of Calgary, Calgary, Alberta, Canada. Referenced from the on-line version, retrieved from http://www.cjlt.ca/content/vol28.1/specht_etal.html on July 27th, 2005.

Spodek, Bernard; Brown, Patricia Clark (1993). *Curriculum Alternatives in Early Childhood Education: A Historical Perspective*, in Bernard Spodek (ed.), "Handbook of Research on the Education of Young Children", ISBN 0-028-97405-0, pp. 91-104, MacMillan, New York, NY, USA. Referenced from the Portuguese translation, *Alternativas Curriculares na Educação de Infância: Uma Perspectiva Histórica*, in Júlia Oliveira Formosinho (ed.), "Modelos Curriculares para a Educação de Infância", 2nd edition, 1998, ISBN 972-0-34451-2, pp. 13-50, Porto Editora, Oporto, Portugal.

Spodek, Bernard; Saracho, Olivia N. (1993). *Right from the start: teaching children ages three to eight*, ISBN 0-205-15281-3, Allyn and Bacon, Needham Heights, Massachusetts, USA. Referenced from the Portuguese translation by Cláudia Oliveira Dornelles, *Ensinando Crianças de Três a Oito Anos*, ISBN 85-7307-436-1, Editora Artes Médicas Sul, Ltda., Porto Alegre, Rio Grande do Sul, Brazil, 1998.

Spohrer, James C.; Soloway, Elliot (1986), *Novice Mistakes: Are The Folk Wisdoms Correct?*, in "Communications of the ACM", ISSN 0001-0782, vol. 29, no. 7, pp. 624-632, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on June 10th, 2005, from http://doi.acm.org/10.1145/6138.6145.

Squeakland (no date). *Drive A Car – The first Squeak Etoys project for children. How they design, build, get to tun, and learn to drive their car*, Web page available at http://www.squeakland.org/school/drive_a_car/html/Drivecar12.html, retrieved on September 14th, 2004.

Squire, Kurt D. (2004). *Replaying History: Learning World History Through Playing Civilization III*, doctoral thesis, Instructional Systems Technology Department, School of Education, Indiana University, Bloomington, IN, USA. Referenced from the on-line version, retrieved on October 25th, 2004, from http://website.education.wisc.edu/kdsquire/dissertation.html.

Stager, Gary S. (2005). *Constructive Technology as the Key to Entering the Community of Learners*, paper presented at the National Educational Computing Conference, June 27-30, 2005, Philadelphia, PA, USA, organized by ISTE – International Society for Technology in Education, Washington, D.C., USA. Referenced from the on-line version, retrieved on August 10th, 2005, from http://center.uoregon.edu/ISTE/uploads/NECC2005/KEY_7582571/Stager_StagerNECC2005_RP.pdf.

Stanley Hall, G. (1901). *The Ideal School as Based on Child Study*, in "Journal of Addresses and Proceedings", no. 488, pp. 475-82, National Education Association, Washington. D.C., USA. Referenced from http://www.teacherscollege.edu/faculty/waite/teach/texts/txt14.htm, retrieved on July 7th, 2005.

Steinmetz, John (2001). *Computers and Squeak as Environments for Learning*, in Mark J. Guzdial & Kimberly M. Rose (eds.), "Squeak: Open Personal Computing and Multimedia", ISBN 0130280917, Prentice-Hall, Inc., Pearson Education, New Jersey, NJ, USA. Referenced from the on-line version at http://coweb.cc.gatech.edu:8888/squeakbook/uploads/steinmetz.pdf, retrieved on September 14th, 2004.

Stowe, Carol Ann (2005). *RE: [Squeakland] Squeak Etoys and preschoolers*, personal e-mail message sent to Leonel Morgado on August 5th, 2005, at 5:38 PM (full text available on Annex II).

Strand, Elizabeth (1986). *A Descriptive Study Comparing Preschool and Kindergarten LOGO Interaction*, paper presented at the Annual Meeting of the American Educational Research Association (70th, San Francisco, CA, April 16-20, 1986), Texas, USA, ED270213, ERIC – Education Resources Information Center, Computer Sciences Corporation, Lanham, MD, USA.

Strommen, Erik (1994). *Children's use of mouse-based interfaces to control virtual travel*, in "Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence", ISBN 0-89791-650-6, pp. 405-410, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/191666.191803 on June 27th, 2005.

Stroustrup, Bjarne (1993). *A History of C++: 1979-1991*, in "The second ACM SIGPLAN conference on History of programming languages", ISBN 0-89791-570-4, pp. 271-297, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/154766.155375.

Studer, Scott D.; Taylor, James; Macie, Ken (1995). *Youngster: a simplified introduction to computing: removing the details so that a child may program*, in "Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education", ISBN 0-89791-693-X, pp. 102-105, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on June 24th, 2005, from http://doi.acm.org/10.1145/199688.199742.

Suppes, Patrick (1995). *The Aims of Education*, in Neiman, Alven; Curren, Randall R., Paul; McCarthy, Christine; Luise Prior, McCarty; Rice, Suzanne; Dummit, Diana; Duncan, Barbara (eds.), "Philosophy of Education Yearbook", ISBN 9995796589, Philosophy of Education Society, University of Illinois, Champaign, Illinois, USA. Referenced from the on-line version, retrieved on July 27th, 2004, from http://www.ed.uiuc.edu/EPS/PES-Yearbook/95_docs/suppes.html.

Sutherland, William Robert (1966). *The On-line Graphical Specification of Computer Procedures*, doctoral thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA.

Suzuki Hideyuki; Kato Hiroshi (1993). *AlgoBlock: a tangible programming language, a tool for collaborative learning*, in P. Georgiadis, G. Gyftodimos, Y. Kotsanits & C. Kynigos (eds.), "Proceedings of the Fourth European Logo Conference, EuroLogo'93, University of Athens, 28-31 August 1993", pp. 297-303, Doukas School S.A., Athens, Greece.

Suzuki Hideyuki; Kato Hiroshi (1995). *Interaction-level support for collaborative learning: AlgoBlock—an open programming language*, in "CSCL '95 - The First International Conference on Computer Support for Collaborative Learning", ISBN 0-8058-2243-7, pp. 349-355, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Suzuki Hideyuki; Kato Hiroshi (2002). *Identity Formation/Transformation as a Process of Collaborative Learning of Programming Using AlgoArena*, in "CSCL 2 – Carrying Forward the Conversation", ISBN 0-8058-3501-6, pp. 275-296, Lawrence Erlbaum Associates, Inc., Mahwah,

NJ, USA. Referenced from the on-line version, retrieved on November 15th, 2004, from http://www.oise.utoronto.ca/cscl/papers/suzuki.pdf.

Swaminathan, Sudha; Trawick-Smith, Jeffrey; Barbuto, Leah M. (2005). *Technology Training at the Preschools: Impact on Pedagogy and Children*, paper presented at the National Educational Computing Conference, June 27-30, 2005, Philadelphia, PA, USA, organized by ISTE – International Society for Technology in Education, Washington, D.C., USA. Referenced from the on-line version, retrieved on August 10th, 2005, from http://center.uoregon.edu/ISTE/uploads/NECC2005/KEY_7469099/Swaminathan_TechnologytrainingSwaminathanTrawickSmithBarbuto_RP.pdf.

Tan, Lesley E. (1985). *Computers in Preschool Education*, in "Early Child Development and Care", ISSN 0300-4430, vol. 19, pp. 319-336, Taylor & Francis Group Ltd., Abingdon, Oxfordshire, UK.

Tanenbaum, Andrew S. (2001). *Modern Operating Systems, Second Edition*, ISBN 0-13-092641-8, Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA.

Tanimoto, Steven L.; Runyan, Marcia S. (1986). *PLAY – An Iconic Programming System for Children*, in Chang S., Ichikawa T., and Ligomenides P. A. (eds.), "Visual Languages", ISBN 0306423502, pp. 191-205, Plenum Publishing Corporation, New York, NY, USA.

Teixeira, Margarida; Sousa, Lúcia; Miguéis, Liliana; Brandão, Irina (in the press). *"Vamos descobrir Portugal" – Como trabalhar com o currículo emergente*, paper presented at the 3rd International Education Congress, "O Mundo da Criança", at UTAD, Vila Real, Portugal, 2004.

TERC (1999). *Screen Shots from My Make Believe Castle*, in "Throught the Glass Wall: Computer Games for Mathematical Empowerment", on-line article retrieved on October 3rd, 2004, from http://www.terc.edu/mathequity/gw/html/MycastlePics.html, TERC Inc., Cambridge, MA, USA.

The MathWorks (no date). *Product List*, Web page, The MathWorks, Inc., Natick, MA, USA. Retrieved from http://www.mathworks.com/products/product_listing/index.html on August 18th, 2005.

Tholander, Jakob (2002). *Children's understanding and explanations of ToonTalk programs. Elaborations on views on objects and actions*, in "Keeping Learning Complex: The Proceedings of the Fifth International Conference of the Learning Sciences (ICLS)", Lawrence Erlbaum Associates, Mahway, NJ, USA. Referenced from the on-line version, retrieved on January 28th, 2004, from http://www.dsv.su.se/research/kids/pdf/ICLS2002.pdf.

Tholander, Jakob (draft, 2002). *Collaborations in children's game programming*. Article available on-line, linking from the site "The Kids Group", Dept. of Computer and Systems Sciences, University of Stockholm, Stockholm, Sweden. Retrieved on January 28th, 2004, from the URL http://www.dsv.su.se/research/kids/pdf/CollaborationsChildrensGameProgramming.pdf. The final version, which does not contain some of the material cited in this thesis, was presented at the workshop "Theoretical perspectives in Human-Computer interaction", 2002, and is available at the URL http://www.nada.kth.se/~tessy/Tholander.pdf, Numerisk Analys Och Datalogi, Kungliga Tekniska högskolan [Royal Institute of Technology], Stockholm, Sweden.

Tholander, Jakob; Fernaeus, Ylva (2003). *Collaborative Computation on the Floor*, in "Proceedings of the International Conference on Computer Support for Collaborative Learning 2003 (CSCL 2003)", ISBN 1-4020-1383-3, Kluwer Academic Publishers, Dordrecht, The Netherlands. Referenced from the on-line version, retrieved on October 7th, 2003, from http://dsv.su.se/research/kids/pdf/FloorProgPoster.pdf.

Tholander, Jakob; Fernaeus, Ylva (2004). *Embodied programming with visual and tangible representations*, to appear in the proceedings of the conference organized by the CSCL group of the Kaleidoscope Network of Excellence, held in Lausanne, Switzerland. Referenced from the on-line

version at http://dsv.su.se/research/kids/pdf/EmbodiedProgramming.pdf, retrieved on July 12th, 2005.

Tholander, Jakob; Kahn, Ken; Jansson, Carl Gustaf (2002). *Real Programming of an Adventure Game by an 8 year old*, in "Keeping Learning Complex: The Proceedings of the Fifth International Conference of the Learning Sciences (ICLS)", Lawrence Erlbaum Associates, Mahway, NJ, USA. Referenced from the on-line version, retrieved on January 28th, 2004, from http://www.dsv.su.se/research/kids/pdf/RealProgInICLSTemplate.pdf.

Thorndike, Edward L. (1912, published 1923). *Education: A First Book*. Macmillan Co., New York, NY, USA. (Reprint edition: ISBN 0405051646, Ayer Co. Pub., Manchester, NH, USA.)

Thrall, Tony; Tingey, Barbara (2003). *SuccessMaker® Motion: A Research Summary*, Technical Note TN030409, Pearson Digital Learning, Pearson Education, Upper Saddle River, NJ, USA. Referenced from the on-line version, retrieved on June 30th, 2004, from http://www.pearsondigital.com/pdfs/whitepapers/SuccessMaker_Motion-A_Research_Summary.pdf

Tomcsányiová, Monika (1999). *Logo Programs for Our 3-year Old Daughter Janka*, in "Proceedings of the Seventh European Logo Conference – Eurologo '99", ISBN 954-9582-03-5, pp. 350-355, Virtech, Sofia, Bulgaria.

Tosic, Predrag T. (2004). *A perspective on the future of massively parallel computing: fine-grain vs. coarse-grain parallel models comparison & contrast*, in "Proceedings of the 1st conference on Computing frontiers", ISBN 1-58113-741-9, pp. 488-502, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/977091.977160, retrieved on August 17th, 2005.

Travers, Michael David (1994a). *LiveWorld: a construction kit for animate systems*, in "Conference companion on Human factors in computing systems", ISBN 0-89791-651-4, pp. 37-38, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/259963.260008 on October 19th, 2004.

Travers, Michael David (1994b). *Recursive interfaces for reactive objects*, in "Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence", ISBN 0-89791-650-6, pp. 379-385, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://doi.acm.org/10.1145/191666.191794 on October 19th, 2004.

Travers, Michael David (1996). *Programming with Agents: New metaphors for thinking about computation*, PhD Thesis, Program of Media Arts and Sciences, School of Architecture and Planning, Massachussets Institute of Technology, Cambridge, Massachussets, USA. Referenced from the on-line version, retrieved from http://xenia.media.mit.edu/~mt/diss/prog-w-agents.pdf on August 20th, 2004.

Tsantis, Linda A. & Bewick, Cynthia J. & Thouvenelle, Suzanne (2003). *Examining Some Common Myths About Computer Use In the Early Years*, in "Beyond the Journal Young Children", November 2003, National Association for the Education of Young Children, Washington, D.C., USA. Web document at http://www.journal.naeyc.org/btj/200311/CommonTechnoMyths.pdf, last retrieved on February 27th, 2005.

Turbak, Franklyn (1996). *First-class synchronization barriers*, in "Proceedings of the first ACM SIGPLAN international conference on Functional programming", ISSN 0362-1340, pp. 157-168, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://cs.wellesley.edu/~fturbak/pubs/icfp96.pdf on August 17th, 2004.

Turcsányi-Szabó, Márta (1997). *Designing Logo Pedagogy for Elementary Education*, in "Eurologo'97 Proceedings", on-line proceedings at http://eurologo.web.elte.hu/prog.htm, part of the Web site of the Sixth European Logo Conference, held in Budapest, Hungary, 20-23 August, 1997. Retrieved from http://eurologo.web.elte.hu/lectures/papthij.htm, last accessed on July 24th, 2005.

Turing, Alan (1936). *On computable numbers, with an application to the Entscheidungsproblem*, in "Proceedings of the London Mathematical Society", ISSN 0024-6115, ser. 2, vol. 42 (1937), pp. 230-265, University of Cambridge, UK. Cited from the online version in facsimile format, retrieved on May 21st, 2004, from http://www.turingarchive.org/browse.php/B/12.

Turkle, Sherry and Papert, Seymour (1990). *Epistemological Pluralism: Styles and Voices within the Computer Culture*, Epistemology and Learning Group Memo no. 3, September 1990, Epistemology and Learning Group, Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. Also published in "Signs", ISSN 0097-9740, vol. 16, no. 1, 1990, pp. 128-155, The University of Chicago Press, Chicago, Illinois, USA. Referenced from the on-line version, retrieved on July 2nd, 2004, from www-personal.si.umich.edu/~rfrost/courses/Women+Tech/readings/Turkle%20(Signs).pdf.

U.S. Bureau of the Census (1988). *Computer Use in the United States: 1984*, U.S. Department of Commerce, Washington, D.C., USA. Referenced from the online version at http://www.census.gov/population/socdemo/computer/p23-155/p23-155.pdf, retrieved on August 4th, 2004.

U.S. Bureau of the Census (1993). *Statistical Abstract of the United States: 1993*, 113th edition, U.S. Department of Commerce, Washington, D.C., USA. Referenced from the online version, retrieved from http://www2.census.gov/prod2/statcomp/documents/1993-01.pdf and http://www2.census.gov/prod2/statcomp/documents/1993-05.pdf, on August 4th, 2004.

U.S. Census Bureau (2001). *Home Computers and Internet Use in the United States: August 2000*, U.S. Department of Commerce, Washington, D.C., USA. Referenced from the online version, retrieved on August 4th, 2004, from http://www.census.gov/prod/2001pubs/p23-207.pdf.

U.S. Census Bureau (2003). *No. HS-42. Selected Communications Media: 1920 to 2001*, Mini-Historical Statistics, in "Statistical Abstract of the United States: 2003", U.S. Department of Commerce, Washington, D.C., USA. Referenced from the online version, retrieved from http://www.census.gov/statab/hist/HS-42.pdf on August 4th, 2004.

U.S. Centennial of Flight Commision (no date). *Commemorating a Century of Wings - An Overview*, Web page at http://www.centennialofflight.gov/essay/Wright_Bros/WR_OV.htm, last accessed on May 18th, 2005.

UNESCO (2003). *Using indicators to assess impact of ICT in education*, Web page at http://www2.unescobkk.org/education/ict/v2_2/info.asp?id=13253, Asia and Pacific Regional Bureau for Education, UNESCO Bangkok, Bangkok, Thailand. Last accessed on July 27th, 2005.

USDE, U.S. Department of Education (2004). *Toward A New Golden Age In American Education – How the Internet, the Law and Today's Students are Revolutionizing Expectations*, ED Pubs, Editorial Publications Center, U.S. Department of Education, Jessup, MD, USA. On-line at http://www.nationaledtechplan.org/docs_and_pdf/National_Education_Technology_Plan_2004.pdf, retrieved on June 24th, 2005.

Vaidya, Sheila; McKeeby, John. (1984). *Computer turtle graphics: Do they affect children's thought processes?*, in "Educational Technology", ISSN 0070-9352, no. 24, pp. 46-47, Educational Technology Publications, Englewood Cliffs, NJ, USA.

Valiant Technology Ltd. (2004). *Valiant Roamer Activity Book*, on-line Web book, at http://www.valiant-technology.com/freebies/book1/book1.htm, Valiant Technology Ltd., London, UK. Last accessed on July 5th, 2004.

Valiant Technology Ltd. (no date-1). *Valiant Roamer User Guide*, Web site at http://www.valiant-technology.com/freebies/userguide/user1.htm, Valiant Technology Ltd., London, UK. Last retrieved on October 26th, 2004.

Valiant Technology Ltd. (no date-2). *Valiant Roamer Design Handbook*, Web site at http://www.valiant-technology.com/freebies/designhandbook/intro.htm, Valiant Technology Ltd., London, UK. Last retrieved on October 26th, 2004.

Vargas, Julie Skinner (no date). *Brief biography of B.F. Skinner*, retrieved on June 30th, 2004, from the Web site of the B. F. Skinner Foundation, at http://www.bfskinner.org/bio.asp.

Vassallo, Charles (no date). *Algorithmic Art*, Web article, retrieved on August 18th, 2005, from http://perso.wanadoo.fr/charles.vassallo/en/art/algorithmic.html.

Vasconcelos, Teresa (1990). *Imaginar o currículo*, in "Cadernos de Educação de Infância", no. 13, APEI – Associação dos Profissionais de Educação de Infância, Lisbon, Portugal.

Vasconcelos, Teresa (1993). *Em busca dos nossos jardins: Margaret McMillan, fundadora das "nursery schools"*, in "Cadernos de Educação de Infância", no. 29, APEI – Associação dos Profissionais de Educação de Infância, Lisbon, Portugal.

Vasconcelos, Teresa (1997). *Ao Redor da Mesa Grande – A prática educativa de Ana*, ISBN 972-0-34454-7, Porto Editora, Oporto, Portugal. This publication is the Portuguese translation of Vasconcelos, Teresa (1995), "Houses and fields and vineyards shall yet again be bought in this land: The story of Ana, a public kindergarten teacher in Portugal", doctoral thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA.

Vasconcelos, Teresa; Silva, Eugénia; Guerra, Ilda; Álvaro, Marieta; Dias, Andreia; Pereira, Helena; Pereira, Magda (1995). *Maria Montessori*, in "Cadernos de Educação de Infância", no. 35, APEI – Associação dos Profissionais de Educação de Infância, Lisbon, Portugal.

Veale, Tony (1995). *Metaphor, Memory and Meaning: Symbolic and Connectionist Issues in Metaphor Interpretation*, doctoral dissertation, School of Computer Applications, Dublin City University, Glasnevin, Dublin, Ireland. Referenced from the on-line version, retrieved on September 3rd, 2004 from http://www.compapp.dcu.ie/~tonyv/thesis.html.

Velgos, Tina (1996). *Imagination rules the kingdom in this interactive gem! My Make Believe Castle*, in "The Review Zone", on-line reviews Web site. Retrieved on August 2nd, 2005, from http://www.thereviewzone.com/mbcastle.html.

Verostko, Roman (2004). *Algorithmic Art – Composing the Score for Visual Art*, in "Intelligent Agent", vol. 4, no. 1, Intelligent Agent, Inc., New York, NY, USA. Referenced from the on-line version at http://www.intelligentagent.com/archive/IA4_1generativityverostko.pdf, retrieved on August 18th, 2005.

Viacom, Inc. (1998). *Viacom To Complete Sale Of Non-Consumer Publishing Operations To Pearson On November 27*, on-line press release, retrieved on July 23rd, 2004, from http://www.viacom.com/press.tin?ixPressRelease=40000904.

von Neumann, John (1945). *First Draft of a Report on the EDVAC*, Contract No. W-670-ORD-4926. Between the United States Army Ordnance Department and the University of Pennsylvania. Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945, Philadelphia, Pennsylvania, USA. Referenced from the electronic version retrieved on May 14th, 2004, from http://www.virtualtravelog.net/entries/2003-08-TheFirstDraft.pdf. The document at this address reports that the document was reproduced «from the companion CD-ROM to the IEEE CS Press book, "The Anatomy of a Microprocessor: A Systems Perspective," by Shriver & Smith"» (*sic*).

VV.AA. (1998). *User Notes on Fortran Programming (UNFP)*, an open cooperative practical guide, available at the Web site http://shum.cc.huji.ac.il/~agay/fortran/unfp/unfp.html, last accessed on August 16th, 2005.

VV.AA. (2004), *comp.sys.sinclair FAQ*. Electronic resource, retrieved from http://www.srcf.ucam.org/~pak21/cssfaq/ on March 5th, 2004.

VV.AA. (2005). *How old are you lemmings fan?*, Web forum thread, Lemmings Forums, YaBB (Yet another Bulletin Board), http://www.yabbforum.com/. Retrieved on August 2nd, 2005, from http://eng-forum.lemmingswelt.de/cgi-bin/yabb/YaBB.cgi?board=offtopic_id;action=display;num=1106062504.

Vygotsky, Lev Semyonovich (1930). *Socialisticheskaja peredelka cheloveka*, in "Varnitso", journal of the All-Union Association of Workers in Science and Technics for the Furthering of the Socialist Edification in the USSR, transcribed by Andy Blunden, in (1994) "Vygotsky Reader", René van der Veer and Jaan Valsiner (eds.), Blackwell's, Oxford, UK. Referenced from the on-line version at http://www.marxists.org/archive/vygotsky/works/1930/socialism.htm, retrieved on July 16th, 2005.

Vygotsky, Lev Semyonovich (1934). *Myshlenie i rech'*. Referenced from the English translation "*Thought and language*", edited and translated by Eugenia Hanfmann and Gertrude Vakar, MIT Press, Cambridge, MA, USA.

Vygotsky, Lev Semyonovich (1978). *Mind in society: The development of higher psychological processes*, ISBN 0-674-57628-4, Harvard University Press, Cambridge, MA, USA.

Walters, Eric; Beaver, Natalie (2004). *Developing and Implementing a Robotics Curriculum in the Lower School Science Classroom*, online paper retrieved on October 4th, 2004, from http://www.marymount.k12.ny.us/marynet/TeacherResources/ScienceDept/other/waltersbeaver.pdf, Marymount School of New York, New York, NY, USA.

Warash, Barbara Gibson (1984). *Computer language experience approach*, paper presented at the Annual Meeting of the National Council of Teachers of English Spring Conference (Columbus, OH, USA. April 1984). Referenced from Clements & Sarama, 2003, and Coniam, 1992.

Warner, Silas (1981). *RobotWar*, game manual, MUSE Software, Baltimore, MD, USA. Referenced from the on-line version, retrieved on October 6th, 2004, from http://files.the-underdogs.org//games/r/robotwar/files/robotwar.pdf.

Weaver, Constance L. (1991). *Young Children Learn Geometric Concepts Using Logo with a Screen Turtle and a Floor Turtle*, research report, ED329430, ERIC – Education Resources Information Center, Computer Sciences Corporation, Lanham, MD, USA.

Wegerif, Rupert (2002). *Literature Review in Thinking Skills, Technology and Learning*, report produced for NESTA Futurelab, Bristol, UK. Referenced from the on-line version, retrieved from http://www.nestafuturelab.org/images/downloads/Thinking_Skills_Review.pdf on June 24th, 2005.

Wegner, Peter (1990). *Concepts and Paradigms of Object-Oriented Programming: Expansion of Oct 400PSLA-89 Keynote Talk*, in "ACM SIGPLAN OOPS Messenger", ISSN 1055-6400, vol. 1, no. 1, pp. 7-87, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/382192.383004.

Weikart, David P. (2000). *Early childhood education: need and opportunity*, ISBN 92-803-1197-2, International Institute for Educational Planning, UNESCO, Paris, France. Referenced from the on-line version, retrieved from http://unesdoc.unesco.org/images/0012/001223/122380e.pdf on July 23rd, 2005.

Wells, Herbert George (1937). *World Brain: The Idea of a Permanent World Encyclopaedia*, in "Encyclopédie Française", Société de gestion de l'Encyclopédie française, Paris, France. Referenced from the electronic version, retrieved from http://art-bin.com/art/obrain.html on July 1st, 2004.

Wenger, Etienne C. (1998). *Communities of Practice – Learning, Meaning, and Identity*, ISBN 0521430178, Cambridge University Press, Cambridge, UK. Referenced from the Web page

http://www.co-i-l.com/coil/knowledge-garden/cop/lmi.shtml, Community Intelligence Labs, CA, USA. Retrieved on July 9th, 2005.

Westbrook, Robert B. (1993). *John Dewey (1859-1952)*, in "PROSPECTS: quarterly review of comparative education", ISSN 0033-1538, vol. 23, no. 1/2, pp. 277-291, UNESCO International Bureau of Education, Geneva, Switzerland. Referenced from the on-line version, retrieved from http://www.ibe.unesco.org/International/Publications/Thinkers/ThinkersPdf/deweye.PDF on July 20th, 2005.

Whitley, Kirsten N. & Blackwell, Alan F. (1997). *Visual programming: The Outlook from Academia and Industry*, in S. Wiedenbeck & J. Scholtz (eds.), "Papers presented at the seventh workshop on Empirical studies of programmers", ISBN 0-89791-992-0, pp. 180-208, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 26th, 2004, from http://doi.acm.org/10.1145/266399.266415.

Whitley, Kirsten N. & Blackwell, Alan F. (2001). *Visual Programming in the Wild: A Survey of LabVIEW Programmers*, in "Journal of Visual Languages and Computing", ISSN 1045-926X, vol. 12, no. 4, pp. 435-472, Academic Press, Cambridge, UK. Referenced from the on-line version, retrieved from http://dx.doi.org/10.1006/jvlc.2000.0198 on August 26th, 2004.

Whitley, Kirsten N. (1996). *Visual Programming Languages and the Empirical Evidence For and Against*, in "Journal of Visual Languages and Computing", ISSN 1045-926X, vol. 8, no. 1, pp. 109-142, Academic Press, Cambridge, UK. Referenced from the on-line version, at http://citeseer.ist.psu.edu/cache/papers/cs/17698/http:zSzzSzcuisung.unige.chzSzVisualzSzlocalzSzWhitley96.pdf/whitley96visual.pdf (dated October 1996), retrieved on August 26th, 2004.

Wilcox, Douglas M. (1999). *What Is Lego Mindstorms?*, Web page retrieved on August 1st, 2005, from http://www.wilcoxusa.net/mindstorms/what_is_lego_mindstorms.htm.

Wilensky, Uriel Joseph (1991). *Abstract Meditations on the Concrete and Concrete Implications for Mathematics Education*, in "Constructionism", ISBN 0893917869, Ablex Publishing Corporation, Norwood, NJ, USA. Referenced from the on-line version at http://www.ccl.sesp.northwestern.edu/papers/concrete/, retrieved on September 1st, 2004.

Wilensky, Uriel Joseph (1993). *Connected Mathematics – Building Concrete Relationships with Mathematical Knowledge*, doctoral dissertation, Program in Media Arts and Sciences, School of Architecture and Planning, Massachussets Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved from ccl.northwestern.edu/papers/download/Wilensky-thesis.pdf on June 29th, 2004.

Williams, Laurie A.; Kessler, Robert R. (2000). *All I Really Need to Know about Pair Programming I Learned in Kindergarten*, in "Communications of the ACM", ISSN 0001-0782, vol. 43, no. 5, pp. 109-114, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on January 21st, 2001, from http://www.acm.org/pubs/articles/journals/cacm/2000-43-5/p108-williams/p108-williams.pdf.

Williams, Michael R. (1990). *Early Calculation*, in "Computing Before Computers", ch. 1, pp. 3-58, ISBN 0-8138-0047-1, Iowa State University Press, Ames, Iowa, USA. Referenced from the electronic version at http://ed-thelen.org/comp-hist/CBC.html, retrieved on April 29th, 2004.

Windelband, Wilhelm (1919). *A history of philosophy : with special reference to the formation and development of its problems and conceptions*, MacMillan Company, New York, NY, USA.

Winter, David [1] (2004). *PONG-Story*. Electronic resource, retrieved on March 5th, 2004, from http://www.pong-story.com.

Winter, David [2] (2004). *Videomaster*, Electronic resource, retrieved on March 5th, 2004, from http://www.pong-story.com/vm.htm.

Wirth, Niklaus (1993). *Recollections about the Development of Pascal*, in "The second ACM SIGPLAN conference on History of programming languages", ISBN 0-89791-570-4, pp. 333-342, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 16th, 2005, from http://doi.acm.org/10.1145/154766.155378.

Witherspoon, Tonya (2003). *Classroom Photos - Marcia Helten's Preschool Classroom*, Web page at http://education.wichita.edu/mindstorms/roamer/grant2003/Marcia/roamer_pics.html, Web site "Robotics in the Classroom", College of Education, Wichita State University, Wichita, Kansas, USA. Retrieved on August 2nd, 2005.

Wolf, Mark J. P.; Perron, Bernard (2003). *Introduction*, in "The Video Game Theory Reader", ISBN 0-415-96579-9, p. 14, Taylor & Francis Books, Inc., New York, USA.

Wolfram Research (no date). *Mathematica – The Way the World Calculates*, Web page at the URL http://www.wolfram.com/products/mathematica/index.html, Wolfram Research, Inc., Long Hanborough, Oxfordshire, UK. Retrieved on August 18th, 2005.

Wright, June L. (1998). *A New Look at Integrating Technology into the Curriculum*, in "Early Childhood Education Journal", ISSN 1082-3301, vol. 26, no. 2, pp. 107-109, Springer Science+Business Media B.V., Berlin, Germany. Referenced from the on-line version, retrieved from http://www.springerlink.com/media/6CC6L524GG5TWN992X2M/Contributions/Q/T/2/2/QT228252724U9101.pdf on August 2nd, 2005.

Wright, Timothy Nicol (2004). *Collaborative and Multiple-Notation Programming Environments for Children*, doctoral dissertation, University of Canterbury, Christchurch, New Zealand. Referenced from the on-line version, retrieved on September 5th, 2004 from http://www.mcs.vuw.ac.nz/~tim/thesis.pdf.

Wyeth, Peta; Purchase, Helen C. (2000). *Programming without a Computer: A New Interface for Children under Eight*, in "Proceedings of the First Australasian User Interface Conference", ISBN 0-7695-0515-5, IEEE Computer Society, Washington, DC, USA. Referenced from the on-line version, at http://csdl.computer.org/dl/proceedings/auic/2000/0515/00/05150141.pdf, retrieved on October 4th, 2004.

Wyeth, Peta; Purchase, Helen C. (2002). *Tangible Programming Elements for Young Children*, in "CHI '02 extended abstracts on Human factors in computing systems", ISBN 1-58113-454-1, pp. 774-775, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://www.itee.uq.edu.au/~peta/wyeth2002.pdf on June 11th, 2005.

Wyeth, Peta; Purchase, Helen C. (2003). *Using Developmental Theories to Inform the Design of Technology for Children*, in "Proceedings of the 2003 conference on Interaction design and children", ISBN 1-58113-732-X, pp. 93-100, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved from http://www.itee.uq.edu.au/~peta/idc2003Wyeth.pdf on August 2nd, 2005.

Wyeth, Peta; Wyeth, Gordon (2001). *Electronic Blocks: Tangible Programming Elements for Preschoolers*, in M. Hirose (ed.), "Human-Computer Interaction - INTERACT'01", ISBN 1-58603-188-0, IOS Press, Amsterdam, The Netherlands. Referenced from the on-line version, retrieved on October 4th, 2004 from http://www.dstc.edu.au/Research/Projects/Ambience/WyethInteract.pdf.

Yamashita, Jun (no date). *Development of a CSCL Environment using AlgoCard*, personal Web page at the Hirose & Hirota Laboratory, RCAST, Department of Information Systems & Intelligent Cooperative Systems, University of Tokyo, Tokyo, Japan. Retrieved on October 6th, 2004, from http://jun.cyber.rcast.u-tokyo.ac.jp/RESEARCH/GRAD/THESIS.HTM.

Yamashita, Jun; Kuzuoka, Hideaki; Suzuki, Hideyuki; Kato, Hiroshi (1997). *Development of AlgoCard for CSCL Environment*, in "IPSJ SIGNotes GroupWare No. 23", Information Processing Society of Japan, Chiyoda-ku, Tokyo, Japan. Abstract available on-line, retrieved on October 6th, 2004, from http://www.ipsj.or.jp/members/SIGNotes/Eng/20/1997/023/article007.html.

Zemelman, Steven; Daniels, Harvey; Hyde, Arthur (1995). *Teachers as Learners: What Helps Teachers Grow?*, in "Best Practice", no. 9, Fall 1995, IFS International, Bedford, UK. Referenced from the on-line version, retrieved from http://www.ncrel.org/mands/docs/9-12.htm on January 27th, 2004. This article is an adaptation from Zemelman, Steven; Daniels, Harvey; Hyde, Arthur (1993), "Best Practice: New Standards for Teaching and Learning in America's Schools", ISBN 0435087886, Heinemann, Portsmouth, NH, USA.

Zuckerman, Oren (2004). *System Blocks: Learning about Systems Concepts through Hands-on Modeling and Simulation*, Master's thesis, School of Architecture & Planning, Massachusetts Institute of Technology, Cambridge, MA, USA. Referenced from the on-line version, retrieved from http://web.media.mit.edu/~orenz/papers/mit/Oren_Zuckerman_MS_Thesis.pdf on August 2nd, 2005.

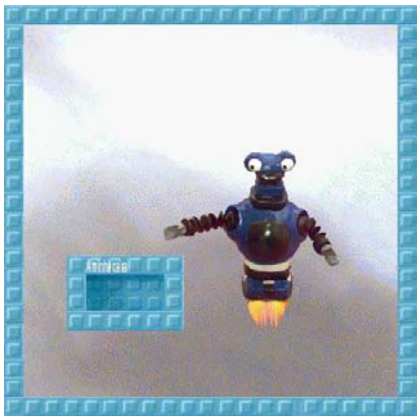
Zuckerman, Oren; Arida, Saeed; Resnick, Mitchel (2005). *Extending Tangible Interfaces for Education: Digital Montessori-inspired Manipulatives*, in "Proceedings of the SIGCHI conference on Human factors in computing systems", ISBN 1-58113-998-5, pp. 859-868, ACM Press, New York, NY, USA. Referenced from the on-line version, retrieved on August 2nd, 2005, from http://web.media.mit.edu/~orenz/papers/mit/zuckerman_CHI05_Digital_MiMs.pdf.

Zuckerman, Oren; Resnick, Mitchel (2003). *A physical interface for system dynamics simulation*, in "CHI '03 extended abstracts on Human factors in computing systems", ISBN 1-58113-637-4, pp. 810-811, ACM Press, New York, NY, USA. Referenced from the on-line version at http://doi.acm.org/10.1145/765891.766005, retrieved on October 4th, 2004.

Zuse, Horst (v. 1999). *The Life and Work of Konrad Zuse*, Everyday Practical Electronics Online, electronic article, retrieved from http://www.epemag.com/zuse/default.htm on May 20th, 2004.

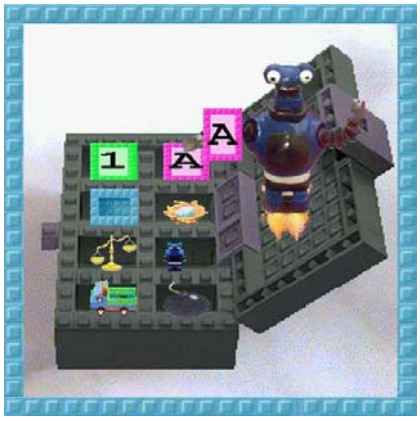
Annex I
—
A robot that writes “MARIA”

Maria, March 29th, 2001 – *Robot "Writes Maria"*



The robot starts with an empty box.

Goes to the letters stack.



Picks up a letter.

Erases the letter.

Writes "M".

Writes "A".

Writes "R".

Writes "I".

Writes "A".

Puts the text in its box.

THE END

Robot “Writes Maria” in execution (various iterations)

Since the starting constraint is an empty box, which in ToonTalk means “a box regardless of its contents”, the robot won’t stop after dropping “MARIA” in the box. It's constraint is still valid, so it will run again, and again, indefinitely. The following pictures provide a sample of the result. I enlarged the width of the box, to make it visible here. In ToonTalk, one might simply pick up the contents of the box directly, to inspect them, or resize the box as needed, while the robot runs.

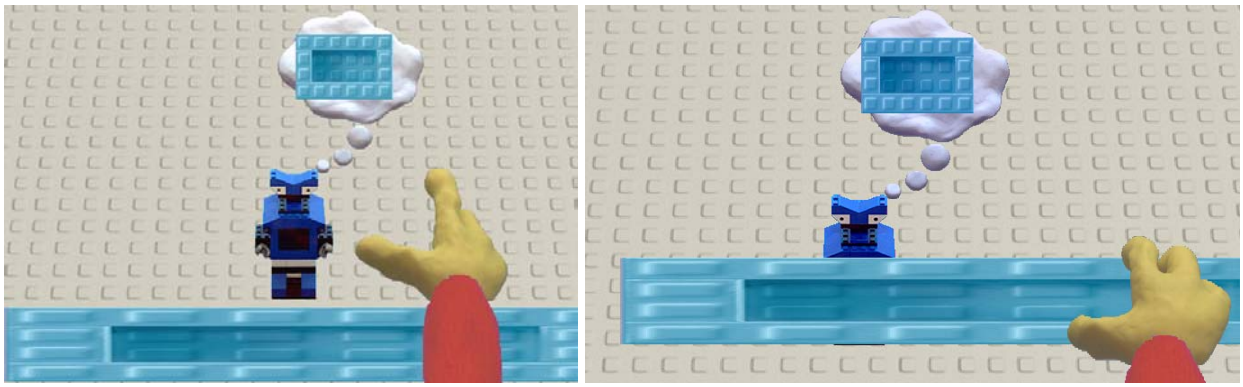


Figure 281 – Original situation (empty box) and starting the process (giving the box to the robot)

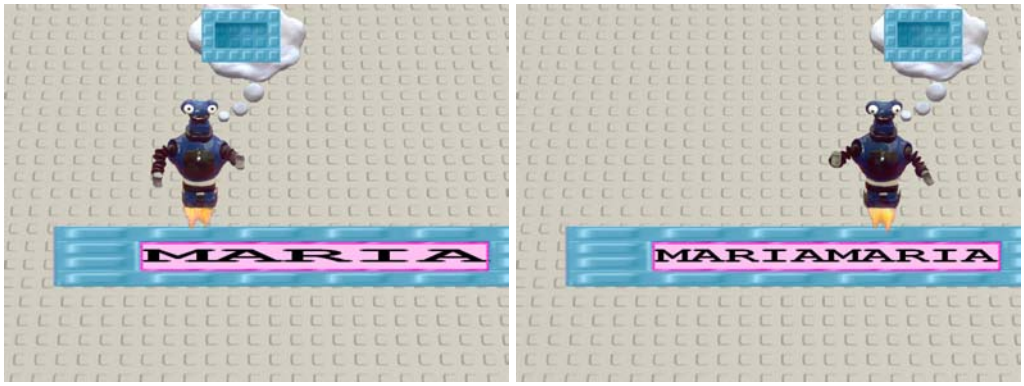


Figure 282 – After the first (left) and second (right) iterations

Figure 283 – During the third iteration, “MARIA” is bammed on the previous contents; result of the third iteration (center); and after 12 iterations (right)

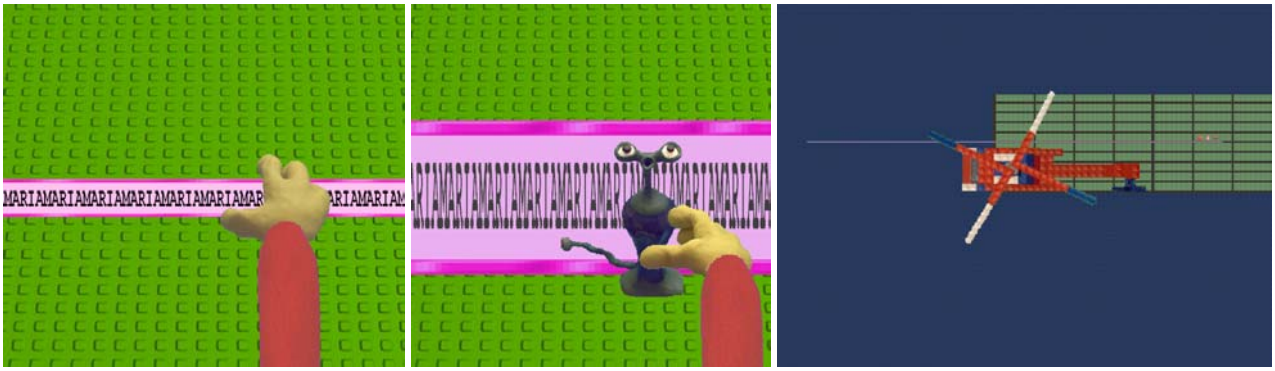


Figure 284 – After many iterations (no need to wait, just stroll outside the house for two seconds while the robot works at full speed), taking the name outside, making it tall, and flying over it: the name extends over the sea!

Annex II
—
E-mail exchanges

E-mail exchange with Radia Perlman

From: Radia Perlman
Sent: quinta-feira, 26 de Maio de 2005 19:14
To: Leonel Morgado
Subject: Re: RE: Computer Programming in Preschool

I don't remember the ages of the children, unless I specified it in the paper that you have (I haven't looked at the paper recently, so I don't remember what's in it).

And yes, you have permission to forward or include the email.

Radia

From: Leonel Morgado
Date: Thursday, May 26, 2005 11:09 am
Subject: RE: Computer Programming in Preschool

Hi,

Thanks for this personal account! These are the kind of remarks and observations that are true gold for what I'm doing, because they provide actual insights on how things were happening at the time, and how it went. I can find several parallels between what happened to you and several of my initial programming sessions in ToonTalk.

My PhD is in computer science, but I am not developing any new software or hardware: plenty of that already, the way I see it, but not enough reflection on how to employ concepts from computer science at the preschool level. The problem isn't just kids: teachers don't realize how computer science can be involved at all, and computer scientists don't often realize how kids so young can get involved. So, I'm trying to bridge both worlds.

You might be interested in a draft paper I wrote, with examples of how to use computer science in preschool activities. It is available here (comments are welcome!), but it's a bit large (6 MB):
http://home.utad.pt/~leonelm/cookbook.pdf

I just had a few more questions:

1. Did you also worked with children aged 3, or only with 4- and 5-year olds?
2. Can I include your mail(s) in my thesis, as an Appendix?
3. If not, can I forward them to my advisor, Ken Kahn?

Inté,

Leonel

From: Radia Perlman
Sent: quinta-feira, 26 de Maio de 2005 18:22
To: Leonel Morgado
Subject: Re: Computer Programming in Preschool

I'm so glad you're interested! Unfortunately, that work was a long time ago, and I was more into building the system rather than rigorously studying how well students learned with it. And although the design was nice, I also built the hardware, and I am not good at that, so it was very expensive and fragile. Today I'm sure someone could rebuild the whole thing a lot better, and then have more studies with students.

Is your PhD in psychology or computer science?

My recollection, from the relatively small trials we did on the equipment, was that 3-5 year olds understood the basic button box (the direct commands) with no problem. They were relatively uninterested in the numbers (it wasn't that exciting to be able to say "5 forward" instead of hitting "forward" 5 times). And they were pretty baffled by the programming buttons.
They did manage to use them to fill the screen with the icons for the commands, but they were ignoring the turtle when they did that. They'd hit "start remembering", then hit lots of commands, in particular the light and horn because those icons were really pretty, and then hit "forget it" and then start again, all the time ignoring the turtle. And when people tried to get them to understand the concept of a program, the teacher would say "ignore the screen...teach the turtle to draw a square...hit start remembering...now have it draw a square".
They could do this, but when they hit "do it", they were confused if the turtle turned the wrong way (because that's how they'd hit the commands when they taught the turtle to draw a square). THey thought the picture the turtle drew was the program, rather than the sequence of commands.
So that's why I did the slot machine, because physically picking up a "command" and putting it into the "procedure" seemed like they'd have to understand it.

The slot machine was quite successful with older kids if I remember correctly (7 and up), but didn't interest the 3-5 year olds too much. But there was also not very much use of the slot machine, because it was so fragile it was often broken...if it was jiggled the cards wouldn't line up and the machine would read the wrong commands. And shortly after building the slot machine, I wound up leaving grad school, and nobody else really picked up on the research.

So maybe you will start where I left off! Good luck, and feel free to keep in touch.

Radia

Research.Sun.Com Feedback wrote:

>Reply-To: Leonel Morgado
>URL: www.utad.pt/~leonelm
>
>Hi,
>
> I am a PhD student in Portugal, working with a language called ToonTalk
>(www.toontalk.com). The focus of my research is the use of computer programming
>in preschools and kindergartens, under the coordination and advice of Ken Kahn.
> Your work with the TORTIS system was an important mark, and I'd
>like to know more about your work with children aged 3 to 5. I found
>two of your papers on the subject: TORTIS (1974) and Using computer
>technology to provide a creative learning environment for preschool
>children (1976). Any information would be greatly appreciated.
>
>Kind regards,
>Leonel Morgado

E-mail exchange with Tiago Correia

I called one of the developers of the “Magic Forest” product, Tiago Correia, on his cellphone, inquiring on what he knew about trials with children aged 4 or 5. This e-mail is his follow up.

```
Floresta Mágica Reports
Data: Mon, 1 Aug 2005 16:30:39 +0100
De: Tiago Correia[+]
Para: 'Leonel Morgado'[+]
Partes:

Olá,



Estiver a conversar com o Secundino e com a Manuela e os testes documentados
forma feitos apenas com crianças dos 6-8 anos. Com as crianças mais novas foram
feitos testes informais e não ficaram documentados que é uma pena. No entanto na
página do projecto Playground, donde surgiu então a Floresta Mágica (no projecto
chama-se Pathways), estão lá vários relatórios. Seguem em anexo um link para os
relatórios finais. Os que te podem interessar são os WP4 e WP6. Mas talvez os
outros também te interessem.



http://www.ioe.ac.uk/playground/RESEARCH/reports/finalreport/index..htm



Qualquer coisa, manda um e-mail.



--

Tiago Correia

http://www.cnotinfor.pt
```

E-mail exchange with Paul Goldenberg

From: Paul Goldenberg
To: <leonelm@utad.pt>
Subject: Re: FASTR system - 1970s
Date: Sun, 31 Jul 2005 00:00:24 -0400

Hi Leonel,

I am that Paul, and tried to find some old document that described the FASTR system, but it's a long time ago and I could find nothing.

A few of us were looking for ways to make programming accessible for non-readers (or limited typers). FASTR was a single-key, immediate response system. Written in Logo, a program scanned for characters typed to the keyboard. When certain ones were typed -- F, B, R, L, and a small number of others -- the turtle responded immediately, with default distances and angles built in. The program also kept a *record* of these moves, so that the young programmer could undo a move (delete key), replay the set of moves (I've forgotten how), and even name a procedure (after having executed all the moves). The list of captured commands was "simplified" by the Logo program that was recording them all, so that if a student did, for example, FFLRRRRFFRRRFFRRRFF, the captured program would combine all the "like" commands that were adjacent to each other, and be FD 100, RT 90, FD 100, RT 90, FD 100, RT 90, FD 100. I forgot how I had had the child designate a name for the program and assign it some key on the keyboard. We were, at the time, assuming that the virtue of having readable programs in "real" Logo was that kids, after experimenting, *would* look at these programs at some later time. I also built several trap-doors in to the system to allow the teacher to change the default values so that, for example, R could stand for RT 90 instead of RT 30, and so that we could assign different keys (which would then have stickers placed on them to identify them) for kids who were no accurate enough in their coordination to, for example, press F without also pressing R by accident.

Radia Perlman was also inventing systems, one involving rearrangeable cards. These days, we'd probably use cards with bar-codes on them, but she used some other technology. Those didn't try capturing the programs in any child-readable form.

If there's more I can help with, please let me know. Good luck with your research. I'd be very curious to know more about what you are looking at, and what you're finding.

--Paul

On 7/30/05 1:30 PM, "leonelm@utad.pt" <leonelm@utad.pt> wrote:
> Hi,
>
> I am researching the use of computer programming in preschool settings, at the
> UTAD University, in Portugal (www.utad.pt/~leonelm).
> As I was reading a Logo memo from 1975, I saw them mention that a Paul
> Goldenberg had developed a simplified programming system for children, called
> FASTR. I am writing to you hoping that you're the same Paul Goldenberg.
> If so, would you have the opportunity to provide me with some info on that
> FASTR system? For instance, any papers you might have written, or photos... If
> nothing else, perhaps your recollections in an e-mail?
>
> Sincerely,
> Leonel Morgado leonelm@utad.pt

E-mail exchange with John R. Nyquist

From: John R. Nyquist
To: leonel.morgado

Date: Aug 4, 2005 6:37 PM
Subject: Stagecast Creator and preschoolers

Hi Leonel,

Your post was passed along to me by Fiedelia Kuang. I have used Stagecast Creator with my boy when he was 4 and 5 (he's now six). He was able to make characters move around the screen using the keyboard, he was able to work through the tutorials (with occasional help), but his favorite part I think was editing the images. He was a big KidPix user at the same time (and earlier). Most of the time, we'd work/play together on Stagecast.

Who are you doing it for? I'm interested in seeing the results of your research. I used to create educational applications in Macromedia Director (years ago). I have also written on how to program in Lingo and taught how to program in Java. I've had an interest in how people (young and old) learn to program, perhaps because I learned as an adult.

Regards,
John R. Nyquist

E-mail Exchange with Kim Rose

From: Kim Rose
To: Leonel Morgado, squeakland@squeakland.org
Date: Aug 4, 2005 4:44 PM
Subject: Re: [Squeakland] Squeak Etoys and preschoolers

Hello Leonel,
There was work done last year at Columbia College's (Chicago) Early Childhood Education dept. directed by Carol Ann Stowe and Val Scarlatta -- I've copied them here. They designed an Etoy interface appropriate for this age child. They will present this work at SqueakFest at Columbia just next week -- I don't know where you are located, but you should try to come!
Anyways...I hope Carol Ann and Val will respond to you directly.
regards,
Kim

At 5:31 PM +0100 8/3/05, Leonel Morgado wrote:
>Hi,
>
> I'm interested on computer programming experiences with children
>aged 3, 4 and 5.
> I was unable to find documented cases of Etoys use in this age
>group, but expect that there should be at least a few user stories.
>
> If anyone has reports, photos, or even just recollections of
>such cases, I would love to hear about them!
>
>Inté,
>
>Leonel Morgado

E-mail Exchange with Carol Ann Stowe

From: Stowe, Carol Ann
To: Kim Rose, Leonel Morgado
Date: Aug 5, 2005 5:38 PM
Subject: RE: [Squeakland] Squeak Etoys and preschoolers

Leonel

The most exciting work took place at the University of Chicago Laboratory School in my daughter Elspeth Stowe-Grant's classroom. She and Val worked together so that her 4 year old class could use Squeak in their study of Matisse. We have hours of video that I am going to analyze for patterns, but Val made some very interesting modifications of Squeak to make it 4 year old friendly. She isolated and enlarged the tiles needed for the project at hand.

We are working on our presentation for Sqeakfest and a book documenting the process. We have been working in two other classrooms too - a kindergarten and a first grade (the next 2 levels here in the States). We will send things along when they are ready.

Carol Ann

E-mail Exchange with Alan Kay

From: Alan Kay
To: Leonel Morgado
Date: Aug 4, 2005 7:54 PM
Subject: Re: [Squeakland] Squeak Etoys and preschoolers

Hi Leonel --

I'm not sure that the PARC internal reports are easy to find -- that was 33 years ago ...

However, I think the best thing is to press on and do a new set of experiments with fresh thoughts.

Also, check out the kindergartens in Reggio Emilia, Italy. They have done some very neat things with 3,4,5 year olds, and with LOGO, etc. I will be visiting there in Sept to see the latest stuff.

Cheers,

Alan

At 11:36 AM 8/4/2005, Leonel Morgado wrote:
>Hi Alan,
>
>   Thanks for this information! A few minutes before e-mailing the
>Squeak list, I had also sent you a personal e-mail, so don't be amazed
>when you come across it.
>   I am quite aware of Radia's work, but when I exchanged e-mails
>with her she didn't mention any duplication of the Button Box or of
>the Slot Machine. So, any info on those paralell experiments would be
>great.
>   Kim has already e-mailed me about that group that has prepared an
>adequate Etoys interface for this age group, but I would love to get
>in touch with that grad student you mention.
>
>Inté,
>
>Leonel
>
>On 8/4/05, Alan Kay wrote:
> > Hi Leonel --
> >
> > We have done little in this age group with Etoys.
> >
> > The UI should be changed for this age children in a number of important
> > ways (some of this was in the original Etoys design but didn't get
> > implemented).
> >
> > You are probably aware of Radia Perlman's work in the 70s at MIT with a
> > "button box LOGO"? This was pretty interesting, and we duplicated her
> > equipment and did many parallel experiments. I have heard recently about
> > very young children and Etoys (some of them will be at SqueakFest next
> > week) but I don't know the details. Kim Rose might. We also have a grad
> > student working with us who is interested in this age group.
> >
> > Cheers,
> >
> > Alan
> > At 09:31 AM 8/3/2005, Leonel Morgado wrote:
> > >Hi,
> > >
> > >   I'm interested on computer programming experiences with children

> > >aged 3, 4 and 5.
> > > I was unable to find documented cases of Etoys use in this age
> > >group, but expect that there should be at least a few user stories.
> > >
> > > If anyone has reports, photos, or even just recollections of
> > >such cases, I would love to hear about them!
> > >
> > >Inté,
> > >
> > >Leonel Morgado

Annex III
—
Reports detailing the sessions of May-June 2000

São Pedro Parque preschool

Date Monday, May 2nd 2000
Participants Z (boy, 5 years) and O (boy, 4 years).
Starting time 11h42
End time 12h03

While they were sitting in front of the monitor, which was already turned on, they began by mentioning that neither of them had a computer. Z said that a cousin of his had one; O told me he had none, but that he was going to ask his parents to buy one for him.

I told them that we were going to play a game in which we could do lots of things – even make games.

They immediately noticed the ToonTalk icon on the desktop, referring to it with the sentence "Look, a frog!"

I said, "It looks like one – it's green: actually, it's a Martian."

I asked if they knew how to start a program and they said they did; therefore, I asked them to move the mouse over to the icon.

Z at once got the mouse and started moving it, but his movements weren't precise enough (the icon had a normal size, on an 800x600 resolution screen), and so I put my hand over his, helped him to move the mouse and performed a double-click with my finger over his.

Afterwards, we released the mouse, which I held (while saying "allow me"), and then I said "Let's play, then." As I was saying this, I clicked the "Brincar" (play) button.

Upon the login dialog, I said, "I'm going to write your names" and I wrote "Z and O". Z said "but my real name isn't Z." [I was writing a shortened form of his real name, which started with a J, not a Z]. I replied: "yeah, you're right. I'll write «J», then." But he said: "It's P."

I said: "Oh! But I shouldn't write «J P and O» here, it's too long[323] a thing."

(As I was saying "long", I moved my hands apart – Orlando's gaze drifted, staring at the hand farthest away.)

Afterwards, I explained that we were going to control a little boy, which could have long hair, a hat or be bald. I asked them what they proffered. Z wanted one with a hat, and O wanted one with long hair. Then I told them that the solution for this would be to make a draw.

I fetched two pieces of colored chalk: O picked up the blue one, Z got the orange one.

I mixed them behind my back, and then showed a closed fist (with the back up). Then I told them "I scrambled the contents of my hands behind my back – the chalk here will decide who chooses." I then asked them to hit my hand – when I opened it, the orange chalk was within. This meant that Z would have the right to choose, and I confirmed if he wanted a boy with a hat.

When the chopper showed up, I asked them if they knew what it was – they both correctly identified it as a chopper, and also the town houses.

I explained (and showed them) that the left mouse button would make the chopper go up, and the right one would make it go down. And also that by moving the mouse we could move the chopper. While doing this, I took the opportunity to leave the chopper at a comfortably high altitude, to avoid it from landing accidentally.

O was the first to move it, even though that when I told him to move the mouse he didn't understand immediately. But I placed my hand over his, just to clarify.

O's hand is too small for the mouse, making it hard to control. But he managed to make the chopper rise and descend. Then, with my hand over his, we controlled the movement of the chopper.

Z was much more at ease, managing to control the chopper very well. I asked Z to make it land, and he performed this action. He did hesitate a bit when the perspective changed, but I told him to go on, and the chopper landed.

I then made the figurine move, we confirmed that it had a hat, and I explained that we controlled the figure with the mouse. And also that the green thing following it was our toolbox, Tooly.

[323] I wanted that later either they or their teacher could easily enter the saved play situation. I feared that by having a long username this might not work out properly.

I made the figurine go into a house; Marty the Martian showed up, but the text balloon was covering almost the entire screen area. I let Marty speak for a while. When they heard the reference to the F1 key, they asked me which one it was. At this moment, I took advantage of what the Martian was saying (that the F1 key would make him appear and disappear) to demonstrate its operation.

I made the Martian go away, while explaining that it was there to help us, but that since I was with them, I could do that, and we could do without having the Martian with us.

Following my instructions, O clicked the mouse making the figurine kneel. On seeing the open toolbox, Z immediately said it contained a letter, and O mentioned that it had the letter A inside.

I explained that we were seeing the figurine's hand; both of them tried to move it. We checked that when the finger was placed over anything (the letter A, the nest) that object would start moving, and that when we removed the hand, it would stand still again.

I pointed to one of the gaps in Tooly and said that it contained robots with which we were going to work. "Like Rosa's?" they asked[324]. "No," I answered, "these have arms and legs." And I showed them a robot.

I asked if they knew what "this" was, while pointing at the thought bubble. They said they did. "What is it then?" I asked. They told me it was a cloud. I said that it did resemble one; that the little balls that came from the robot's head were there because in the cloud were the robot's thoughts.

"Can you tell me what's he thinking of?" I asked.

"Snow," said Z.

"Yeah," I said, "but there's nothing else there."

I then showed them a box. Then I picked up another, and let them move them. Z immediately tried to place one besides another. I took the opportunity to say that he could place the second one slightly over the edge of the first one, which he did, and the mouse with the hammer came up to join them.

Orlando, from then on, started calling it "the man with the hammer."

Then we all moved the box a bit, so that they could realize that it was now a single box.

I then picked up the images notebook, from which I took out a tree and a flower.

I placed each in one of the box's holes.

I then explained that we wanted to teach the robot how to swap things that were in the box: to move the flower to one side and the tree to the other side.

The first time I tried to do it, the size of the tree, within the thought bubble, caused a few headaches: I accidentally dropped it over the flower while swapping them, and they got linked.

To sort this problem, I apologized and told them that I had made a mistake. And I also told them I was going to do everything again.

Therefore, I left the thought bubble and vacuumed the robot. I picked up another. Removed the tree from the box and made it smaller, using the pump. While I was doing this, they said (regarding the pump) "That's the P for Pai[325]". "Yes," I answered, but it is here because it is also the P for Pequeno ("small"). Are you noticing that the tree is now small? This way it's easier to move it."

Afterwards, with the small tree in the box, Z gave the box to the robot. Inside the thought bubble, I showed them that the mouse was now controlling the robot – which they tested. I removed the flower, and changed the tree to the hole where the flower was. Then, I placed the flower in the other hole, and explained that I was swapping places between the tree and the flower.

Leaving the robot's thought bubble, I showed them that the box they had was identical to the one on the robot's thought bubble.

Then Z gave the box to the robot, and we stayed still, watching the robot perform as it had been taught. Then we watched it raise its arms, unable to proceed.

I draw their attention to the raising arms detail, and told them to compare the box with the robot's thoughts.

I asked O: "What's the difference?" to what he answered "It's the other way around."

[324] Rosa, my wife, was the person that performed computer sessions weekly on the preschool. She also used a physical robot regularly – a Valiant Roamer (*vd*. p. 172).
[325] "Father", in Portuguese.

Then, together with O, we placed the flower and the tree in the right position. He handed the box to the robot, which again managed to swap its contents.

We were crossing the 12 o'clock morning closing time, so I ended up the session. Z knew that we would continue on Thursday. I let them know that for the next session we were going to teach the robot to swap the images not caring about their position.

I asked them if they had enjoyed it – well, at least they hadn't even hinted at being fed up – and they both said yes while nodding.

Date	Tuesday, May 23rd 2000
Participants	Z (boy, 5 years) and O (boy, 4 years).
Starting time	11h37
End time	11h58

I arrived as everybody was finishing a small morning snack – today it had started a little bit later than usual.

O immediately said "Now?" showing surprise.

I waited until they were finished, but turned on the computer while I was waiting (but left the monitor off). Windows' starting tune drew all attentions. From his table position, O started looking at the computer. Since the image didn't show up, he started to grow impatient, even though he was still eating, and started saying "Well?".

Both Z and O were eager to start, and the moment he did, Z immediately moved the mouse to the ToonTalk icon. This time, I merely held the mouse in place, after he had moved it, so that it wouldn't budge while he double-clicked.

We talked about the last session, and I was able to confirm that they both remembered what we had addressed.

O landed the chopper; Z went into the house and made Marty go away.

Since we hadn't saved the robot, we had to program ("make") it again. I let them do it on their own, from start to finish, providing only spoken assistance from time to time. Z performed the programming movements, but O was encouraging him to do them faster ("Come on, pick up the tree!").

The only action I did perform was picking up the book with images from within the larger one.

Full list of actions performed:

- Pick up and drop a robot.
- Pick up and drop a box.
- Pick up another box and drop it over the border of the first one.
- Seek a tree in the images book.
- Take it from the book.
- Set the pump on the "P" for "pequeno" ("small") and shrink the tree. (O turned the pump on.)
- Place the tree in a hole, in the double box.
- O went seeking a flower in the book, took it out and placed it in the box.
- The box was handed to the robot.
- I called to their attention the fact that the robot wasn't thinking about anything.
- Inside the robot's thoughts, Z performed the image-swapping.
- I pressed Esc, explaining that it allowed us to leave the robot's thoughts.
- We tested the robot.

We talked about how annoying it would be if we had to do this all over again next week; and how it would be convenient to save the robot in a book.

Throughout this conversation, O said: "Next week, Santa Claus will be coming[326]", and this sparked a discussion, because Z picked up on this by saying "Santa Claus doesn't exist", which was something O didn't agree with.

They asked for my opinion, which I evaded by replying "Well, as far as I'm concerned, it sure would be nice for him to exist." I focused their attention by saying "Well, we are drifting away. The computer is what we're dealing with, here."

Then, following my instructions, Z looked for an empty page on the book, and we placed the robot there.

[326] In May? Could this have been due to their ideas about "snow" in the robot's thought cloud?

Placing my hand over O's, which was hanging on to the mouse, I picked up the letter blocks and wrote "Trocas" ("Swappy"), to name the robot. They seemed puzzled about the word, when written, even though it seemed natural enough when I had spoken it.

Confirming, by placing questions and listening to their explanations, that they had understood the necessity to make the box and the robot thought match (they went as far as stating "This way it's no good, because in here [thought bubble] there's a tree and here [box] there's a tree; in here [thought bubble] is a flower and here [box] is another, but they are swapped."

I then explained that if we emptied the box contents from the robot's thoughts, it would always do the swapping, without worrying about whether the tree and the flower were on the right places or not.

This confused them, so I showed what I meant, by emptying the boxes in the thought bubble with the vacuum cleaner. Then Z handed the box to the robot, which started doing swaps repeatedly.

Although with some difficulty, Z managed to grab the robot to make it stop.

They were clearly puzzled by this behavior. And the other objects, such as the bomb and the truck, were starting to draw their attention, distracting them. I told them that we could use them at the end of the session.

I tried to explain the generalization that had been achieved by emptying the boxes, but with no success. Even placing a tree and a flower over the robot's thoughts and then moving them aside with the hand, as I was doing the explanation, was to no avail. However, close to the end an expression on Z's face left me confident regarding the possibility of putting this concept through.

To fulfill my promise, we ended up the session by launching a truck with a Swappy robot and a box with a tree and a flower. And then we activated a bomb.

Before I could do Esc, Z went into the house that the truck built, where the robot was. I concluded the session by reminding them that I would return on Monday.

Date	Monday, June 5th 2000
Participants	Z (boy, 5 years) and O (boy, 4 years).
Starting time	11h43
End time	12h00

Before beginning the session, I booted the computer and arranged on the floor of the preschool the following physical material: two light-blue square plastic baskets, whose proportions were similar to the ToonTalk boxes; an A4 sheet of paper on which I drew a thought bubble with a ToonTalk box; two paper squares, one with a tree, another with a flower, attached to the thought bubble's box with scotch tape; two empty A5 sheets of paper (half of a A4 sheet); six markers: 2 black, 2 yellow and 2 green (the drawings are shown in Figure 249, on p. 329).

This time, it was O's turn to choose what the figurine would look like – he chose a bald one. We went through the previous actions: Z showed lots of ease on moving the chopper – so at ease that he got lost in the city, and I had to suggest him to rise high enough so that he could see where the houses were.

We fetched the Swappy robot – at first they told me it was in the toolbox, but after I replied "no, that's where we keep robots that we haven't taught yet. Where is the robot we taught?" they immediately remembered that it was in the notebook.

They browsed the notebook, picked up the robot and dropped it. Afterwards, with no difficulty whatsoever, they built a two-hole box. I then fetched a tree and a flower from the images notebook, and they both used them to try out the robot's operation. Without major hurdles, we repeated the experiments of handing the robot a box with a tree and a flower, both with right positions and with swapped positions. At this moment, O realized that all the other kids had gone to play in the yard, and wanted to join them. I told him that right now we were all three there, together, and so he should stay. "If you get up and only Z remains here, he will learn more than you will. Is that what you want to do?", but he got up anyway and started to go out "to see the other kids". I then said, with a calm tone, "go then, but then I will be a bit crossed with you, for I would have liked you to stay here." He then came back, and made no more motion to leave.

I then said "do you both remember what was puzzling you the last day?" – They couldn't manage to remember. I then went through the several bits again: they remembered the vacuum part. So, I vacuumed the contents of the boxes in the robot's thought bubble. We confirmed that when the boxes were empty, the robot would swap the tree and the flower incessantly, no matter the position of the contents.

Just like in the previous session, this behavior puzzled them.

To avoid imposing the concept, by repeating it too much, I initiated the prepared experiment: "Let's play robot! Who wants to be the robot?" – They both did, so we decided by taking a draw with chalk, just like we had done previously. Z got lucky again, so he was the first to play the robot.

They both got very amused and excited with the play.

I began by drawing a tree in on of the A5 sheets, while O was drawing a flower on another.

Then, O (following my instructions) placed each of the sheets of paper in a different basket. Their positions weren't matching the robot's (Z's) "thoughts".

Z assumed they were in the right position, because he was seeing the baskets from the opposite side: from his point of view, they were in the right position!

I handled this issue by placing myself in the midst of the handing process: I was receiving the boxes from O and delivering them to Z, by exchanging them on the process – this would ensure that they both shared the same point of view.

Z understood the process quite well. When I took the pictures from the robot's thought, Z immediately started to swap the images continuously. But he hadn't realized that he should look at the though bubble, in order to check whether after a swap operation the status was valid or not: he simply had retained that when the boxes were empty the robot would be swapping without stopping.

After a few swaps, and just like grabbing a robot to stop him, I picked up Z, by grabbing him under his arms, and "dropped" him a bit to the side. While doing so, I said "I'm going to grab the robot to make it stop." They loved this!

Now, with O playing the robot, I was careful to place both Z and the robot on the same side of the baskets, to avoid the problem of mixing left and right sides. O didn't realize at first whether he could or could not swap the contents of the baskets, but I kept reminding him to "look at the robot's thoughts ", and this way he managed it.

Performing the no-images situation, he first thought that he couldn't do the swap – because the thought and the baskets didn't match – but did it when Z encouraged him and I told him so. He did it, but I'm pretty sure he didn't grasp the concept (I decided to let him perform the movement anyway, so that he didn't feel his performance to be incomplete.)

Z then immediately grabbed him and lifted him, in order to "stop the robot." And there they were, quite amused, grabbing each other.

In order to conclude, we went to the computer again, and repeated the process.

In seeing that I couldn't go very far on this line of action, I opted to change the subject; I then announced that for the next session we would learn how to handle birds – and to leave them a little bit eager, I let them pick up a nest and drop it, so that they could see a bird.

Date Thursday, July 8th, 2000
Participants Z (boy, 5 years) and O (boy, 4 years).
Starting time 14h23
End time 14h45

This session took place in the afternoon, because in the morning the children had been visiting the firemen barracks.

They were very interested in having the session, because they both took their chairs to the computer, the moment I arrived, and Z turned on the monitor. When the system booted up, Z immediately launched ToonTalk. This time, I didn't press the "Play" button – I told Z to do it.

I could see that O was following quite well, because he mentioned, upon seeing the selection dialog, and without my intervention, that he wanted the bald figure.

Throughout the entire session, Z took the initiative, and led the operations. Once in a while, I spurred O into acting, which he did, but it was very hard for him, due to the small size of his hand. However, Z used all opportunities to pick up the mouse and lead all operations.

However, O wouldn't fall behind or get annoyed: he would let Z handle everything, and then would assume a command stance: "pick up that", "I want to hand the truck to the bird", "come on, into the nest", etc. I believe that they can cooperate really, really well, because neither of them "closes" himself inside a shell, they both communicate and act together.

Z landed the chopper, after a small tour, and entered a house. O took the figure outside the house, but brought it back in, and made it kneel following my instructions – he still has difficulties due to his small hand. On this session, as I had arranged with them on Monday, I introduced them to the birds.

Z placed a nest on the ground, from which a bird popped out. I then explained that we can give square things to the birds, and they will take them to their nests. So Z picked up a box and gave it to the bird. This way we confirmed this behavior. Afterwards, by following O's instructions, we tried with a truck: it didn't work (it didn't have a square shape). So that they could better see the bird scratching its head, I shrunk the truck with the pump – and just like it had happened on the previous times that we had used it, Z and O started saying "it's the P for Pai ("father"), for Pata ("female duck")"... This time, they were quite inventive regarding words beginning with P.

The truck was again given – this time smaller – to the bird. And this time we could clearly see the bird scratching its head, which was similar to the arm-raising of robots. I noted this to them.

O then wanted to try with the scales. He did it, and to my amazement it worked[327]! I then told them that they could give to the birds square objects or "drawings", but I knew I had to delve deeper on this explanation, later.

At this time, an issue I worked with them, quite a bit, was the fact that the birds always place what they are given under whatever they already have in the nest! A box, and then a pair of scales, had been given to the bird, and we confirmed that these were indeed under the box. This experiment was repeated, with some combinations of box and scales, which I used to ask from time to time "And this time, if we hand it the scales, will they be placed over or underneath?" Even though they haven't memorized this, I believe (from the promptness of their replies) that they understood the basic idea.

Z wanted to give the bird a nest. It didn't work, but the egg on the nest hatched and we now had two birds. Taking advantage of this situation, we started giving different things to each bird, in order to realize that a bird always takes to his nest what we hand it.

Once, we were on the top left of the screen and brought the birds far down, away from the nests. Then, I told Z to fetch an image from the book – "A pretty one," he said – and I helped him select the book, reminding him that the "-" (minus) key is used to flip back through the pages (I knew that at the end of the notebook was where one would find the cutest pictures).

Z chose one of the hands. Since this was too big, we picked up the pump to make it smaller.

Then, accidentally, Z picked up the magic wand. But still I told him what it was. The word "magic" aroused interest and admiration. I made the figurine drop it, and pointed the hand to the "C", asking Z: "Do you know what letter this is?" – "Yes I do," said Z, "it's the C for Cláudia!" I said his answer was indeed right and told him it was also the "C" for "Copiar" ("copy"). And I also told him that the wand could make copies of anything. Although this seemed a fairly common statement to me, the thing is they loved it! They said at once: "Oh...! Cool!" with quite some amazement.

And so we started a session of box copying and side-by-side placement. Z wanted, in his words, to "glue" two boxes, like he did in the previous session, but since he placed one over the other (not over the border), the second box got into the first. But what mattered was copying, so we kept on and copied the hand that Z had taken out of the notebook.

Then, O said that we could have gone to the book to pick up another hand, because the book was still open on the right page.

"That's true," I said, "but that would be a big hand, and the one we have here is already small."

He didn't agree, so I said "In that case, let's make an experiment: get a hand from the book."

He fetched one, and immediately realized it was too big. "See? From the book you always get them big, we have to shrink them out here afterwards," I said.

In the meantime, I realized that this session was getting quite long, so I suggested that we should use the birds to tidy up everything: by giving the objects to a bird, it would carry them to its nest. There was always a waiting period for the bird to return, because it had to place what we gave it underneath whatever was in the nest (and the nest was quite high, far away out of sight).

Almost everything was handed by Z to a bird, but O did hand at least one of the scales.

It then occurred to me that we could use both birds to tidy up faster, making each one carry things to their nest. Z loved the idea, and he executed it: this was their first experience with parallel tasks.

Now that the house was tidy, I had some difficulty ending the session, because they wanted to go on: O wanted to put a robot and a bomb on board a truck. It didn't work at first – you can't put a bomb directly on a truck. In the meantime, Z picked up and dropped a bomb and tried to activate it with the space bar, without holding it. I explained that he had to be holding the bomb for the space bar to work. So he did, and the house explosion was the end of the session – not before (without any instruction or suggestion from me) they had placed the figurine inside the chopper and lifted off – just like we had done in the previous session.

[327] I had read the instructions on how only images and pads, all square-shaped things, would work with birds, and had checked that bombs and trucks and the like wouldn't work, but I hadn't extensively tried out every single available item.

Date	Monday, June 12th, 2000
Participants	Z (boy, 5 years)
Starting time	11h50
End time	12h00

I started by setting up the computer that meanwhile had been taken to the University computer centre for repairs – it still refused to boot, so I opened it up and sorted the problem out – a disk cable that was way too stretched was the problem, so I changed its position a bit, so that no random disk or CD-ROM problems would arise. The only tools available for this job were a screwdriver that was way too large and some pliers. All this meant that I ended up with little time for the actual session.

O was not present, anyway; so I opted to simply let Z play around a little in ToonTalk, in order for him not to get much more practice than O.

Z is now almost autonomous: his only difficulty is performing the double-click to activate the program. From then on, he clicks the play button, chooses one of the figurine buttons and navigates quite well within ToonTalk.

Since we had been saving ToonTalk's state from session to session, the moment we entered a house we found the Swappy robot working, swapping on and on.

I asked Z to stop it by holding it. Then, using the vacuum, we cleaned up the house.

We worked solely this behavior: essaying the placement of a nest in a box and giving the box to the robot.

Z picked up a box, but chose to copy it with the wand, instead of picking up another one. After building a two-hole box, I told him "we can also use the wand on the robot's thoughts", and showed him what I meant, by copying a tree and a flower.

Then, we placed the tree on a hole and the nest on another, and handed the box to the robot.

At first, Z thought that the robot was rejecting it, because the left arm was raised. But I called his attention to the fact that it hadn't dropped the box.

I said: "It's waiting to find out what will the bird take to the nest. Try giving it a flower."

"Is the flower a drawing?" – asked Z (he remembered the rule I had instituted in a previous session, saying that we can only give square things and drawings to the birds – well, I can't be sure he remembered it all, but at least he did remember the "drawings" bit).

Since I said it was, he gave the flower to the bird and we watched as the robot performed the swapping. I concluded the session, explaining that for the next one, we would set robots exchanging things among them, using birds for assistance.

Date	Thursday, June 15th, 2000
Participants	Z (boy, 5 years)
Starting time	11h35
End time	12h00

O, again, wasn't attending the preschool. Therefore, I moved on with Z, on the theme of combining the use of robots and birds. We started by going over the working of the Swappy robot, after we had placed a nest in one of the holes.

In order to get the result I wanted, we had to start from scratch, and therefore I asked Z to clean up the working area, which he did with the vacuum.

I then proposed that Z should make a robot that would receive a flower in a box and a bird in another, and give the flower to the bird!

He did as instructed, with a trick he had devised before: he picked up and dropped a box; but instead of repeating the procedure, he used the wand to create a copy.

Programming the robot was easy; it was a simple matter of picking the flower and giving it to the bird.

The robot was tested, without further complication.

I then suggested that we picked up another nest – which Z did. Then, I instructed him to place the nest where the flower was: "What if we now gave the flower to the bird from the new nest, what will happen?" Says Z: "It will take the flower to his nest." "And then, Z?" – I asked. He replied: "The robot will give the flower to this bird, which will take it to that nest."

We tried it, and it worked just as Z had described it.

We were almost finishing the session, so I proposed that we should get another robot, just like this one – Z copied it with the wand – and we set the robots with crossed birds and nests; *i.e.*, each robot would have a bird and the nest from the bird on the other robot.

Then, to test this team of robots, we placed a flower in each nest – and right away the robots started to trade flowers among them, by passing them from bird to bird.

Z loved it, and we called the preschool teacher to watch. Z was enjoying what he had built, and managed to explain what was going on.

Araucária preschool

Date Monday, May 30th 2000
Participants M (boy, 4 years).
Starting time 11h00
End time 11h17

The preschool educator, Edite, told me that only M was available for the session, because the other kids that should be present hadn't come (they had participated on a tour the previous day, which arrived quite late – they probably were sleeping a few more hours today, she said).

I had already turned the computer on and installed ToonTalk.

I asked M if he knew who I was – he said no, shyly.

He was very shy throughout the entire session, particularly regarding conversation, but never hesitated when it came to use the mouse.

"But you do know Rosa, the computer teacher," I asked. He said he did, so I explained (to make him feel more comfortable): "I'm Leonel, Rosa's boyfriend." I carried on: "I'm here to show you a game, in order to see whether you like to play with it or not."

I also explained that two other boys, R and L, were supposed to be at this session. He told me that they would only arrive in the afternoon.

I told him that was all right, because we would all be present next week. But for today, it would be just us.

I showed him the ToonTalk icon. "Do you know how to start a program?" I asked. "No," he said. "OK then. Take the mouse pointer over the program icon" – this he did. "Now you've got to press the button twice." He was too slow for the action to be interpreted as a double-click. "You got to do it faster," I said. The second time round, he managed to do it.

Upon the ToonTalk starting window, I said: "This is where we're going to play. I'll press a button, to let it know we want to play."

I explained to him that we could choose the style of figure we wanted to play with: long hair, with a hat or bald. He pointed to the "Com chapéu" (with a hat) button.

I asked him if he knew what "this" was, as I pointed to the helicopter on the screen. He correctly identified it as being a chopper. I then asked "what about that, down below?" – "Those are houses," he replied.

I encouraged him to move the mouse, and he showed no difficulty controlling the chopper. After explaining what the buttons were for (go up or down), he also controlled the chopper easily using these movements.

Since he was quite at ease, I told him to land near the houses. First time around, the perspective fooled him, and he almost landed a street down too far away, but following my indications he went up again and finally landed on the right street.

He moved the figurine and Tooly, without any difficulty, and following a suggestion of mine, entered the first house.

I then explained him that the Martian was there to help us when play alone. But that since now I was there, the Martian wasn't required, and I was going to send him away.

After I pressed the F1 button, he moved the mouse at will, quite at ease: he even picked up the word "Imagens" (images) from the notebook. I therefore told him to pick up a box, without any further delay. My intention was to make him build a two-hole box, but before I could say anything he picked up a letter block.

Well, since he was holding a letter, I told him to drop it in the box. I must say that it was not necessary to tell him that the mouse buttons were used to get objects – he immediately assumed it (notice that he had originally taken the initiative of picking up the word "imagens"). I merely told him that the mouse buttons were also used to drop things.

After placing the letter in the box, he was idling, not knowing what to do. Then I told him to pick up a box, which we did together, my hand over his. And then we picked another box. Always keeping my hand over his, I explained that I was going to join both boxes, by dropping one on top of the other. The mouse with the hammer came along – since the boxes were next to the top of the screen, there was enough time for him to observe the mouse.

I noticed that the initial enthusiasm was being replaced by some frustration, because he had no idea of what to do. I then explained that what we could do on this game was playing by training robots, teaching them.

I picked up a robot and showed it to him: this refreshed his interest.

Then, I quickly picked up the flower and tree images, and placed one on each side of the robot. I put forward that we could train the robot to exchange the positions of these images.

I then told him to put the flower in one of the holes of the box and the tree in the other. Although the tree was in full size (huge, compared with the box), the fact that the box was near the top of the screen made it easy to place the tree in the left hole of the box.

"Can you figure how to teach the robot?" I asked. "No", he replied. I said: "We give him the box and then we teach him how to handle it."

I then handed the box to the robot and we watched as everything went into the robot's thought bubble.

Inside the robot's thoughts, I told M to move the robot, by using the mouse, and to make it swap the places of the tree and flower. A small detail: he removed the tree, then the flower, and then placed them on their final positions – he didn't place the flower directly on the tree's vacant spot.

"There," I said, "now the robot knows how to make the tree and the flower swap positions." We left the thought bubble, and I told M to hand the box to the robot. We then watched as the robot performed as it had been taught. When it stopped, I placed the swapped box next to the thought bubble, and asked: "Why did it stop swapping?" He correctly identified the problem in that the positions of the tree and flower didn't match what the robot expected. Then we corrected this, placing them in the desired position, and he again handed the box to the robot, this time without my hand to guide him. We had some problems, because the box got associated with the one that had the letter "A", which was near the robot. Therefore I had to split the resulting box. But then we managed to correctly hand the box to the robot.

He was quite amused now, so I asked him what he would like to teach the robot to do. Seeing as he was not sure, I told him we could try and swap other things – and immediately he chose the truck and the bomb.

"Let's train another robot, then," I said. I picked another robot, but he couldn't remember how to start the training. "Well, we have to make a box that can hold the truck and the bomb." This he did. Then he went and picked another robot. However, this one was accidentally laid over the box that had the truck inside, and therefore ended on the truck. In an instant, the truck image was gone. I thought a process had been triggered, so I said: "Look, we dropped it on the truck and it went away. Let's get another truck." But we couldn't get the truck into the box, and I realized that the previous truck and robot would probably still be inside it, although invisible due to some environment bug. By picking up the apparently empty space with the hand, this was confirmed: a truck was being held, the robot on top of it "shooting" straight to the top of the screen, out of sight. I dropped the truck on the box and the robot returned. I picked up the truck again and all events occurred in the exact same way.

I decided to vacuum the truck and told M that we had to use the other truck he had picked up. This he did, but all this confusion had been too much for him, something I confirmed, because he nodded when I asked, "you're fed up, right?" Therefore, I told him we would stop for today. But a mere 3 seconds later, he was again interested in ToonTalk, and therefore I took the opportunity to train the robot for swapping trucks and bombs, which we tested.

I did this, because by having lots of robots, each one able to swap a specific pair of objects, the usefulness of generic robots could become more obvious. And this was something I intended to try on the following sessions – inspired by the ToonTalk tutorial/examples.

M didn't want to stop, but it was convenient to do so, in order to avoid over-extending the session, and therefore I suggested that we could simply finish by cleaning up the workspace with the vacuum cleaner. He then picked it up, pressed the space bar (after I told him that it turned the vacuum cleaner on and off), and vacuumed up everything, in a joyful manner. He even started vacuuming truck after truck from the toolbox, until I told him that there were too many trucks there, and that we would never be able to vacuum them all.

Once the workspace was clean, I finished the session. We agreed that next week he would be able to teach R and L how to program robots.

I asked him what he had enjoyed the most, and he said "Everything."

I then took him to the main room, and exchanged a few impressions with the educator (it was then that I found out that M, in spite of his size, had turned 4 just two months before).

Date	Thursday, June 15th 2000
Participants	M (boy, 4 years), R (boy, 5 years).
Starting time	10h30
End time	11h15

R hadn't been at the previous session. So, for this session, he was the one choosing the figurine style (bald) and was the first operator. He correctly identified the chopper, but not the houses. However, when I told him that what he was seeing were houses (roofs of houses), this statement was perfectly acceptable to him.

R landed the chopper quite easily, and immediately entered a house.

Overall, the mouse at this preschool is, I believe, the most adequate of the ones found in the three preschools[328], because it has a left button that is large, but not huge, and a smaller right button – this allows us to use separate functions per button if necessary (when flying the chopper, for instance), but prevents accidental clicks.

I asked M to explain what kind of task we could perform – he was a bit confused, but small hints allowed him to move on. We started by getting hold of a robot. I explained that in the bubble were the robot's thoughts – *i.e.*, what was inside the robot's mind/head.

I also explained that M had already learned how to teach a robot to swap a tree and a flower.

So, R (following my indications as well as M's) grabbed a box, and then placed another one over it, resulting in a double box.

I opened the images notebook on the page containing the tree, and then R picked up the tree. Since this was too big, I told them (M hadn't seen this yet) that we could shrink the tree, with the "air-removing pump". "It bears the G for Grande" (big), I said, "but you've got to press that button until you get the P for Pequeno" (little), I added. "Press on." The letters cycled, until "P" showed up, and then I said, "That's the one! P for Pequeno." Then R grabbed the pump and shrunk the tree.

Then he placed the tree in the box, and we proceeded to pick up the flower.

The teaching of the robot went on without issues – R understood very quickly the instructions he was getting and had no problem executing them with the mouse.

We tested the robot, and confirmed that it did work. Seeing that the robot stopped after performing the swap, I asked them if they could figure out why it had stopped. The answer was not given right away, but when I placed the box near the robot's thoughts, they answered immediately. The problem had been identified.

M was eager to use ToonTalk, and therefore he was the one that now programmed a robot to swap the tree with the flower starting from the positions they had after the original swap. We also tested this robot.

Now that we had two robots, we tried giving the box to one of them (which would swap the contents), then to the other robot, and so on.

I then asked, "They are, in fact, doing the same task, isn't it? They are simply too picky, right?" There was an overall consensus on this. "What we should do is teach a robot to not care about the position of the flower and the tree!" I said. Then I asked: "Do have any idea about how we could do this?" The answer was: "No.". So I said: "We simply vacuum the tree and the flower from the robot's mind! This way, he still knows how to swap, but no longer checks to see if the tree and the flower are in the right positions."

We performed this procedure – although they were the ones turning the vacuum on and off, I was careful to tell them to turn it off right away[329], in order to vacuum only the tree and the flower, not the box.

This worked, as we expected. I was indeed surprised that, unlike what had happened in the São Pedro Parque preschool, this procedure raised no confusion or doubt; they reacted as if it was something obvious.

While I was finishing this session, I told them that, on the next one, we were going to work with birds – and as an appetizer, we performed the simple experiment of handing a box to a bird, to see that it carried it to its nest.

Then, I finished the session.

Date	Tuesday, June 20th 2000
Participants	M (boy, 4 years), R (boy, 5 years).
Starting time	10h45
End time	11h25

After I arrived, the moment M and R were called, they darted off into the computer room. And another kid who hadn't been appointed by the educator wanted to go, too.

We turned on the computer and M said, immediately: "R will start."

R started ToonTalk, following M's indication, quite easily. He chose the bald figure again, and the session was initiated.

I asked if they now knew how birds worked. They said they did. "And what do birds do, then?" I asked. "They take things to their nests," they answered.

[328] In 2000: Araucária, As Árvores, São Pedro Parque.

[329] In the current version of ToonTalk, pressing the space bar only makes the vacuum cleaner work once. But at the time, it would stay on until the user pressed the space bar again to turn it off (or dropped it).

“OK, then you already know how to work with birds and robots,” I said.

"What can robots do?” I asked.

"They swap things!” R said.

"Yes, but they can do lots of other things!” I said.

"Can you come up with something else for us to teach the robots?” I asked.

They couldn’t imagine anything, but did decide they wanted to use the pump to enlarge things. I took this cue and suggested that we could teach a robot how to enlarge things.

R initiated the programming: for starters, I reminded them that we needed a box with something in it. He then picked up a box. Then M fetched the images book and picked a chopper from it. R placed the chopper inside the box. I suggested that the robot could enlarge the chopper and give it to a bird. So, they picked up a bird, joined another box to the one they had, and placed the bird inside. Then they gave this double box to the robot.

R taught the robot by taking the chopper from the box, enlarging it a bit, taking the bird from the box and giving the chopper to the bird. Then he placed the bird back in the box. When we tested this robot, it didn’t work: when it started pumping up the chopper, this action yielded no effect[330], and the robot kept pumping, never moving on.

I then said that the problem could be the chopper, and that we should try with something else. I was merely considering that perhaps the problem was the fact the chopper was an animated image.

I asked: "can you remember what we can do so that the robot isn’t so picky?" R said at once that we could achieve that by erasing the chopper from the robot’s mind. M was a bit lost by now, focusing on the images book, looking for a tree. He found it and laid it on an empty space in the floor. Then he vacuumed the chopper and placed the tree in the box. And then he handed the box to the robot.

"Why didn’t he accept it, M?" – Before he could answer, R said: "Because the box is different. There [he pointed to the robot’s thought bubble] is a bird and a chopper, but here [he pointed to the robot’s box] is a bird and a tree."

"And how can we make the robot less picky?" They answered at once, together (although R showed more confidence), saying that we had to erase the chopper from the robot’s mind, which M did. But even with a tree instead of a chopper, it still didn’t work.

We then erased the robot, and following my indications M, without help from R, programmed a robot that would take a box with a bird and a robot, would place the robot within a box and give it to the bird, and then would place the bird back in place. Then, by his own initiative, he decided to pick a new robot and place it on the empty box hole.

Since this was a chance to produce continuous action, I told him to pick up the bird’s nest and take it far away, in one of the corners of the house. Returning to where the robot was, we started it. As expected, it started functioning continuously, sending boxes of robots into the bird’s nest.

As we watched this process, with the bald figure up, M left the house for a moment. Of course, this moment was enough for the nest to get lots and lots of robots. I then tried to explain them the difference in quickness between robots working when we watch them or not, but this proved impossible, for they were completely distracted, particularly M. I still managed to show them the huge number of boxed with robots that we had in the nest, but M was already vacuuming things up, laying birds and robots on the floor, and so on.

I then said it was time to conclude the session. In the meantime, I explained to them how the wand worked, and this proved a success. They started copying all sorts of things – the pump, the vacuum, etc.

A curious thing was: while copying the bird’s nest, R asked "Will the bird take the boxes to both nests, this way?" – I was not about to lose this opportunity, so I spurred him so that he would try it, and he did try – the moment the box was handed to the “producer” robot, the bird was perfectly visible splitting in two to perform the double delivery.

From then on, I merely tried to find out what they wanted to do – they both wanted to pick up letters and write their names, and also handle numbers (so I showed them that the hammer mouse could be used to add up numbers). I showed them that the vacuum could be used to spit things, not just vacuum them, and then they told me they wanted to experiment with the other letter on the vacuum control ("L", for “Limpar” – erase).

They were quite entertained, doing copies with the wand, vacuuming, spitting, etc. When I had the chance, at a quieter moment, I then really finished the session.

[330] This would have worked in the current ToonTalk version. At the time, robot’s couldn’t be programmed to enlarge or shrink objects, something I wasn’t aware of.

As Árvores preschool

Date Friday, June 9th 2000
Participants J (boy, 4 years), M (girl, 4 years), S (girl, 5 years).
Starting time 15h20
End time 15h50

The preschool educator, Gina (with whom I had spoken previously regarding the sessions), was not present when I arrived at the preschool room. I set up the computer, and called J and M, in order to introduce myself and explain that we were going to play with a new game, and that I intended to see whether they liked the game or not. They told me they already knew about that, and they also told me that "our parents were told about it, and they told us we were going to do something on the computer, both of us and also S". Well, since Gina in fact had mentioned in passing the possibility of including other children in this activity, besides J and M, I assumed this information to be correct and therefore called S and we started the session. Later on, I found out that what the children had told me was not correct.

Due to limited space near the computer, and to the height of the table, chairs were not practical. So, I used the solution employed in the computer literacy sessions that take place weekly on this preschool: to do everything standing, without chairs. In order to avoid being at a much higher level than the children, I sat on a table, by the computer.

The mouse on this computer only has two buttons, which cover a mouse area much larger than the one occupied by the buttons on the mice of the other preschools[331]. Therefore, children have an easier time pressing them.

This even turned out to be a bit counterproductive: upon seeing ToonTalk, the children (except S) wanted to use the mouse, to the point of placing their hands on it simultaneously, which resulted on almost random mouse clicks.

Initially, we drafted who would choose the figurine style, using three distinct color pens: each child picked up one, I scrambled them and dropped them on the ground; the furthest to the left was deemed the winning one.

The chopper was identified as such by M and J, but not by S. All of them, one at a time, had the chance to move it, and make it go up as well as down. The perspective used raised some difficulties on selection of landing place: they were always landing on the street above the houses, being tricked by the perspective.

After several consecutive "kneelings" of the figurine on the street, which I always cancelled with the Esc key, we managed to enter a house. At this moment, they were starting to exchange control more often – Maria was ruling, most of the time. They were all quite excited, and (or so it seemed to me) without enough focus for any explanation about the method of robot programming. Therefore, I decided that indeed the best thing would be for us to simply explore the overall functioning of the software.

So, I started by getting the ball from the images notebook, and showed them how to use the pump to make it big. In order to simplify operation, the keyboard was shoved to a side and laid vertically, so that the letters would not draw attention, but the Esc key and the space bar would still be easily accessible.

They immediately started using the pump to make a robot big, very big (until the screen could only present its legs), and also a very big pair of scales.

Afterwards, I taught them how to use the vacuum to clean up everything.

Without giving me a moment to breathe, they picked up a nest, and I took the opportunity to explain to them how birds worked. We tried giving all sorts of objects to the bird and watched as it took them to its nest. We created another bird and this allowed the children to confirm that each bird carries objects to its own nest. A bird was then taken to another house, in order to make clear that distance from the nest is no obstacle to a bird's operation. From this point on, the children started creating bird after bird... We already had 7 or 8 on the screen!

The children were very entertained, but half an hour had elapsed already, so I decided to end the session. They all told me they had enjoyed it a lot, and I scheduled another session, on the following week, and said that on that session they would learn how to teach robots.

[331] São Pedro Parque and Araucária.

Date Friday, June 16th 2000
Participants J (boy, 4 years), M (girl, 4 years), S (girl, 5 years).
Starting time 10h45
End time 11h20

There is a serious hindrance to ToonTalk operation on this preschool: no sound effects, due to an operating system problem!

I started by asking if they remembered what we had done last week. Unsurprisingly, they all remembered having created lots of birds on the last session. I then said that today, as I had promised on the last session, they would learn how to teach robots.

J was the first to pick up the mouse. I would let them choose what they wanted to achieve.

I then asked J: "What would you like to teach this robot?" "To clean up everything", he replied – in fact, the screen was quite crammed, from the last session.

"In that case", I suggested, "we can start by teaching it to vacuum. In order to teach it, you have to hand it a box with something for it to vacuum."

J did exactly this, giving to the robot a box with a nest.

Inside the robot's thoughts, I told J that he could get hold of the vacuum in order to vacuum the nest. This he did, but started by vacuuming the toolbox! Then, besides vacuuming the nest, he also vacuumed the box that he had handed to the robot.

After this was done, I pressed Esc to leave the robot's thoughts, but ToonTalk crashed.

After rebooting the computer, we repeated this operation, taking care to vacuum only the nest, this time. The robot had been taught, then tested, and worked perfectly – great!

I then asked J to put the robot in the notebook – he did it immediately, without giving me the chance to choose a page! He stored it on page 2, on top of the picture book, which resulted[332] in preventing me from accessing it!

As these events developed, and also afterwards, there was a striking difference on S's behavior regarding the previous session: this time, she was taking active part on all tasks, taking initiatives. For instance, she said that the hand should be moved higher, towards the desired objects; suggesting how things should be done, and even going as far as saying "I want to try it too." What a difference regarding the previous session, on which she hadn't said a word!

Now M, what would she choose to do? "I want to make a robot to cut trees." I silently reckoned that there was the possibility of playing with pictures of different size, combining them (dropping the smaller on top of the trees) in order to simulate a tree being cut. But how would I get hold of a tree picture? J, by saving his robot on top of the images notebook, had prevented all access to it[333]!

Since getting hold of a tree wasn't possible at the time, I put forward that we could make the robot play a sound. I picked up the sounds notebook and started reading the names of the sounds. M told me she wanted the "magic" sound, so I picked up "Magia" – this was when I found out that while synthesized speech worked, the sound effects didn't.

As a last attempt, I suggested that we could teach a robot how to change the positions of a truck with a pair of scales – M agreed, fortunately, and we proceeded with this task, without major surprises.

Having two robots ready, M and J darted off to the playground, so fast I didn't had the time to call them back for any final chat, but S stayed – she then took her place by the computer. I asked her what would she like to do, and she let me know that she wanted to make a robot that would vacuum bombs. I helped her achieve this, and then we both tested the robot operation.

Since M and J were already absent, I was able to conclude the session.

[332] This environment bug has since been corrected.
[333] I knew I could repair this by copying and pasting an images notebook from another user, but that would ruin the flow of the activity.

Date	Monday, June 19th 2000
Participants	J (boy, 4 years), M (girl, 4 years), S (girl, 5 years).
Starting time	10h30
End time	10h50

This session didn't actually took place as intended, because all the kids were practicing for the several plays and games that would take place during the end-of-year party to be held on that afternoon.

We did initiate the session, but there were already cakes and refreshments available, and all the kids were too excited to actually be able to focus on ToonTalk.

But I still managed to take these notes:

• I imported a bunny picture, because M wanted to play with a rabbit picture on ToonTalk. After importing the picture, by dropping the filename on the appropriate sensor, I taught them how to use the wand, but this ended up copying the sensor, not its contents. Given the situation, I had no opportunity to find out how to overcome this[334], and so I decided to let them go rehearse the plays and eat cake with all the other children.

• S, even though I already had dismissed them, decided to stay, in order to play with ToonTalk a bit. She mainly picked up nests to create birds, and moved several toolbox objects around, using the pump to enlarge and reduce them – I concluded the session after letting her play around in this manner a bit.

[334] Only later did I learn that one needs to vacuum and spit the contents of the sensor, not copy them.

Annex IV
—
Details of sessions conducted between February-June 2001

The children

During these sessions, the children were divided into 5 different groups. In each group, all the children took part in computer activities simultaneously, in pairs.

GROUP 1
A pair of three-year olds. The sessions took place at a specific university room (setting 2, below). One of the parents would take the children to that room, but wouldn't be present during the sessions. From the fourth session onwards, the preschool decided to replace 3-year old R with 4-year old P, as detailed further ahead in this annex. These children were selected by the preschool teacher at the SPP preschool.
C (girl, 3 years); R (girl, 3 years); P (girl, 4 years)

GROUP 2
Six children aged mostly 5 (there was a 4-year old, and a girl who had just turned 6). A specific room was set up for the sessions, in the University (setting 1, below). Children were brought in by their parents, who weren't be present during the sessions. The children were paired, although the pairings were changed informally, depending on the activities taking place. These children came from the AA preschool. Those marked with an * were those involved in ToonTalk sessions, the previous year.
J* (boy, 5 years); M* (girl, 5 years); S* (girl, 6 years); L (girl, 4 years); G (boy, 5 years); R (boy, 5 years)

GROUP 3
Six children of mixed age (4-year olds and 5-year olds). These sessions also took place in setting 1, under conditions identical to group 2, above. The children came from the SPP preschool. None of the children involved in the sessions the previous year was able to attend.
A (girl, 5 years); B (boy, 4 years); D (boy, 4 years); J (girl, 5 years); M (girl, 4 years); R (boy, 4 years).

GROUP 4
Six children of mixed age (4-year olds and 5-year olds). These sessions also took place in setting 1, under conditions identical to groups 2 and 3, above. The children came from the SPP preschool. None of the children involved in the sessions the previous year were able to attend.
A (boy, 5 years); AA (girl, 4 years); DA (boy, 4 years); DU (boy, 5 years); H (boy, 5 years); J (girl, 4 years);

GROUP 5
A single three-year old. Some of the sessions took place at a specific university room (setting 3, below), while other took place in setting 1, usually before the sessions with the older children. The child's mother would take her to that room, and the child would be sitting on her mother's lap while using the computer. This child came from SPP preschool.
J (girl, 3 years).

The environment

There were three basic settings, described below.

Setting 1:

A main room was prepared for the large groups (in the list above, groups 2, 3, and 4). This had a direct access from the parking lot, a few tables and chairs, and three 15" CRT monitors. The computers used were a combination of laptops and desktops, but the laptops were equipped with regular mice. Instead of using the laptop screens, these computers were connected to the CRT monitors, for better visualization from wide angles.

Setting 2:

This was a meeting room in the engineering building, at the university. It was used by group 1, from the list above. There was no direct access from the parking lot, but a parent would bring the children to the entrance of the building, from where the researcher would lead the children to this room. A laptop computer was used, equipped with a regular mouse.

Setting 3:

ToonTalk was installed on a desktop computer, at a University office also used for assessment of computer-support required by citizens with special needs. This was used in some of the sessions with the child in group 5, for the convenience of her mother, a teaching assistant whose office was located in the same building. There was direct access from the parking lot, although this was not relevant to the case, for the child would often already be inside the building, at her mother's office.

Group 1

Session 1 (February 12th, 2001):

Only C attended. Initially, she practiced the manipulation of ToonTalk elements, including the tools and images from the notebook. She wanted to make a house with a garden, which was achieved using three built-in ToonTalk images: a house façade, a colored rectangle and a flower (figure on the right).

Session 2 (February 19th, 2001):

Both C and R attended. They expressed their desire to work with "trees, fruits, and girls", took out these pictures from the notebook, and proceeded by combining and moving images around. Then I suggested that pictures might be organized in boxes, and from this they practiced putting things inside boxes and taking them out, and joining boxes.

The two children frequently need my support to handle the mouse. Usually I need to help them place their hands on the mouse properly, so that they can move it and click at the same time.

Following my instructions, Rita executed the robot-programming procedure for exchanging a tree and a flower. Then she did the same for another robot, to exchange a girl and a ball (C helped her completing this one).

The robots were tested and afterwards I placed them side by side near the boxes, which I intentionally mismatched (figure on the right[335]). Both girls were able to correctly identify to which robot each box belonged.

Session 3 (March 5th, 2001):

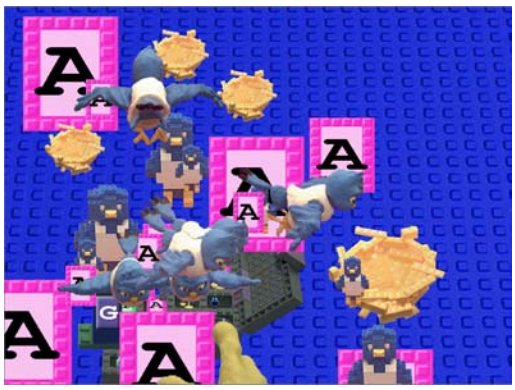
Only C attended. Due to computer problems (constant crashes for lack of disk space), I ended up only performing freehand drawing on MSPaint with C, so that she should practice mouse skills.

Session 4 (March 26th, 2001):

The preschool teacher decided to replace R, and the 4-year old P was sent instead. The girls arrived late; therefore, this session was only 30 minutes long, instead of the usual 40 to 50 minutes.

[335] In this and subsequent figures, occurrences of the children's names have been grayed out with a diagonal pattern (▨).

I suggested for C to show P how to use the bike pump. C identified it, but no longer remembered how it was used. However, she now managed to use the mouse and ToonTalk without major difficulties, selecting and picking up what she intended. After C played around with the bike pump and the ToonTalk objects for a while, P practiced using the pump to enlarge and shrink nests.



They practiced the use of the magic wand for copying.

C referred to the smaller birds as "baby birds".

The session ended after both girls tried giving the letter A to a bird, and seeing it multiply to deliver it to all the copies of its nest (figure on the right).

Session 5 (April 2nd, 2001):

Both C and P attended. I tried to leverage their use of boxes, birds, and magic wand, by proposing the programming of a robot that would "make birds". I demonstrated this: the robot would take a bird, copy it, and give the copy to the original bird, so it would be taken to the nest. The children were not able to understand this at all.

Session 6 (June 4th, 2001):

Both C and P attended. Since there was a two-month gap between sessions, I asked them if they remembered which game we were playing. They said they didn't remember. However, upon launching ToonTalk, C said she did remember it after all.

C practiced the control of the helicopter, with some difficulty: the top-down perspective fooled her and she often landed away from the houses, but P was able to keep prompting her to correct the movement, and was able to point the direction of the houses, even thought neither could see them at low altitude.

I explained how to "send away" or "call back" the Martian using the F1 key. Oddly, C didn't immediately associate the behavior with the keyboard key, and tried to push the paper keyboard overlay where the Martin was drawn, just above the F1 key.

They could not understand the text-to-speech engine at regular volume, but they did understand the speech at high volume, and ensued by repeating/responding to the Martian's speech: "Good morning/Good morning", "See you/See you". C liked this and spent a few minutes sending the Martian away and calling him back, but P was soon fed up and asked her to stop.

They both vehemently wanted to make nests, so we proceeded along that line of action. P also wanted to build houses with a truck – something which I hadn't shown any of them before, so this idea must have been passed down from the other children at the preschool, 12 of which did that kind of activity, in other groups.

They proceeded by playing around with the bird's behavior, giving them objects and figuring out where they were being taken, what kinds of objects birds carry, etc. After building houses using trucks, I proposed a game and we took a bird's nest to a different house, to try and see if the bird would be able to find its nest, still. After confirming delivery (which they found "great"), I ended the session.

Group 2

NOTE: Three children in this group had prior ToonTalk experience, albeit very limited (3 sessions), in the previous year.

Session 1 (February 14th, 2001):

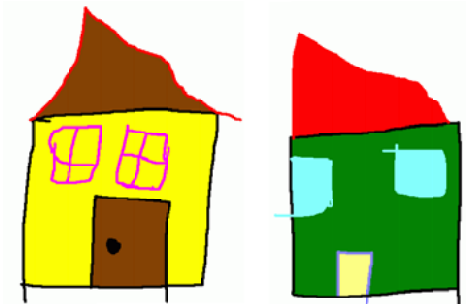
For this session, I had the assistance of Rosa Cristóvão.

The children picked up and dropped a large number of ToonTalk objects and controlled the figurine, without major difficulties. Among the lesser ones: confusion about the different functions of the mouse buttons, while controlling the helicopter, and needlessly pressing of the mouse buttons while moving the figurine, causing many unwanted "kneelings".

For helping J, M, and S remember their prior ToonTalk experience, I suggested that they might want to teach the robots how to play. J & R gave an empty box to a robot, and my suggestion was for the robot to create trucks (*i.e.*, place a truck in the empty box). I then suggested that they taught another robot how to vacuum trucks, which J did without any problems.

Then J wanted to teach another robot – but he couldn't decide on what to teach it. So I suggested that they postponed the teaching of the third robot and put the first two to play, instead.

S had programmed a robot with no instructions – it simply kept there, floating around.



L & G wanted to draw, but S & M wanted to program. M wanted to create "1000 birds." So she set up to pick them from the toolbox. I tried suggesting that she used a robot, that would create the 1000 birds by joining boxes and placing birds, but my suggestion wasn't welcomed. So I taught M how to use the magic wand and... What a whole lot of birds!

I suggested a different approach to teaching a robot for making birds: I showed them how to make a robot that, starting with a bird, would give it boxes with birds inside. This suggestion was grasped, unlike the previous one, but the result didn't convince the children: in ToonTalk a pile inside a nest looks just like a single element. So, although they could try and explore the size of it, it doesn't stand out, visually, as an enormous pile.

Meanwhile, L & G had each drawn a house in MSPaint (see the figure, above; L's on the left, G's on the right). J had been intensely experimenting with the ToonTalk environment: he had shrunken the pump, created another one, enlarged the vacuum cleaner, narrowed it, and shrunken the toolbox until it became invisible.

Session 2 (February 21st, 2001):

For this session, I had the assistance of Rosa Cristóvão.

Due to several technical problems, for the most part of the session children were playing without computers, while the technical problems were being sorted out. Afterwards, children were thirsty and I had to go fetch them some water while Rosa helped them use the computers at will: S, J, and R wanted to use ToonTalk, L didn't want to use the computers out of being tired, and M & G simply wanted to draw freehand.

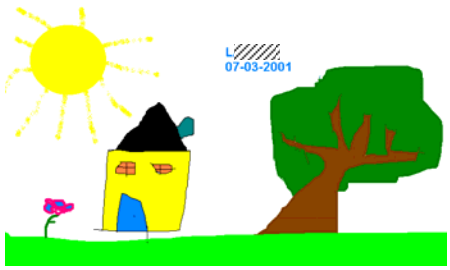
J created a large set of birds on the floor (initially by picking them from the toolbox, but then, by his own initiative, by using the wand). I believe he was making an "army", but S said it looked like a mass and J adopted that description from then on (see the figure on the right).

S declared her intention to teach a robot how to make birds. When asked "what do we need" she said "a box". Her strategy to make birds involved making the robot pick up nests from the toolbox and dropping them inside the robot's box. I suggested that she could teach the robot to join new boxes to the original one, to store the extra nests and she did this without difficulty.

Session 3 (March 7th, 2001):

For this session, I had the assistance of Fernando Lemos.

S proceeded with her desire to teach robots. But if she has a specific goal during the teaching of the robot, it is not perceivable from her actions (*i.e.*: starting with a girl's picture, drop a 1 on top of an A, put the resulting B on top of the girl's picture, hold and drop it several times, pickup any words from the main notebook, such as "Images" and "Sensors" and drop them on the girl picture, etc.). (Note: see the report from session 8, further ahead, where one can see her recounting the actions of one of her robots from a comic-strip version.) S expressed no desire whatsoever to "test" what she had taught the robot, to the point of vacuuming the contents of its thought bubble.



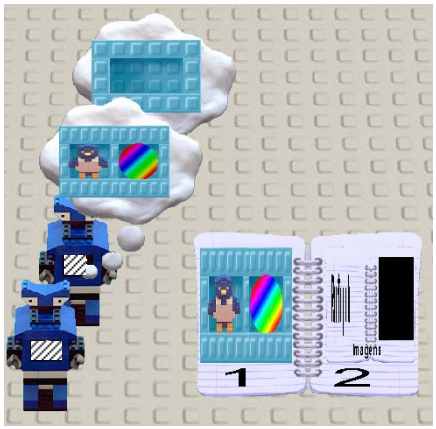
M & L, with the support of Fernando Lemos, went through the basic process of acquaintance with the system: landing the helicopter, picking up and dropping objects, vacuuming them, spitting, etc.

J was deeply involved in doing huge number of copies of elements.

G wanted to make freehand drawings, which he couldn't, since the computers were being used for other purposes, as described above. He explored the closed-lid notebooks and found that they were showing the same image presented on the CRT monitors, which interested him. S got involved in this too, and both enlisted R to try out different things: R would move the ToonTalk figurine very, very quickly, and G and S were trying to confirm whether the image was still the same in both screens. They were telling R to kneel the figurine, to move the hand... soon all children were watching this discovery.

At the end of the session, while the other children played without using the computers, J and L, each went to a computer to make freehand drawings (above, on the right side: topmost from J, bottommost from L).

Session 4 (March 29th, 2001):



Not all children attended this session, but those who did were involved in programming robots. I suggested teaching robots to write their names, something that happened in slightly different ways.

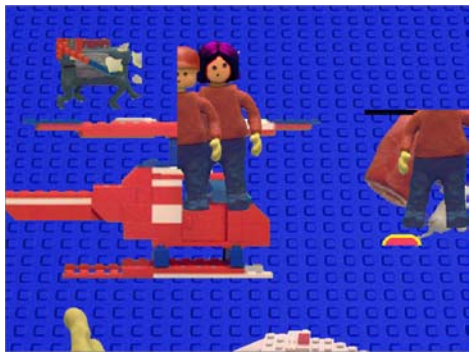
M programmed the robot shown in the comic strip of Annex I (p. 441), which picks the letter A, deletes it, types her name and drops the pad in the box.

R did it similarly, but was initially confused by the "A" in the pad taken from the toolbox, so his robot was also taught to vacuum up an extra pad he had dropped on the floor, in the process.

L used M's strategy also, but demonstrated less control and decision: she would drop the element she was working with after each step, and would have to pick it up again.

J & G did a different activity: they created a box with a bird and a ball, which I told to save in the first page of their notebook. J used this box to teach a robot how to give the ball to the bird. I suggested that G taught a robot to pick up that box from the notebook, which he did: he taught a robot how to pick up that box from the notebook and turn an empty box into a two-hole box with those contents. I then showed them that their robots could work as a team, with G's robot preparing a box for J's (see the figure above on the right).

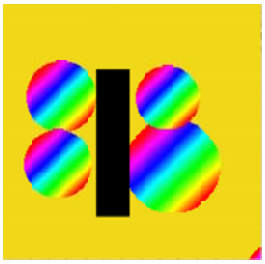
Session 5 (April 5th, 2001):



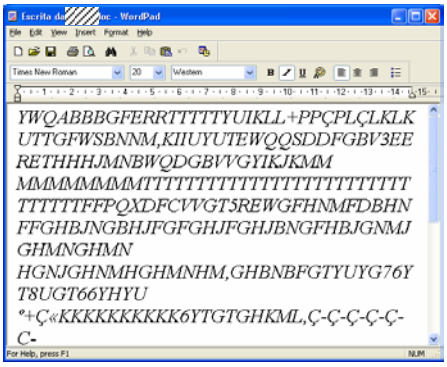
No programming in this session. But I had printed on paper their robots from the previous session and posted them on the wall, and they are eager to admire them.

J discovered how to duplicate the wand, and started repeating the process. Then he decided to enlarge a vacuum cleaner until it filled up the screen, and I successfully asked him to show L how to use ToonTalk items, since she was very focused on his work.

S started combining lots of images, but imagining much more than what they actually represented. She monopolizes her computer, saying she wants to "show something", "make something", but really never getting to a point where her work conveys any meaning to anyone but herself (see figure on the right).



I tried to focus S and her partner for the session, M, by proposing that they could set out to combine shaped in ToonTalk to make a picture. They suggested a butterfly and I tried showing how to use the balls to make wings. But they had a very hard time trying to control the resizing tools properly[336], and the fact that ToonTalk allows no repositioning after two pictures get bammed also complicates matters, but the butterfly did end up being made (see the figure on the right). I also showed them how to use the "-" and "+" keys to select the color of the underlying rectangle. J also tried to make a composition, by combining a tree picture with flower pictures. G tried to do the same, but got completely confused in the process, because he picked up J's tree and dropped it on another accidentally, then dropped the set on a notebook, and that notebook on another notebook, etc.



For the remainder of the session, S opened a text editor and started to type out randomly, just for the fun of it. M wanted to do the same, with such vehemence, and so they both kept typing in the editor (see the figure on the right for a sample).

A recurrent theme is that they keep trying to find the "end of the notebook", no matter whether I tell them that there is no end to the notebooks or not. Before I know it, they often have reached page 130, or even 240.

[336] At the time, ToonTalk required that children would press the space bar once to start a tool and once more to stop it. For these young children, this made it too hard to control the process.

Session 6 (April 18th, 2001):

Only M, J, and R attended. Given G's problems with notebooks in the previous session, I decided to let them practice a bit the concept of using notebooks to store things. My suggestion and welcomed, and this practice took out most of the session (see the figure on the right, with a notebook created by each child).

Session 7 (May 3rd, 2001):

R did not attend. This session was not successful. I had offered to show them how to "make pictures move". But G, who from the very first session had a habit of bothering his colleagues, kept harassing all the other children, disrupting their activities. They all got quite fed up with the entire situation and lost all desire to use the computers. To top it all, a bullfinch flew in through the window and they got scared, so I had to make it fly out, which wasn't all that easy. I tried to program some robots myself, to see if they would regain interest upon seeing my robots working, but to no avail.

Session 8 (May 10th, 2001):

G and M did not attend.

L wanted to pick up a bomb and expand it, expand it more, and more, and more. So I coached their efforts, to have R hand a bomb to L, who pumped it until it was large, and finally R set it off to blow.

Since the house was gone, I proposed teaching them how to build new houses, which R found interesting. But L was more into making copies and playing around with items.

Suddenly, she dropped a box on a robot and upon entering its thought bubble I asked "look, we are now teaching a robot, but that wasn't what you intended to do, was it?" She replied: "Yes it was. I want to teach it how to write my name." She managed to do this, but as it was often the case after they taught robots, afterwards she couldn't care less whether the robot really did manage to write her name or not.

Afterwards, I could follow up with R and show him how to build a house and go into it to confirm that the robot's "gift" (*i.e.*, the contents of the box used to build a new house) had indeed been taken into the new house (J also took part). Taking my hints, R taught a robot how to put the contents of his box on the room wall, and used this robot to make new house. Then I coached him into going into those new houses, to confirm that the "gifts" given to the robots were really on the walls (see the figure on the right).

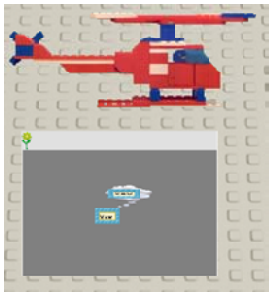
S would have nothing with programming today: she wanted to make a freehand drawing: a butterfly. After she completed it, I gave her a paper comic I had produced with snapshots from the robot she had programmed in session 3, and she really wanted to tell the robot "story" to the other children – who couldn't care less (see the photograph on the right). 4 times she tried to assemble the others in a small entry hall, with my help, to listen to her story. It had little to do with the actual robot actions depicted in the comic. Rather, she seemed to be making it up as she went, and the story made no sense to the other children, who obviously weren't interested in listening.

Session 9 (May 24th, 2001):

G didn't attend. Only one computer was available, the others had been locked up.

I showed them a new thing: the sounds notebook. Although I assured them that they could turn the sounds on while they were holding them, all children insisted on dropping them (take them out of the notebook) and only turn them on afterwards.

S exhibited a now-common pattern: she flipped the notebook until she reached the last sounds, listened to them and proceeded to create many copies of the notebook, to have many things with which to tell a story.



I showed to J and R how to put a sound behind a picture, and J realized it wouldn't play by itself. Then explained that they would have to teach a robot a put it behind the picture, for it to play the sound. J did most of the experimenting, and then instructed R on how to do this, without much trouble: soon there was a helicopter making a lot of noise.

It was finally M and L's turn, and they learned that by flipping the helicopter programmed by J and R, they could make the sound stop. Then they set up to do the same thing (play a sound) with a flower. M was doing most of the things, with L on her lap! But L would take over control from time to time. The figure on the right shows the helicopter (which is making a lot of noise) and the flipped flower, with the sound-playing robot behind.

M and L had taught the robot to play the sound 5 times, but they were surprised that the sound would keep playing over and over. I tried explaining that the robot would still find the sound in the box after playing it 5 times, and would repeat it again, over, and over, but I don't think they managed to understand this. Lastly, they unflipped the helicopter and found it very funny for both sounds to get mingled, so they called all the other children to listen.

Session 10 (May 31st, 2001):

G and S did not attend.

L practiced the manipulation the ToonTalk elements. During the various sessions, she has been the one with the least opportunities for using the system on her own, and needs more time to become comfortable in terms of overall control. She seemed to be attempting to build a house, but not really sure of how to proceed, so I helped her. Mid-way through the process, I realized that she was often pressing the middle mouse button (with no effect) and this was confusing her. She also was eager to reproduce whatever M was doing, in another computer.

M, albeit demonstrating a nice level of overall control, still has some basic problems. Specifically, she sometimes presses the mouse button and the space bar together when operating a tool, resulting in that the tool is dropped instead of turned on; but it works when the purpose is stopping the tool, because either dropping it or pressing the space bar do the trick. Also, she also gets confused often between the operating mode letters of the tools. I assume that lack of specific practice, tool-oriented, led to these confusions.

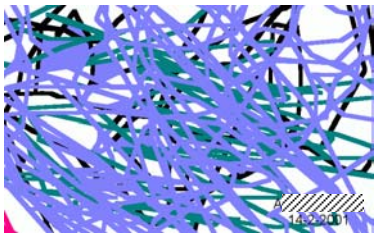
M drew a bird and, together with L, started to program a robot to play the bird sound. Regarding the rule that "robots must receive everything in boxes", she asked: "why not in the toolbox, then?" ("That box is too big," I said.) They did not complete this because of the high room temperature: by now they were too tired.

I did not provide much guidance for J and R during the session. They were involved in composing an arrangement of various pictures, and using a large rectangle as background. J couldn't remember how to find a rectangle ("paper sheet"), but my verbal instructions sufficed: he no longer required for me to demonstrate or point to anything in the environment. J in particular is demonstrating a great ease of orientation and control, as can be gathered by his choice of images and their placement (see the figure on the right). He also managed to store the composition in his notebook, without any help from me.

Group 3

Session 1 (February 14th, 2001):

All children performed, with only slight differences, the same initial sequence: land, select a house, go in, kneel, scatter objects on the floor, and vacuum them. They also did a few freehand drawings (on the right is presented A's drawing, she called it "a spider's web").



B & D were thrilled about traveling around with the helicopter and controlling the figurine. The change in perspective when kneeling and getting up was also enjoyed by them, and they spent a fair amount of time doing these things.

Together, A & M taught a robot how to vacuum bombs, which I generalized afterwards. I suggested that they might now teach a few more robots any way they liked, but when I came back to see the result, they had only scattered birds and numbers, and a few boxes (and boxes inside of boxes). But at that moment A taught a robot how to vacuum the contents of a box that, inside, had another box with a nest.

I've shown R & J how to program a robot to vacuum a bomb. Then I challenged them to teach a robot how to vacuum trucks. R managed to progress and do it eventually, as J watched. But in the end, R couldn't find out how to make the robot work, but J did: she pointed R to the right box.

For the remainder of the session, I let them all explore the environment at will.

Session 2 (February 21st, 2001):

J did not attend.

Initially, all children did freehand drawings in these sessions (on the right is presented R's drawing). Upon starting to use ToonTalk, they all demonstrated the ability to notice the wiggling cue provided by the system to show which object is the "target" of operation.

M still needs to get more used to controlling the mouse: she mistakes directions (up/down with left/right) and has a tendency of pulling suddenly the mouse towards her to drop objects, instead of clicking the buttons.

A was demonstrating a great ease in controlling the system, so all children were gathered round her computer, as I proposed a "funny" activity to her: to program a juggling robot (see figure on the right). Basically, the robot would pick a ball and give it to a bird, but the bird's nest would be located under the ball, so the bird would replace the ball in the original position and this behavior would continue endlessly.

A had no difficulty whatsoever in programming the robot. And she tested it with ease, also. I told her to place the nest in the robot's ball hole, which she did, and there it was: a robot "juggling" with a ball.

After this demonstration, the children were allowed to do whatever pleased them in ToonTalk: R was laying boxes on the floor and filling them up with trucks. D was traveling around with the helicopter or exploring by foot with the figurine, M was enjoying the sense of control by shaking the programmer's hand continuously.

D also learned how to browse the notebooks, and overall they enjoyed vacuuming things, and even spitting them, either by clicking the appropriate letter on the keyboard or clicking the tool button, but the keyboard method was easier for them.

Session 3 (February 28th, 2001):

Only A, J, and B attended. Rosa Cristóvão assisted me in part of this session.

B wanted to play on his own at a computer. A and J could have done likewise, but they chose to share a computer. B only wanted to make freehand drawings, A & J wanted to play in ToonTalk. B was not that familiar with the Windows Paint program, so I gave him a short briefing on the paint tools and color tools and left him alone while we drew, but I had to help him out from time to time.

A & J really wanted to see the Bammer mouse, so they starting dropping numbers on one another, to see it come and bam them together. They were also trying to guess the resulting number. I suggested they might try the same thing with boxes. Initially, they dropped a box inside another; but when I told them that they had to drop the boxes' sides together, they required no more help: J did the first combination and A followed; pretty soon they had created a very long box. Then they wanted to make a second row, but since this isn't possible in ToonTalk, they created another long box.

Then A & J, while I helped B, made four copies of the pump and were trying to put them inside the boxes, to no avail, since ToonTalk doesn't allow this. I explained that boxes only hold "toys" and "drawings", and they went on storing objects and pictures inside them.

B wanted to play with them at the same computer. I didn't think the three of them together would work out, so I set up a ToonTalk session for him and tried spur him into achieving the same things at his own computer. After entering a house, he also started bamming numbers (and letters) together, and then went on using boxes.

B then held a robot and tried to give it the "A" letter. I explained that robots can only take things inside boxes. So he picked up a box, placed the A inside and gave it to the robot. I asked what he wanted to teach the robot: "to take out the A and put a bomb there." "OK," I said, "take the A out, then". As I'd seen previously with other children, he tried using the robot's moving arm, instead of the extended arm. After I explained the problem, he managed to teach the robot as he wanted (the figure on the right shows his robot as it is about to drop a bomb in the box), but in the end was fed up with the situation. However, I easily managed to persuade him to try out the robot, and when he did he was most amused when he confirmed that his robot had been "well taught". Following my indications, he saved the robot in his notebook and showed it to A & J, proudly. A & J clearly understood the situation, to the point of hurrying up B while he demonstrated the robot: "come on, give it the box".

Although they enjoyed seeing this, A & J didn't seem interested in teaching robots themselves. B joined them – he really wanted to be with them at the same computer, but this proved unmanageable: B was disrupting the nice cooperation between A & J.

Rosa Cristóvão arrived at the room, and while B showed her the robot he had done, A & M resumed their play: they wanted to fly in the helicopter and "lose the helicopter". So they flew away from all houses, landed, and then walked far away from the helicopter. Then they wanted to return to the houses "without the helicopter", but they weren't able to, so I showed them they could summon the helicopter by pressing H.

When they were content with flying and walking around, they welcomed the idea of decorating a room wall, and this went on fairly nicely, they even used the pump to resize the images to their liking. They checked on the wall the result of the decoration work done on the sensor, and then also did decoration on the roof. For checking they had to fly out, and the confirmation delighted them.

Meanwhile, B was doing freehand drawings with Rosa's assistance, because he preferred it over ToonTalk play. One of these drawings is presented above, on the right.

Session 4 (March 7th, 2001):

B was the first to arrive. Alone, he landed the helicopter and went into a house, and said he wanted to teach a robot, so he gave it a box. Then he programmed this robot to fill the box with a bomb (see the figure on the right).

However, when the remaining children arrived, B could never again focus on ToonTalk activities. I paired him with M, but it didn't work out: they both wanted to make drawings, but they were unable to help each other.

D & R spent most of the session manipulating the various objects and exploring the environment.

A & J were again bamming numbers together. I suggested that they could teach a robot to write their names, after receiving the initial letter of each name. Following my verbal instructions, while I assisted the remaining children, they each placed their initials inside boxes and joined boxes. Then I told them to show the robot how to write their names, and when they finished I spurred them into testing it. It was a great joy for us all when, while I was helping out R, they started shouting "it worked, it worked!" The figure on the right show their robot has it is about to complete the programming.

After about half an hour without specific support, A & J were getting fed up, and so they left ToonTalk and started making freehand drawings.

Session 5 (March 28th, 2001):

Rosa Cristóvão assisted me during this session.

J & M were moving on without major troubles ("we want to play with Bammer the mouse"), so I spent most of the session assisting the pairing of B & D, while Rosa did the same for R & A.

Initially, B & D were enjoying themselves in exploration and bamming. For instance, they created the string "AAA" and B went carefully to each A, and dropped the number "1" to get "BBB".

B still remembered his "bomb-making" robot from the previous week, and started to demonstrate it to D. While he was doing this, I suggested that after putting the bomb in the empty box he could join another box to the initial one, so the robot could make many bombs. And then this happened:

- he was both trying to join boxes laterally and vertically, no matter how much I told him that from above it wouldn't work; this information that I was providing only made him place them even more carefully together, even though Bammer wouldn't show up;
- after placing the first two bombs inside boxes, he didn't stop: we kept placing more, and more boxes and bombs side-by-side, because he "wanted more"; he couldn't realize that the robot would later repeat his efforts and produce as many bombs as he might want;
- after exiting the robot's thought, which he did only after I insisted, he wasn't keen on trying out the robot by giving it a box ("again?", he said with a clear dislike in his tone).

My interpretation is that B thought that the box was the trigger for the programming process, not the absence of pictures on the robot's thought bubble. This is a clear indication of the importance of actions regarding interpretation, over visual conditions.

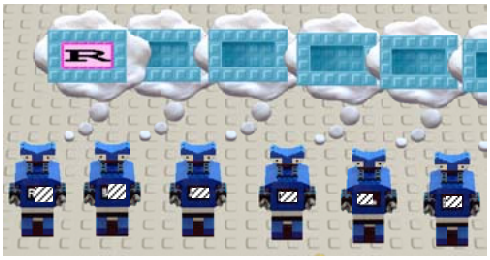
J & M learned how to browse the main notebook and the subnotebooks to find pictures. They also enjoyed discovering that the + and - keys changed the rectangle's color, and used this to make a combination of rectangles of different colors. However, they are at a loss regarding the reason why some images "disappear" when Bammer joins them together (*i.e.*, when two transparent corners are superimposed).

R wanted to write his name, so I suggested he taught a robot how to write his name. Rosa was helping D do the same thing, but since they started with an empty box, the robot would write D's name over and over, which he found very amusing and B immediately wanted to reproduce that. I and Rosa took this cue and all the children programmed robots to write their names (the figure on the right shows D's robot writing his name).

Session 6 (April 4th, 2001):

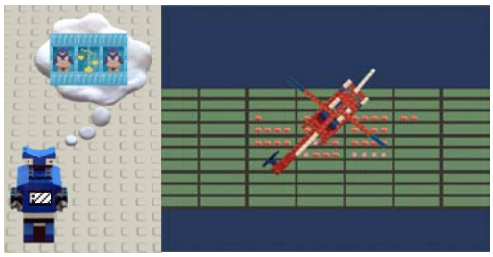
I kept no written account of this session. However, I did save several robots and a screen shot.

The children have again played with their name-writing robots (see them all lined up below).



R programmed a robot that would take two birds with a scale between them, and then would replace the right-side bird with a truck. D started making houses, lots and lots of houses, manually, and then taught a robot how to build houses.

The figures below present R's robot and the result of both D's work and his robot's work.

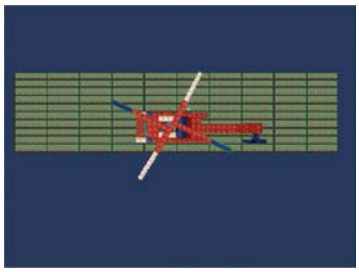


Session 7 (April 18th, 2001):

A did not attend.

They all chose to make houses, including D. But D, who had already done that, last week, was in fact keener of trying out the bomb: he was making tall trucks, narrow trucks...

J built several houses easily. M build two, but didn't enjoy it and decided to dance in the middle of the room (herself, not the figurine) instead. R & B were not getting together very smoothly but houses were being built, nonetheless.



I explained to J, R, and B, that if they taught a robot how to build houses, it would make many houses for them, very fast: they wouldn't have to build them all one by one. They tried this out, and enjoyed this immensely (M watched it, liked the result, but didn't feel like trying it). For instance, when his parents picked him up, R came back to the room within a couple of minutes, to continue making houses (his parents were right behind him to fetch him); B took the utmost care to ask me to please save his houses. The figure on the right depicts his result: a ToonTalk city full of houses (the other children achieved identical results).

Session 8 (May 2nd, 2001):

D and B did not attend.

They wanted to build houses, by programming a robot. Since A had missed last week's session, this was a novelty for her. She started by building houses manually.

R's robot, while building houses, picked up a bomb and tucked it away again, and R was very puzzled about this. It took some effort until he remembered that he had indeed picked up and dropped a bomb while teaching the robot. But neither he nor his computer pair, M, could understand why it would be necessary to teach a different robot, in order for it not to pick up the bomb.

Unlike what had happened in most sessions, M was very interested in using ToonTalk, but due to lack of practice, at a very basic level: playing around with numbers, letters, etc.

J & A, following my indications, programmed a house-building robot. While it worked, A kept watching, as it developed the city. An interesting note is that they did this robot in two attempts. Initially, their robot would use the entry box to make a new house. But this robot would only build one house, because after that one it wouldn't have access to more boxes. So they learned that they had to use a new box each time, in order to develop the entire city (the figure on the right shows their first robot).

Session 9 (May 9th, 2001):

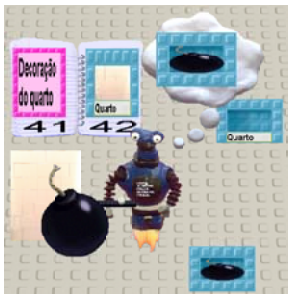
The children offered me a bouquet of clover flowers.

To my surprise, the moment I turned on the computers, all children immediately double-clicked the ToonTalk icon successfully - something that had been troublesome up to last week. B even said "twice, fast".

While I assisted D & M, some difficulties were being experienced by R & B. ToonTalk was always exiting. In fact, B was taking advantage of every opportunity to press Esc until ToonTalk exited; and this was something that bothered R. And this, in turn, made B press Esc more often...

D was making houses. I suggested that he taught a robot how to decorate the wall, to send in the trucks: that way, the new houses would have the walls painted instantly. He got excited about the idea. I coached him verbally while assisting B and watching how A & J were doing (they were copying trucks to build many houses, fast).

D finally got his robot ready, and protested that it hadn't painted the wall. I asked him to check the robot's thoughts, to see what it needed. He gave it the necessary box: a box with a bomb. But he decided to get up to watch it work, and ToonTalk got excruciatingly slow, slower, and finally crashed, therefore I could not show him the result of sending that robot in a truck. The problem was that he taught the robot to put a copy of a bomb on the wall, instead of the original bomb (see the figure on the right). And so, ToonTalk was dealing with more and more pictures in each cycle.



Meanwhile, I had been coordinating R and B, who were combining objects, and scaling them, but they didn't manage to cooperate very well. Finally, they decided they would make freehand drawings in separate computers.

Session 10 (May 16th, 2001):

Only A & D attended.

I announced that I was going to show them how to put robots to work behind pictures. Each child chose a picture from the notebook (A chose a house, D chose a flower).

I asked if they could teach a robot how to turn a bomb on, to blow everything up. They said they could, and it was true: each programmed one such robot.

I showed them how to place their robots on the back of the pictures, using D's flower as an example. But when it was turned, D said "that thing is not the flower". This made me explain that the flower was just a drawing on a paper sheet, and that we were watching the back of that sheet.

When I unflipped the flower, without giving the robot a box, nothing happened, but this didn't surprise them. So I asked D to give that robot a box. After unflipping, nothing happened. After dropping the unflipped flower, nothing happened. I was surprised with this behavior, myself.

Why would that be, I asked them? Why wasn't the robot working?

They said that the bomb that was given wasn't the robot's bomb. The problem wasn't in the robot, they thought: it was something with the bomb. I tried this again after the session ended, and this time the robot worked. It must have been some transitory bug.

They programmed a new robot, which would pick a bomb from the toolbox instead of using the one received in its main box. This time, it worked, but the children were expecting the flower to disappear after it was dropped, not when it was flipped.

After this, D decided to pursue a different path of activity. Unsurprisingly, he started lining up trucks to build houses. So I asked "what if the robot that we send to the new houses is that one which sets off a bomb?" The reply came from A: "The houses would blow up." So was started the programming of "Explodes", the house-blowing robot. I also suggested that in the current house they could place a robot building houses for "Explodes" to blow up, which was similar to what they had done in the previous session. This one was programmed by A, whose only difficulty, and not a big one at that, was finding "Explodes" in the notebook.

However, before A put to work the "makes houses" robot, D programmed his own robot, albeit sending a non-taught robot to the new houses.

Both children were excited, because they could feel that "many things were going to happen." They would say "let's go outside to watch, hurry, hurry." And so, as they went outside, there were lots of houses, but only some of them were blowing up. A said: "those are the ones made by my robot."

From this point onward, D wanted to make new houses to go there and blow them up. He didn't want the use the "Explodes" robot several times, though – he wanted to blow them up himself.

Session 11 (May 23rd, 2001):

B didn't attend.

Right from the start, A told J that she should build houses. And that, first of all, she – J – had to teach a robot how to build houses. While other computers were booting, I spurred J to go inside a new house and look around. There it was, the robot she had sent there. "Why wasn't it doing something?" I asked. I provided the answer: because we hadn't taught it anything. So I proceeded with my inquiry. "What if we taught a robot how to blow up houses, and then sent him to a new house?" And J replied: "It would blow up the house."

And so, J programmed a robot to blow up houses, and A programmed another, to build houses, using J's robot from the notebook (shown on the right figure[337]). However, A's robot was quite unusual: it built two houses in each cycle. The first house was built by sending J's robot and a new box; but the second house was built giving J's robots the entry box for A's. Therefore, it had a bug: it only ever did two houses, no more.

I hadn't watched A program this robot, so I asked, unaware of the problem: "It only makes two houses and then stop, why should that be?" And it was A that immediately answered: "Oh! It is because we used its own box, so now it has none." There was such a tone of "obvious" to this answer, that it amazed me. So J and A started teaching a new robot, that wouldn't use its own box, while I helped R, D, and M. When I returned, J&A's robot did lots of things, because they had run into trouble: they had picked up the wrong notebook, then they had dropped it and moved it aside to find J's robot. While J&A's robot did this, A said: "yeah, it now does all these silly things we have been doing." But anyway, it worked, so they left the house and took off in the helicopter, to see the houses exploding. Upon seeing this, I congratulated them, and A said: "well, it's easy..."

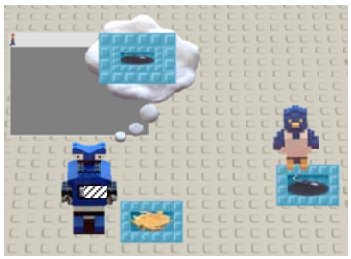
Another interesting note: A said, "How can we get up and leave the toy box there?" I replied that we can't do that, it always trails behind us. So she said, "but won't the robot stop then, if this box goes away?" This is a most insightful observation: if a robot stops for lack of its blue box, why doesn't it stop for lack of the toolbox where it gets the trucks?

As for R, I helped him out because he was confused: pressing too many times Esc, kneeling on the lawn too often, etc. Following my instructions, he did a house, but said, "no, I want to build houses faster". So I told him that if he taught robot, that robot would make houses very fast. He managed to program it easily. But he did ask: what should he put in the truck first, the box or the robot? I told him it was all the same, but that he might start by the robot that was going to live in the new house, and then place the box, as a gift for the robot to take into the new house. He was satisfied with this replied, and then started blowing up the houses his robot was making.

Meanwhile, M was together with D, making simpler things: identifying the letters in her name on the keyboard, for instance. She is much less at ease with computers than the other children, she needed a special attention. She and D were trying to make houses, and had placed everything needed on the ground, but it wouldn't work because they were working on the lawn - they didn't remember that it would only work if done inside a house. D knew there was a shortcut key for the wand, because he asked me which one it was. And I could see some objects whose surface had been erased, so they had been using the vacuum cleaner also.

D couldn't remember how to make houses that exploded, but when talking about it, he demonstrated that he had grasped the concepts, the general process. Therefore, in order to do something different, I asked him: what would happen to a drawing if we placed a bomb-setting robot behind it? "It would explode," he said. But then I added an extra complexity. "But we wouldn't want it to blow right away," I said. I continued: "What if we gave the robot a box, and then used a bird to take a bomb into that nest?" He replied with a question: "would the drawing explode, by then?"

"Yes," I said. "Do you want to do this?" He nodded, and was excited with the prospect.



I helped him out along the way. The first task would be preparing the box for the robot. He picked up a box. I asked him to get a bird and to put the nest in the box. "Now we need to give the bird a bomb. Since it doesn't take bombs, we need to put it inside a box, beforehand." He did this. Then came the tricky bit. "Let's give this box to the robot, then," I said. He was puzzled: "This one?!?" He clearly couldn't see the connection. But he did so, and then, while teaching the robot, set the bomb off.

The following bits were simple. He chose a picture (a boy walking) and placed the robot on the back. I told him not to give the bomb immediately, so it didn't blow up at once. But he hadn't grasped the nest connection, so I had to tell him what to do next.

However, once he unflipped the drawing and I spurred him into giving the bird a box with a bomb, he saw the bird fly into the back of the picture, the picture vanish, and bird returning. And he said, at once: "the robot blew up the picture," he said. Clearly, he had understood the global process, the global idea, but not each specific internal step. The figure on the right shows his robot, with its box by its side, the flipped picture, and the bird near the bomb-in-a-box.

[337] The notebook name doesn't match the programmers' because they had used the last user's login.

Group 4

Session 1 (February 15th, 2001):

The usual initiation activities were used: landing the helicopter, entering a house, taking objects from the toolbox, vacuuming them.

Right from the start, two children wanted to make freehand drawings. But they agreed about playing in ToonTalk first, and drawing freehand later (one of the resulting drawings is shown on the right). But given this situation, I suggested that all children could now decorate the ToonTalk house they were in. Since this was the first session, I placed the room sensor on the floor, as well has the images notebook. Thus they could place elements on the sensor and then get up to check the overall result on the actual wall.

Afterwards, while A, DU, H, and J did some freehand drawings, AA and DA started playing with ToonTalk's helicopter, flying around and exploring on foot with the figurine. When they entered a house, I suggested to AA that she could teach a robot how to paint the wall. I told her to give the robot a picture of her own (two overlaid house images), and then, inside the robot's thought bubble I fetched the room sensor for her. She placed the image on it, and then we left the thought bubble. She and DA then tested this robot with different images.

Session 2 (March 1st, 2001):

Due to other, unexpected pressing professional matters, this session lasted only 15 minutes. These were used to let the children practice the ToonTalk environment overall, manipulating the objects and tools as much as possible.

Session 3 (March 8th, 2001):

Fernando Lemos assisted me during this session.

I suggested, as a general theme for the session, for all children to make things bigger or smaller, thus practicing the use of the pump. I also intended to suggest tucking the resulting objects in a box.

Led by Fernando Lemos, H and DA played proficiently, creating differently-sized bombs. Afterwards, they used boxes, and decided that it was funny to be able to have a very large box, and set out to build one with 13-holes, with many different objects (see the figure on the right).

DU and AA were trying to monopolize their computers; this was somewhat of a complication. They managed to use the pump effectively, although still needing a bit more practice. J also demonstrated having enough control, but lacking willingness, which is understandable.

A was repeatedly taking out object from the toolbox and putting them back in, and seemingly enjoying himself doing it.

As the session neared its end, AA wanted to draw a garden, but there wasn't enough time to do it in Windows Paint, so I suggested that she planted a garden of flowers in ToonTalk and used this suggestion to allow her to practice the change in color and size of a rectangle, and change the color of flowers. The resulting garden is shown here on the right.

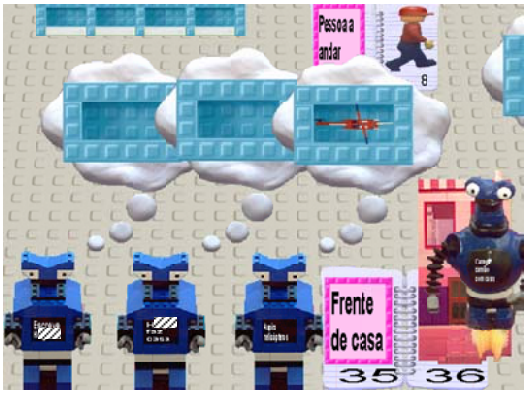
Session 4 (March 29th, 2001):

There are programs and screen shots of this session, but no descriptive document. The children must have chatted in the preschool with the children from group 3, who by this time had already benefited from 5 full sessions, because they are exhibiting a very advanced control of ToonTalk elements and even programming concepts.

A & DU used the magic wand to create many trucks and boxes. So, following my suggestion and instructions, DU made a robot that would place a truck inside an empty box and A made another that would vacuum a truck from a box with one. I then suggested that they could join these two robots in a team of their own (see the figure on the right, above). However, this team wouldn't work because of the way ToonTalk interprets constraints: the empty-box constraint means "anything or nothing", rather than "empty box". Therefore, when the box already has a truck inside, the team leader tries to put another truck in it, rather than hand it over to the next robot in line[338].

[338] Even if the team order was inverted, it still wouldn't work, because ToonTalk would make the robot holding the empty box wait until something is put in it, rather than pass it over to the next robot in the line.

H managed to program 4 robots in all during this session: one to write his name, another to build houses, another that would vacuum helicopters from its box, and yet another (shown working in the figure below) that, if given a box with a truck, loads the truck with pictures of houses.



On their part, J and AA each programmed a robot to write their name. Since their robots had no starting constraints (*i.e.*, they were taught starting with an empty box), the names would be repeated over and over again, quickly extending along the room floor, something they found quite amusing (as shown in the figure, below).

Session 5 (May 3rd, 2001):

H and DA did not attend. Compared with the last session, this one was frustrating, probably due to a one-month gap between sessions.

I showed them their robots from the last session. J keeps insisting on wanting to make drawings, rather than play in ToonTalk. This time, I only managed to have her enjoy the selection of paper color and objects to compose a picture, no more than that. Her drawing is shown here, on the right.

A and DU, each alone at a computer, seemed to have regressed. They found it hard to land near the houses, and only managed to manipulate objects, doing nothing more complex than that. I tried showing A how to flip a ball and change its position control to impact the ball's position, but this didn't catch his attention.

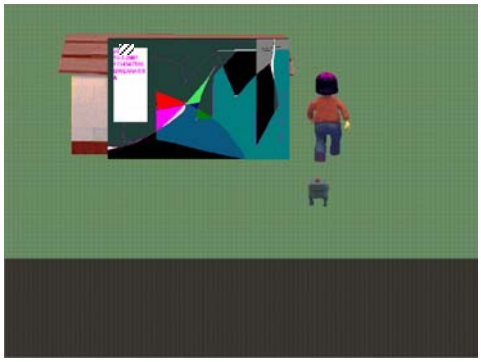
Session 6 (May 10th, 2001):

Only J and DU attended.

Today I found out that DU, although he regularly manipulated ToonTalk, hadn't grasped basic manipulation concepts which had been used in the previous sessions:

- he hadn't realized the importance of the wiggling cue;
- he hadn't realized that the space bar turned the tools on & off;
- he didn't now how to flip the notebook pages.

Following his own goal, he set out to compose a garden. I helped him achieve this, but mainly tried to let him practice the control of ToonTalk as much as possible. Then I instructed him on how to put his composition on the wall (see the figure on the right).



J was constantly requesting support while I was with DU. She still wants to make drawings, but now has another area of interest: writing. She can't really write, but pretending to write is enough to satisfy her. So I showed her how to write text in ToonTalk, and have a large white rectangle as paper sheet for the text, but she quickly grew tired of this. However, upon seeing DU's garden, she decided that it would be a nice thing to do. Since she still wanted to make a drawing, I decided to let her draw in Windows Paint, and then use her drawing to decorate a ToonTalk house, to try and spur some extra interest on her about the programming environment. The resulting house is presented in the picture on the right. Note: after making the drawing, she said that "without help, these are the only kinds of drawings I can make." But she didn't really care about decorating the house, I'm pretty sure she did it just to please me. What she did find interesting was when she saw the ToonTalk sensor "Left Mouse Button clicked", something she experimented with for a while. As for the decoration, she only found it amusing when she went outside to check the looks of the house and finally realized the consequence of her decoration. That did amuse her!

Today, I discussed these events with her mother, who reported that J was "*serious about everything; she complains with me about the preschool teacher, because she makes them play and 'school isn't for playing, it is for working'; she conveys any messages when the auxiliary staff forgets about them; at home, she prefers to 'work' in the neighbor's farm, helping out with the potatoes, picking eggs... serious stuff, instead of regular child play, which she abandons; when role-playing with other children, she's never the daughter or baby: she always plays the role of mother*".

Session 7 (May 24th, 2001):

Only A, AA, and J took part.

This session failed. I thought that presenting sounds would spur their interest, but this didn't work out.

J only cares about drawing and writing. She doesn't care about ToonTalk the least.

AA enjoyment varies a lot. Today, she didn't want to do anything, yet in the previous sessions she mostly enjoyed herself. I asked why she was saying she didn't like this: "because it's hard", she said.

When I showed them the sounds, AA and J tried them a bit, but J wasn't the least interested and AA followed her stead.

Since A was using his computer alone, today, I realized how much more support he needed. He was at a stage where simply dragging the mouse and seeing the figure move excited him. He doesn't realize that he can grab things when they wiggle, keeps clicking outside the desired object, and inches closer and closer, clicking, but without ever, ever putting the figurine's hand on top of the object. When I help him, dragging the mouse so that the screen hand is over the object, he says "this is it! this is it!" and gets terribly excited about being able to grab something.

AA and J were combining pictures, while I was devoting my attention to A. We were absolutely interested in everything, but his background is completely different from that of the other children. When I mentioned teaching the robot he held the mouse in his and asked "and what is the robot? is this it?"

J had chosen as her "picture" not a colorful one, but rather a plain geometrical shape (a circle). And she wanted a plain white sheet to place the circle. I told her about changing the color of the black rectangle, but she didn't want to have nothing to do with that. It was hard to convince her to try and change the color. Once she got convinced, she changed the color of the black rectangle effortlessly and then placed the circle on top – with absolute precision! And then, also easily, she enlarged the resulting picture to almost fill up the screen, but not quite – exactly the size she wanted. And that was all she wanted to do.

As for AA, only at the very end, after J's actions, did she expressed her interest in creating a garden and make it almost as large as the screen. She started making it and I could verify that she had the skill to use the pump properly, but there was not enough time to complete this.

Session 8 (May 31st, 2001):

H and DA did not attend.

During this week, the preschool teacher was able to explain to J that these sessions were not plain child play, but serious stuff. This was enough to change her perspective entirely. Now she was devoting herself to any tasks involving ToonTalk.

J and AA started composing a picture. J wanted to do it based on diagonal lines, and knew pretty well what she wanted to achieve, but her lack of effort on the previous sessions meant that she hadn't enough tool-manipulation skills to enlarge and shrink the diagonal lines with the precision she intended. Turning tools on and off was trivial for her, but

she often failed to point them at the proper object, for lack of focus. I showed them birds, but they didn't care about their operation – they just wanted to create as many as possible.

At the end of the session, I realized that there were a huge number of gigantic robots on the floor (see the picture on the right). All of them had nonsensical text for names. This was most likely J, who must have realized that she could type away at will after enlarging the robots enough. The picture below, on the right, shows a sample of this.

A is still unable to control ToonTalk well enough to achieve his intentions. As for DU, he's a bit better, but still clicks the mouse buttons without being necessary, and has difficulties getting hold of the objects. Even with sounds, A and DU enjoyed trying them out but lacked the necessary hand coordination to browse the notebook, pick sounds, play them, etc.

I decided to try a last situation: I programmed a bird picture to play the sound "Yaw" whenever it collided with something. And asked them to try and test it. To do so, A chose a house picture and they dragged my bird towards it, to check out the sound. But they kept clicking the mouse needlessly, meaning that often the bird picture would be dropped on the house picture and bammed in it.

Finally, I show AA and J that the sound was produced by a robot on the back of the picture. They didn't really understand it, or at least were unable to explain it. But, for the very first time, J got really, truly, involved. Not to the point of trying to reproduce it, but she was really looking at the situations, considering it – pondering it. AA was interested, but seemed more puzzled than anything else.

Group 5

Session 1 (March 29th, 2001):

This session was devoted to providing J with enough latitude for exploring the environment: landing the helicopter, walking with the figurine, entering a house, grabbing objects and dropping them, and the combination of letters and numbers with Bammer the mouse.

Session 2 (April 5th, 2001):

J still remembered how to use the helicopter, and make it rise and descend. During this session, I noticed that she has developed a technique for using a regular-sized mouse: she holds it by the bottom (the end nearer to her), until the cursor is positioned. Then, she releases the mouse, lifts her hand and goes up towards the buttons, to click them. So, throughout the session I would hold the mouse while she did this, to make sure her clicks would occur exactly where she positioned the mouse, and not elsewhere.

Following my indications, J grabbed the main notebook and took out the images notebook. I showed her how to use the notebook, and she browsed until she found an image she liked (the roof of the pink house), laid the notebook on the floor, took out the image and laid it on the floor also. From then on, she started combining other pictures with this one, in any way that she desired. While doing this, she also practiced the use of the resizing pump. An interesting point is that while looking for it, she called it "hose", and indeed I usually recommend that children use the pump by "pointing the hose to the picture."

When her composition was complete, she got the picture frame from the notebook, resized the frame and her composition suitably, and set it all together. It was "her painting." It should be noted that one of her father's hobbies is painting. So, I took out the room sensor from the sensors notebooks and laid it on the floor, for her to hang her painting on the wall. After resizing the framed picture adequately, she got the figure up to check out how it looked on the wall (see the figure on the right).

Another interesting aspect is that before the end of the session she still exited the house with the figurine and strolled around the ToonTalk city a bit. While doing this, she referred to the house where she had been as "my house", and said about the other houses: "those are not my house", which shows that she was perfectly capable of knowing where she was, in the ToonTalk world.

Session 3 (May 3rd, 2001):

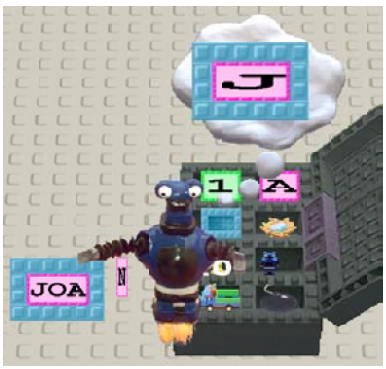
A month had elapsed since the previous session, so I let her recall a bit how to move around in ToonTalk. I showed her, in the notebook, the framed composition she had done before. And told her that, since this was a new city, with new houses, the picture wasn't on the wall, and I suggested she put it there.

However, she didn't want to put the entire framed composition, just the composition itself. This was a nice moment to introduce the use of the vacuum cleaner, both for vacuuming and for spitting out objects. Regarding the use of the room sensor, she searched it herself, following my suggestion, in the sensors notebook, until she found the "wall."

She took the boxed sensor out, and then the sensor itself from its box, and laid it on the floor, and finally she set her composition on top of the wall. Then, she asked: "how can we put the wall in place?" I explained her that what she had there wasn't the real wall, but rather a make-believe, magic wall, that would take care of sending stuff to the actual wall. I also pointed to the sensor stack in the sensor notebook, to show her that her composition was also visible there. And so, she got the figurine up, to check out the result on the wall.

I then suggested we might play with the robots. She agreed, but made notice that afterwards she also wanted to play with all the other toys.

As for the robot, I suggested she gave the robot her first name initial, and then teach the robot how to write her name. She liked this idea, and proceeded to execute it. The name was written by having the robot pick the letters, one by one, from the toolbox, editing them, and joining them all in the robot's box (as shown in the right-side picture).



The session ended soon afterwards, because she then wanted to play with the bomb. So I led her towards saving the robot in the notebook, and soon after she set a bomb off, and the session was over, but I promised her that I'd teach her how to make houses in the following week, to make up for the bomb.

Session 4 (May 10th, 2001):

There were only 20 minutes available for this session. She had perfect recollection of my promise to teach her how to make houses. So, that was the theme for this session.

I introduced the method to her in the following manner:

- The trucks have inside construction workers, to build houses ("Can you see them inside, there, by the steering wheel?")
- But the workers are waiting for a robot, who is going to live in the new house (as I said this, she placed a robot in truck).
- It's only that the robot wants to receive a gift, to take it with "him" into the new house (and as always, it only accepts well-tidied gifts, inside boxes).

After she completed the process, we went into the new house, to check that the robot was there, as well as the gift. I pointed out that it hadn't done anything with the gift, and the reason was: because the robot hadn't been taught how to do anything.

I suggested she might teach it how to put its gifts on the wall, and she liked this. Therefore, she handed the gift to the robot, and following my indications went to the sensors notebook, and sought the room sensor (it was buried deep inside, and this annoyed her a bit). She completed the process of laying the robot's gift on the room sensor, and I told her to leave the robot's thoughts.

I then asked her if she thought the robot had put the gift on the wall. "No," was the sharp answer. I was startled, but nevertheless asked her to get the figurine up and check – and obviously, there was nothing on the wall.

I then proposed her sending the robot to a new house, with the gift, so that it could put the gift on the wall as it got there. And she managed to do all of this, with little or no advice at all.

Using the helicopter to see all the houses, she correctly identified the original houses, by moving the helicopter over each one at a time as she spoke, and she also identified the first house she had built. She was a bit unsure about the house she had just built, because it was in a different block. But once I reassured her that it was indeed the house she had built, this was no longer a doubt for her. So she landed and entered it without any problems, saw the gift on the wall and loved it!

I then let her play a bit in ToonTalk with the help of her mother, without any goals, while I readied the computers for session with the older children, which was about to start. She basically did compositions of pictures.

Session 5 (May 24th, 2001):

Two weeks had elapsed, and neither J not her mother remembered how to make the figurine kneel. They started playing before I arrived, and they were trying to make the figurine kneel using the keyboard, and pressing Esc, continuously, until the program exited... until eventually they realized it was simply a matter of clicking a mouse button. Afterwards, J started making another picture composition: a rectangle with a ball inside. After I arrived, I mentioned that she might want to explore ToonTalk's sounds, which she did, gladly. I must say I am amazed at her coordination, because she always checks carefully which object is wiggling, before trying to click the mouse to pick it up, and is also very careful not to drop any pictures while anything is wiggling below, unless she means to combine them.

She was already perfectly able to turn the sounds on, so I suggested she placed a sound behind a picture, after showing her how to flip the picture. Unflipping it, no sound was heard, and I explained her that it was just as it should be, since she wasn't holding the sound with the hand to turn it on. So I inquired her, on how could we look at the picture and still turn the sound on. "Could we teach someone how to turn the sound on for us?" I asked. She got the idea, but wasn't sure of how to achieve it. So I asked, "Who can we teach, around here?" "The robot?" she replied, a bit uncertain. That was it, the way to start programming a robot.

Without any problem whatsoever, she flipped the picture, to give to the robot the sound that was behind it. It wouldn't accept it, and I had to remind her that the robots only take things in boxes. With this cue, she placed the sound in a box, gave it to the robot, and inside the thought bubble took out the sound and played it. I told her to teach the robot to put the sound back in the box and exit the teaching by pressing Esc.

We tried out the robot, and she was very amused watching it play the sound. Grabbing the robot to stop it was a bit harder, because since she released the mouse to click the buttons, in the meantime the robot moved away from the cursor. But we did it together: as she moved the mouse, I clicked one of the buttons.

From there, all was simple: she placed the robot behind the picture, gave it the box, unflipped the picture, and there it was, a sound-playing image! The figure on the right, above, shows all the pieces. Then, to check her understanding, I asked: "and now, how can you stop the robot?" She had not doubts: at once, she flipped the image, to pick up the robot. Even her mother was surprised, because the solution hadn't occurred to her.

Session 6 (May 31st, 2001):

Today, we started by using the birds. J tried giving the bird first a number, and then a letter, and correctly realized that it was carrying them to its nest. We then tried hiding the nest in another house, and confirmed that nevertheless, the bird could always find its nest. She tried this out by giving her own name for the bird to carry.

I then suggested hiding the nest behind a drawing. While she did it, I realized not just her easiness in manipulating objects, but also her simplicity in linear decision-making: to catch a robot, for instance, she takes a notebook out of the way, puts it aside, and gets the robot. Another example: to get a nest, she takes out its items, and gets it; then, she takes it to another house, returns to the original house, gives something to the bird, goes once more until the house where she put the nest, checks to see if the bird delivered it – and all with minimal hints from me, without doubts.

She demands to make things herself ("Mom, I know how to do it!"), and is perfectly capable of deciding when she wants to turn a tool on or off. To make her drawing, she used a rectangle with a paddle on top. Then she dropped a picture of the Martian on top, but since it was too big, the head was off sight. No problem. She took out the Martian, enlarged the rectangle, and placed the Martian again.

After completing her "drawing", she no longer remembered about the original goal of hiding the nest behind it. But when I reminded her, she no longer wanted to do that: she wanted to hide it inside other houses.

So she went into another house, but I thought of suggesting her a different experience: copying the nest. She had never used the wand, so I had to teach her now, but she learned easily. After making two copies, she placed them neatly, one above, another below the original nest.

So I said, "Now you probably got the bird all mixed up! How's it going to know which nest is the right one?" She found this interesting and fetched the scales to test it, seeing then in amusement how it divided itself in three to deliver copies of the scales to all the nests.

Finally, she wanted to try it out with a different bird, so she took one from the toolbox. I spurred her into testing the new bird, to make sure she realized which bird was the original one, which was the new one. Then I asked, pointing at the original bird, "will this one take things to all nests, or only to its own?" She focused, pondered this a while, and said "I think it will take things only to those nests" (she pointed at the three nests that belonged to the bird).

A final comment from J: she didn't want Tooly the Toolbox to open whenever she kneeled. Firstly, because tools should be kept tucked up nicely. Secondly, because by popping out they sometimes cover things she wanted to use.

Session 7 (June 7th, 2001):

I reminded her of what was done in the two previous sessions: teach a robot how to play sounds and hide nests from birds.

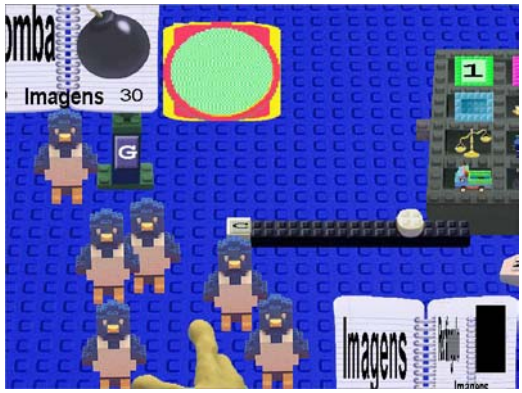
J started by exploring the sounds notebook. Today, I told her that she could return to the first page of the notebooks by pressing 1, instead of pressing "-" all the way back, and she immediately started to use this new technique.

After taking out several sounds, she wanted to make them all play at the same time. I suggested teaching a robot how to play one, and then using several robots at the same time to play all sounds.

In programming, she seemed a bit confused about the starting point. Twice she tried dropping the robot on the sound, not the other way around. But I reminded her about how robots liked things inside boxes, and this was enough to sort out the problem. She programmed in the same way: taking the sound out of the box, and only then playing it. She also didn't remember how to exit the robot's thoughts, and further ahead forgot about the Esc key, again. But she did remember that she had to leave the robot's thoughts to complete the programming.

Upon exiting the programming thought bubble, she seemed to be upset because the robot wasn't immediately doing what she had taught him, and started spinning around in her chair. She lost interest, and only regained it when I reminded her that we needed several robots in order to play several sounds at the same time. But now she wasn't focused. She would give the robot its box, but in a detached manner. Her attention only came back when her robot started playing the sound.

Then she brought the robot down towards the other sounds on the floor. She tried giving it "a very small sound", but it wouldn't work. I pointed out that the robot only wanted to receive the sound we had taught it. My original plan was for her to program a few robots, for playing different sounds, and then introduce generalization in a "useful" manner, a strategy which had provided nice results in the previous year. But in the present context, I thought it would be too bothersome for her to program more sound-playing robot. So, I tried explaining her about the difference between "vacuuming" and "erasing" in the real world, by using the mouse pad: vacuuming would be to swallow it; erase would be to clean it. From this I told her to try and erase the sound on the robot's thought bubble, and she did it and handed it another sound, but her interest on the matter was definitively lost: she said, "I want to go into another house."



In the new house, she said, "can I do whatever I please?" I said she could, so she placed a nest and a truck on the floor. Then she said "that other sound was from this truck, right?" Then she tried to put the truck in the nest, which can't be done. So she said, "But if I gave the truck to a bird, it will take the truck to the nest, right?" I replied: "Yes, but don't forget that birds need to get the truck inside a box." Then she started to copy many birds, as shown in the figure on the right. I challenged her to copy many at each time, by copying a box with many holes, and she said she didn't know how to make one. I offered to teach her, but no, she wanted to copy the birds directly. Then she wanted to make nests for all the birds, and I reminded her that those birds were sharing the same nest. But she didn't seem all convinced, so I told her to try it out with something. She selected some pictures and started to give one to each bird, joyfully. In the end, she was quite concerned whether she had given pictures to all the birds, or missed any. I pointed out one she had missed, and of course she gave it a picture as well.

Throughout the session, J continued to try out lots of things, such as tidying up the robot into the nest, putting things into boxes, and even, under my recommendation, teaching a robot to vacuum the contents of the box. Then she also wanted to explore other houses, and made a curious reference, in the helicopter, while pointing to a house: "Is this the one that has other houses?" I have interpreted this as meaning she remembered having programmed there a robot to make houses, in a previous session.

Annex V
—
Details of sessions conducted in Nov/2001-Feb/2002

The children

During these sessions, the children were divided into 3 different groups. In each session, only one group was present.

GROUP 1
The girl J, aged four, who was the 3-year old in group 5 of the previous year (*vd.* Annex V). This group had a single session, in setting 2 (see below). Afterwards, J started having session in her preschool room, along with a colleague, and so became part of Group 2, below.
J (girl, 4 years)

GROUP 2
A pair of four-year olds. The sessions took place in their preschool room (setting 1, below). The other children in that preschool would be conducting their own activities at the same time, in that room. One of these, J, was the 3-year old in group 5 of the previous year (*vd.* Annex V). The other child was selected by the preschool teacher at the SPP preschool.
J (girl, 4 years); C (boy, 4 years)

The environment

All but the first of the sessions took place in setting 1, described below.

Setting 1:

Within the preschool room, at the SPP preschool, a Windows 98 computer was available, nearby a window. There was no physical separation between the computer space and the other activity areas of that room. The computer had a 15" CRT monitor and regular-sized mouse and keyboard. It was usually turned off, but children could request permission from the educator to turn it on and use it at any moment.

Setting 2:

Identical to setting 3 in Annex V.

Group 1

Session 1 (November 7th, 2001):

Duration: 30 minutes.

J didn't remember some things very well, since our last session in June. According to her mother, she used ToonTalk at home since then, but only 3 times. She didn't remember how to make the helicopter go up and down, but had no problem once I reminded her of the mouse buttons. She didn't even get confused with the top-down perspective. She easily strolled into a house, but pressed Esc to kneel, so I had to remind her that the mouse buttons did that.

I decided that it was better to ask her what she remembered about ToonTalk. She replied that she didn't remember how the trucks were used to build houses. So I explained the process to her again, this time using the version of the robot's box being its luggage. She then made a new house, and went out. Seeing the other houses, she asked if any of them was the new house. I said, "no, these are the original houses. How many houses were there, initially?" She answered: "three." So I said we should go looking for the new one. This we did, walking to the right, and confirming that the robot was inside the new house.

I asked if she wanted to decorate the wall, like last year. She wanted to, but couldn't remember how. So we went over that process again, and she placed a bird picture and a ball on the wall. She got up to check the result, satisfied.

Then she asked about the purpose of the wand. “To copy things,” I said. I spurred her into trying, and soon we had three vacuum cleaners. Then she decided to use the pump, and asked “what letter is that one?” “G,” I said (grande=big). She asked: “Where is the P?” (pequeno=small). She held the pump and tried to identify the P on the keyboard, but couldn't, so I told her which key it was. She made a small vacuum cleaner. Then she set the pump on enlarging, and created a gigantic vacuum cleaner, and got scared, thinking it would grow large enough to swallow her. She dropped the pump and seeing the growth stop placed a hand on her chest and sighed in relief.

She then asked about the use of the number pads. “We can put them on things, and also use them for counting,” I said. Following my lead, she tried dropping a 1 on top of another, and enjoyed the appearance of Bammer. Then she placed the resulting 2 on the wall, on top of the ball.

Then she decided to try out the letter pads. She dropped an A on top of another. Then she wanted to know where to find the sounds! She was remembering everything, but piece by piece. And so she played a sound.

Finally, she vacuumed everything, to leave the house well tidied up, and the session ended.

Group 2

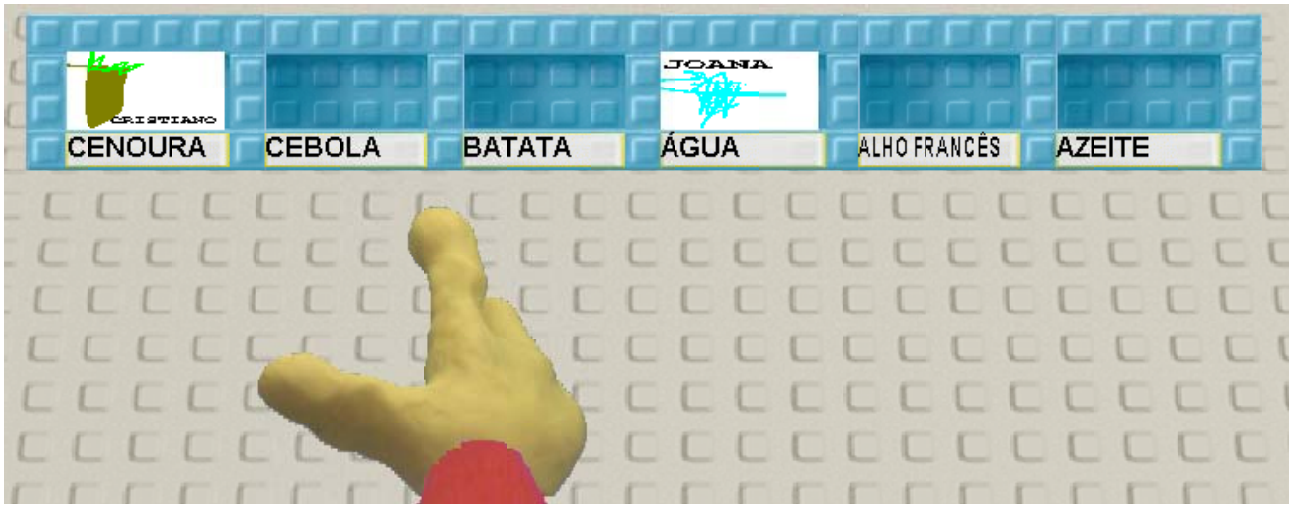
Session 1 (November 13th, 2001):

No written record was kept for this session, but my recollection and the stored files allow me to present the main points.

J & C built a new house. J already knew how to do it, and C, although he had never used ToonTalk before, could describe the procedure in detail. According to the preschool teacher, the children from this preschool who took part in the session, the previous year, had described their activities several times, and this was how C learned about the process. So C was the one using the mouse, to practice.

Then, I wanted to develop a long-term activity, and discussed this with them, in terms of what should we do in ToonTalk during the year. They said they’d like to make soup, likely because that was a topic being addressed in the preschool, that week.

So we started by making a list of all the soup ingredients, and created a box with a hole for each. Then both J and C drew one of those in Windows Paint, and I imported their drawings into ToonTalk, for storage in the box of ingredients. (See the figure, below: from the left, it says Carrot, Onion, Potato, Water, Leek, and Olive Oil).



Session 2 (November 20th, 2001):

No written record was kept for this session, but my recollection and the stored files allow me to present the main points.

Proceeding with the "soup" theme, J and C have drawn two more ingredients, this time by hand: leek and potato (left and right figures below, respectively).

In ToonTalk, we went over the procedure for making soup: for starters, the carrot had to be placed on the water, so they first did that, to see the result, and then a robot was taught to do the same thing. I must say I need to consider how to employ robots for a large number of pictures, since in the common situation any change in order will render the robots useless. A different strategy must be devised.

Overall, it was interesting to see that the children could be kept focused on a specific project, with ToonTalk sessions separated by one week.

Session 3 (November 27th, 2001):

No written record was kept for this session.

Annex VI
—
Documents used as guidance for the group of preschool computer-activity teachers (translated into English)

ToonTalk & the ICEI project – Guidelines

Overview

There is the intention to conduct computer-programming activities with children aged from 3 to 5, based on the language ToonTalk.

These activities aim to collect information on possible results from distinct approaches, within an environment that children are familiar with (preschool), under supported and guided use (by the ICEI professional), in a regular and continued way (one weekly session per child, from September 2002 to June 2003).

Basic structure

The following are the basic assumptions of the entire action.

- The activites are to be performed involving all children in each preschool room.
- Each activity is performed with pairings of children of the same age (two 3-year-olds, two 4-year-olds, or two 5-year-olds); exceptionally, there may be groups of 3 children or a child may be alone.
- In each session (understood as an activity undertook by a pair of children) will yield a detailed report of its development, containing:
  - predefined objectives;
  - session start time;
  - session end time;
  - session development and outcomes, regarding learning;
  - created programs, created objects, and other elements that were actually produced (including illustrative screenshots);
  - general comments.
- The activities are programmed and developed under a global strategic orientation, regarding the approach to programming concepts, as is described further ahead.
- Computer programming is the main learning theme; other themes should be coordinated around this, complementing it.

Approach strategies

Three different strategies were selected, to approach computer-programming as a learning theme:

1. **Programming as the basic activity**
2. **Programming as an extension based on automated behaviors**
3. **Programming as a development of generic activities**

Description of the strategies

1. Programming as the basic activity

It is assumed that before starting programming activities, there is a minimal acquaintance to the ToonTalk environment. The programming of robots starts during the second session, or during the third session at the latest.

The activities should always be based on the programming ("teaching") of a robot or of a robot team, rather than on the complexity of the environment that is being created. The theme under exploration and development is robot behavior, the attention devoted to it (or not) by the children, and specific programming skills: setting of goals before programming, comparing results with the initial goals, feel the need to test the robots that one is programming, etc.

The development of general ToonTalk skills is performed alongside programming, on a "as needed" basis.

2. Programming as an extension based on automated behaviors

After a minimal acquaintance with the ToonTalk environment, one should initiate experimentation-based activities, focused on objects with self-behaviors: birds that carry rectangular objects to their nests; trucks that build a house when they are loaded with a robot and a box; and also objects with programmed behaviors, which can be combined in order to produce complex behaviors.

The programming activities are based on building up from the automated behaviors, in order to simplify them, make them faster, or improve them. For instance, after building houses, one can teach a robot how to do it. After realizing that by placing a name on the roof sensor the name shows up on the roof, a robot can be taught to do that, so that when it is used in the creation of a new house, the roof of the new house displays the name; etc.

This strategy assumes that overall there will be more building up of the environment than on the first strategy. But still, this shouldn't replace programming as the main goal. The development of general ToonTalk skills is performed alongside the programming, on a "as needed" basis.

3. Programming as a development of generic activities

This strategy assumes the continuous development of complex environments, rich in context, before any programming is done.

Before performing programming, each child should learn how to master the tools of ToonTalk, the combination of images, boxes, letters and numbers, and how to get around in the city. These activities are to be conducted without resorting to any programming (*e.g.*, creating a flower garden, setting a puppet theatre, filling up weather-recording tables, etc.). The overall expectation is around "common" computer activities for preschool children. Programming would be an extension: after creating a garden, why not teach a robot how to plant more flowers, for instance.

Observations on modifying a strategy

Since there is the intention of analysing results from the adoption of each strategy, comparing it with the others (and also by itself), it's very important that a strategy is kept throughout the school year.

However, should it be become obvious, *in situ*, that a strategy is inadequate, it can be changed, in view of the specific needs put in evidence by each child. However, it is essential to avoid that such changes turn the preset strategy in any of the other two being assessed. Even if a strategy appears not be producing significant advancements, one should remember that:

- the aim is not only to assess how regularly and easily learning progresses come by, but also to assess the global results during a somewhat long time lapse (a school year).

In fact, nothing tells us that a strategy that appears to be better, at given stage, will not face significant obstacles later, which some other strategy – previsouly seeming less adequate – can then overcome with greater ease.

Sample activities for each strategy

1. Programming as the basic activity

The "Swappy" robot

This is a robot to which one gives a box with two holes, each holding an image. Then the robot is taught how to exchange them. This can lead to activities with different images, perceiving the image between the robot's box and its thought bubble, and the generalization (abstraction) of the swapping activity.

The robot that plays with the bird

A robot is taught with a two-hole box, one holding a bird, the other holding a picture (a ball, for instance). The robot learns how to give the picture to the bird, who then takes it to its nest. This can lead to placing the nest in the second hole of the box, under the picture, so that the robot and the bird keep on playing.

The robots that play ball

This is an application of the second robot, above: by copying it, and fetching another bird, one can put in a box the nest from the bird that is in the other box, so that the birds pass the ball along from one robot to the other.

Our name written many times

This is about teaching a robot to write our name, starting with some sort of signal. But once generalized, it starts to write continuously: JoséJoséJoséJosé. It allows for various complexity levels, with different demands of abstracting and prediction. For instance, in order for the name repetitions to appear below each other, one must press ENTER to change line, after typing each "José".)

2. Programming as an extension based on automated behaviors

Building houses

This involves learning the automated behavior of trucks (by loading it with a robot and a box, it goes away and builds a new house). The child should confirm that it worked, by going out to see where the new house was built, he/she should try building more houses, etc. Specific features can be explored, such as: the new house is identical to the one from where the truck came; new houses appear near the house where the truck came from, not at random; if the robot loaded on the truck was previously taught, it will do what it was taught when it reaches the new house; what kind of strategy allows for faster house-building, etc.

The robot that builds houses

Instead of having the child build houses, he/she teaches a robot how to do it. He/she can then compete with the robot, to see who makes houses faster; he/she can learn that a robot works faster when the programmer's figure is standing up, and even faster when it isn't even inside the robot's house; and that there can be two robots working in different houses, so that new houses are built with different roofs, etc.

Birds that carry gifts

This involves the discovery of the birds' ability to carry to their nests any rectangular object they are given, wherever the nest is. Suggestions: experimenting with the nest somewhere else on the room floor; inside another house; hidden behind a picture; etc. There can also be activities where each child "owns" a bird and they exchange gifts among them.

Houses with our name on them

This is the usage of the roof sensors to place the children's names on the roof (or other decorations). It can have the practical benefit of allowing a child to identify to whom each house "belongs". By teaching a robot how to place a name on the roof, this can be combined with the house-building activity, so that the new houses display the name of their "owner."

3. Programming as a development of generic activities

Tidy up the room

After scattering many "toys" around the room floor, the boxes are joined together, in order to organize (tidy up) those toys. This can be combined with the vacuum cleaner, to vacuum unwanted things or to spit whaterver was vacuumed by accident.

Big/small, wide/narrow, tall/low

Training on the control of the air pump, associated with the development of the skill to identify dimensions and contraries.

Birds mass/picnic/army – crowds

Training on using the magic wand to create copies, producing in this fashion a large number of identical objects – which are then set in thematic context determined by the child.

The flower garden and the gardener robot

This can start as the third activity, above, but over a green rectangle of the image of a garden, rather than on the room floor or outside. Afterwards, a robot can be taught how to plant flowers (these must have a random movement behavior, otherwise they will be overlaid one on top of each other).

A painting for the outer wall

In this activity, the child combines images on a picture frame, creating a painting in the process (this involved training of dimensions, vacuuming and spitting, and ordering – the latter drawings are laid on top of the former). The complexity can be increased, taking advantage of the fact than when two images are overlaid, the resulting image is the size of the bottom one; the top picture is truncated to fit.

This can be expanded with a robot to place the picture on the wall; using it with a truck, the robot will place the picture on the wall of the new house.

Essential ToonTalk manipulation skills

Setting aside for now the programming skills, just using ToonTalk itself requires the development of specific skills:

- when pointing at an object, it wiggles and changes its color: that's the object one will pick up, or the one on top of which we'll drop what we're holdingm should the mouse button be pressed;
- the space bar turns on the vacuum cleaner and the air pump, which remain in operation until turned off[339]; this action must be coordinated intentionally, with the movement of the hand, so as to avoide vacuuming the wrong object (the same goes about spitting on the right place, enlarging, shrinking, etc.);
- robots only work inside houses (suggestion: "to avoid losing their toys outside");
- any work that children want to keep must be placed on an empty page of the main notebook;
- the entire city can be saved (associated to ToonTalk's login name), by pressing the "Guardar tudo" button, when exiting ToonTalk;
- all drawings are rectangular (as can be seen by "erasing" then, which shows the empty paper sheet used to draw it);
- birds are nice, tidy pets, they only carry rectangular stuff, including drawings and toys stored in boxes;
- robots are even tidier than birds: they only play when one gives them toys inside a box.

Children-adapted terminology

Original ToonTalk terminology	Adapted terminology
Toolbox	Toy box
Tools (Dusty, Pumpy, Magic Wand)	Tools
To program	To teach
Image	Drawing
[X] (room, roof, etc.) sensor	make-believe [X], magical [X]

[339] This was the actual behavior of ToonTalk at the time this document was written. Now, ToonTalk turns the tools off automatically (*vd*. section 3.3.5).

Children-adapted explanation

Abstraction by erasing the robot's thoughts

This concept was found to be employable, by adopting an approch of usefulness for the child. Specifically, by programming several different robots that do the same things, only with different thought bubbles (exchanges tree with flower; exchanges truck with bomb; exchanges boy with helicopter, etc.), these can be labeled as "picky", and the child can be shown that if we erase the drawings from the thoughts, they no longer care about the differences.

Fuller and more diverse approaches have not been developed yet: perception that one doesn't have to erase everything, just what needs to be generalized; difference between erasing the box contents and the box itself; etc.

Why does the truck build a house with a robot and a box?

So far, the adopted explanation has been:

"This is a construction-site truck; it will only build a house if it has a robot to live in it. And the robot only allows the truck to leave when it has its lugagge – the box."

Why do birds only carry drawings, numbers, letters or boxes?

"Because they are very tidy." This explanation can certainly be improved upon.

Why do robot only take things that are inside boxes?

"Because they are even tidier than the birds." This explanation can also be improved upon, but it is more consistent than the one used for birds.

Why does a robot stop if we vacuum its box?

Since it is very tidy, it won't do anything without its box, because afterwards it would have to leave everything scattered on the floor.

Why must we press the Esc key to stop teaching a robot?

Because it is flying in the clouds, at robot school. If we don't tell it to come down, with the Esc key, all the stuff is going to fall on the ground and get ruined.

Session reports – 2002/2003

Preschool name Room identification

Date

Jane Doe	**John Doe**	**Starting time:** 10h00	**End time:** 10h23
Predefined objectives:			
Session development and general outcomes:			
List of attached records (files):			
Comments:			

Jane Doe	**John Doe**	**Starting time:** 10h00	**End time:** 10h23
Predefined objectives:			
Session development and general outcomes:			
List of attached records (files):			
Comments:			

Jane Doe	**John Doe**	**Starting time:** 10h00	**End time:** 10h23
Predefined objectives:			
Session development and general outcomes:			
List of attached records (files):			
Comments:			

Annex VII
—
Summaries and highlights from the sessions conducted by computer-activity teachers in preschools

Mateus preschool

Computer-activities teacher profile

Name initials	FL
Age	21
Education	Bachelor in Early Childhood Education
Professional experience	He had been a computer-activities teacher at 5 preschools, in the two previous years. That had been his first full-time job, and was also his current job.
Computing skills	His degree included a course on basic computing. Had been a regular computer user for several years. Received regular training and guidance on computer use in preschool contexts, within the ICEI project.
ToonTalk training	Attended two training sessions on the use of the environment tools and basic programming.

The children

19 children: 8 girls, 11 boys; 2 three-year olds, 11 four-year olds, 6 five-year olds.

Group	Child 1 (Age, Gender)			Child 2 (Age, Gender)		
1	PL	4	Girl	LF	4	Boy
2	PR	4	Boy	JPM	4	Boy
3	AR	4	Girl	RJ	4	Boy
4	L	4	Boy	MT	4	Girl
5	AA	5	Girl	AC	4	Girl
6	MA	5	Boy	M	4	Boy
7	IX	4	Girl	LE	5	Boy
8	FG	5	Boy			
9	AF	5	Boy	T	5	Girl
10	R	3	Girl	JPF	3	Boy

Sessions table

Group	Session	Date	Start	End	Time
1	1	28-Out-2002	10:45	11:10	0:25
1	2	11-Nov-2002	9:40	10:06	0:26
1	3	18-Nov-2002	10:15	10:37	0:22
1	4	25-Nov-2002	9:35	10:01	0:26
2	1	28-Out-2002	10:30	10:55	0:25
2	2	4-Nov-2002	11:24	11:49	0:25
2	3	11-Nov-2002			
2	4	18-Nov-2002	9:40	10:02	0:22
2	5	25-Nov-2002	10:10	10:30	0:20
2	6	9-Dez-2002			
2	7	16-Dez-2002			
3	1	28-Out-2002	11:10	11:30	0:20
3	2	4-Nov-2002	9:40	10:05	0:25
3	3	11-Nov-2002	10:10	10:35	0:25
3	4	18-Nov-2002	10:06	10:35	0:29
3	5	25-Nov-2002	9:35	10:05	0:30
3	6	2-Dez-2002	10:07	10:31	0:24
3	7	9-Dez-2002			
3	8	16-Dez-2002			

Group	Session	Date	Start	End	Time
4	1	28-Out-2002	10:20	10:40	0:20
4	2	4-Nov-2002	10:05	10:26	0:21
4	3	11-Nov-2002	10:40	10:55	0:15
4	4	18-Nov-2002	10:40	11:10	0:30
4	5	2-Dez-2002			
4	6	9-Dez-2002			
4	7	16-Dez-2002			
5	1	28-Out-2002	9:40	10:13	0:33
5	2	4-Nov-2002	10:30	10:53	0:23
5	3	11-Nov-2002	9:45	10:28	0:43
5	4	18-Nov-2002	10:40	11:00	0:20
5	5	25-Nov-2002	14:10	14:36	0:26
5	6a*	2-Dez-2002	9:40	10:00	0:20
5	6b*	2-Dex-2002			
5	7a*	9-Dez-2002			
5	7b*	9-Dez-2002			
5	8a*	16-Dez-2002			
5	8b*	16-Dez-2002			

(*The children were paired by the teacher, due to absentees.)

Group	Session	Date	Start	End	Time
6	1	28-Out-2002	9:55	10:24	0:29
6	2	4-Nov-2002	10:55	11:22	0:27
6	3	11-Nov-2002	10:30	10:57	0:27
6	4	18-Nov-2002	11:15	11:35	0:20
6	5	25-Nov-2002	14:10	14:36	0:26
6	6	2-Dez-2002	9:40	10:00	0:20
6	7	9-Dez-2002			
6	8	16-Dez-2002			
7	1	28-Out-2002	11:25	11:55	0:30
7	2	4-Nov-2002	11:24	11:49	0:25
7	3	11-Nov-2002			
7	4	18-Nov-2002			
7	5	25-Nov-2002	15:11	15:39	0:28
7	6	9-Dez-2002	15:11	15:39	0:28
7	7	16-Dez-2002			

Group	Session	Date	Start	End	Time
8	1	28-Out-2002	11:00	11:18	0:18
8	2	4-Nov-2002	11:45	12:15	0:30
8	3	11-Nov-2002	11:30	12:07	0:37
8	4	18-Nov-2002	9:40	10:12	0:32
8	5	2-Dez-2002	10:05	10:25	0:20
8	6	9-Dez-2002	15:11	15:39	0:28
8	7	16-Dez-2002			
9	1	25-Nov-2002	14:40	15:06	0:26
9	2	9-Dez-2002			
9	3	16-Dez-2002			
10	1	28-Out-2002	11:00	11:18	0:18
10	2	4-Nov-2002	11:45	12:15	0:30
10	3	11-Nov-2002	11:30	12:07	0:37
10	4	18-Nov-2002	9:40	10:12	0:32
10	5	25-Nov-2002	10:40	11:08	0:28
10	6	2-Dez-2002	11:25	11:55	0:30

General introduction – all groups

This teacher paired the children in *ad hoc* fashion, and changed groupings from time to time. This is reported in the full reports of Annex VIII: whenever a session was conducted with a child from group X and another from group Y, its report is provided only once, in group X, and a note is included mentioning that the session matches another session in group Y. Conversely, in group Y a note is included referring the reader to the appropriate session in the reports for group X.

Although the sessions were conducted throughout the year, after a few months the teacher shifted his focus to different computer activities and used ToonTalk only sporadically. He provided no reports of these further events.

Sessions overview – Group 1

In all the four sessions, the children were having trouble navigating and controlling the ToonTalk environment. The teacher presented robot-programming during the third sessions, but he wasn't sure himself of what programming was, or how it was connected with goals and results. In the fourth session, the children inquired why were they playing only one game.

Highlights – Group 1

> *"(...) LF had some difficulty grabbing objects and vacuuming. This was due to the fact that he had not **assimilated** the "rule" that in order to grab any object or vacuum, he must first see whether the desired object is moving, **as I transmitted to him and exemplified**."*
>
> *(FL, session 1, my emphasis)*

> *"(...) PL only taught the robot how to take objects from the **toolbox**. I can remember neither how one exits the thought, nor how to make the test to check if the robot learned **something**."*
>
> *(FL, session 4, my emphasis)*

Sessions overview – Group 2

One of the children (JPM) has experienced some difficulties controlling ToonTalk, which the teacher neglected. Possibly due to that, that child did not enjoy the game right from the start, although he later managed to acquire the necessary degree of control. However, the other child (PR) did manage to acquire a nice level of control quickly and enjoyed the game.

PR learned how to program an exchanger robot well enough to be eager to show the method to the other child, which he did, successfully.

JPM felt some frustration due to the lack of variety in the computer activities.

Highlights – Group 2

*« (...) **I don't think these difficulties matter much**, since they have more to do with his hand dexterity than with game features. (...) JPM (...) didn't **like the game**. »*

(FL, session 1, my emphasis)

« JPM asked, at the start, for what reason were we always playing the same game. »

(FL, session 3)

« (...) taught the robot how to swap the places of objects and place them again in their proper positions. Afterwards, he took out scales from the toolbox, to know when the robot ends its task (this placement of the scales was done following my indications). »

(FL, session 4)

---

*« PR has been teaching JPM what he had learned [about robot programming] in the previous session. **He had no major difficulties demonstrating how to do it**. JPM didn't understand very well all the swapping that was done by PR, but with my explanation he managed to perform the activity without major difficulties. »*

(FL, session 7, my emphasis)

Sessions overview – Group 3

During the introduction to the ToonTalk environment, the teacher reports are mainly focusing on whether children are managing to deal with the environment or remember how to work with it. The details on specific elements used by the children, and how, are scarce.

In the first programming opportunity, the teacher didn't build up from what children were doing. They were delighted building houses but the teacher suggested the programming of an exchanger robot. But in the following sessions the children programmed several different robots and eventually their interests were met.

Most of the robots programmed by these children had no connection with specific goals; they were simply showing the robot how to do something for the sake of showing. And in most sessions children would combine their interests (building houses, combining boxes, giving things to birds) with the teacher's interests (programming robots with no specific purpose).

The teacher insisted on doing the ToonTalk sessions as scheduled, even though the children have at least once expressed their lack of interest in participating either in ToonTalk or in the planned activity.

Highlights – Group 3

*« AR had barely sat down and started playing right away, making more and more houses, without help. (...) **After she had built several houses, I suggested that she swapped objects inside the boxes**, a task she performed without difficulties. Then I explained to her that the robot could do the same thing, and that all she had to do was teach it. (...) RJ wanted nothing but building houses and blowing them up; **I couldn't drive him to perform the same tasks as AR**. »*

(FL, session 3, my emphasis)

---

*« **Both children said they didn't want to work in ToonTalk, at the start of the activity.** »*

(FL, session 6, his emphasis)

« RJ didn't want to do the proposed activity. Although I have tried to drive him towards my plan, I couldn't manage to persuade him away from his will to blow up houses in order to see the construction workers rebuilding them afterwards. »

(FL, session 7)

Sessions overview – Group 4

The two children had a large difference in mouse skills, with MT being much more experienced than L. MT was very keen on exploring her control of the environment, rather than programming robots.

The teacher was keener on following his planned course, rather than adapting it. His programming examples were also to reproduce the children's activity, rather than enhance it or provide useful results for the children.

Highlights – Group 4

« (...) MT (...) managed, right away, to perform with quite some skill the route with the chopper and also all the other tasks that she faced (...) L felt many difficulties from the start, for he mixed up the keys several times (...) not once did L grab the chosen object (...) and this was cause for some discomfort. »

(FL, session 1)

« She, in spite of my spurring, didn't want to program the robot. I programmed one to make copies, since this was what she had been doing, but even so she didn't want to try. For the next session I'll have to "force" her into programming. (...) For the next session I'll start with L making programs, to try and see if this way, I can influence MT, and avoid L from being influenced by her. »

(FL, session 4)

Sessions overview – Group 5

The two children had quite different fine motricity levels, with AA controlling the mouse easily and AC experiencing lots of difficulties.

The teacher took advantage of an incidental entry into a thought bubble to explain the teaching process, but demonstrated a program without goals (just giving the robot's box to a new bird). Unlike with other children, however, the programs in this group followed the interest expressed by the children.

The teacher has shown interest in understanding the motivations and ideas of children (for instance, see the highlight below regarding session 6a).

The child that was demonstrating major difficulties in control, when paired with L, from group 4, surprised the teacher by assisting L, explaining to him what to do, although incapable of executing the actions herself.

Highlights – Group 5

« (...) I asked her if she still remembered the way to teach a robot how to work and why not teach it to build houses. She took my challenge and gave the robot a box with a truck (I asked her the reason for being a truck and not something else, ***to check if she was already reasoning that we need a truck to build houses****; however, it was a random choice, which means that she had no such intention). »*

(FL, session 6a, my emphasis)

---

*« (...) Then he exited the thought and tested the robot. (****This entire task was performed by L following the indications of AC****.)*

AC was merely working on the street, placing several objects on the ground, and simply carrying them from one place to another. ***This surprised me a lot, because she had been constantly helping out L.*** *»*

(FL, session 6b, my emphasis)

Sessions overview – Group 6

Both children had an adequate level of mouse-control skills, but the younger (M) adapted faster to the manipulation of the ToonTalk environment.

The teacher demonstrates quite a different approach: instead of pushing forward a plan, he's expressing care for the children's interests (for instance, see the highlight below regarding session 3).

Mostly, the robots have no purpose: for instance, starting with a box with the letter "A", a robot picks up a box, puts a truck inside and then vacuums the box with the truck – and this is described by the teacher as "taught a robot how to vacuum trucks." The teacher doesn't seem to notice that it is useless to create a truck and vacuum it immediately after.

Highlights – Group 6

« (...) MA had the same problems of the previous session, although a bit less pronounced. M surprised me in a positive way; it was just a matter of reminding how to perform the various tasks, for him to start working on his own, not caring for anyone else. »

(FL, session 2)

« (...) although I tried to have him do something different, I didn't manage to, because I don't want to force him to make something that he doesn't feel like at the moment, for if I did so I might be endangering his entire performance and future interest in the game. »

(FL, session 3)

Sessions overview – Group 7

This group was based on the five-year old LE, since the four-year old IX only participated in a couple of initial sessions. LE controlled the mouse and the ToonTalk environment easily, I needed more time to get acquainted.

There is an "out of the blue" report of LE programming house-building and house-blowing reports, which leads me to consider that either the teacher was now very used to the concept or LE had seen or chatted with other children regarding this possibility.

The teacher reveals a desire to "show some service", as evidenced by the highlight below, about session 6.

Highlights – Group 7

« FG didn't want to perform the task on the room floor, and so found it a bit harder to do it in the robot's thought. However, with some help from me, of verbal nature, he managed to unswap what LE had done. (...) ***Had the activity been performed first by this child, I certainly would have had many more difficulties in fulfilling it, that's why I put him to do it after LE.*** *»*

(FL, session 6, my emphasis)

Sessions overview – Group 8

This is was a somewhat unexpected group, since it was composed of a five-year old (FG) and a three-year old (JPF). However, as was then seen in the first sessions, the three-year old managed a fast adaptation to the ToonTalk environment, compared to the five-year old FG.

The three-year old did a program in the fourth session, imitating FG, but in the following sessions he surpassed the five-year old FG. FG slowly developed his skills in the ToonTalk environment and programming, but the teacher didn't build up from his desire to blow up houses.

Highlights – Group 8

« The easiness demonstrated by JPF when performing the tasks is in fact something that surprised me, given that this child presents some difficulties in executing drawings on Paint ***and in playing computer games****. »*

(FL, session 2, my emphasis)

« In another house, JPF entered the robot's thoughts, picked up a bomb and made it blow up. He tested the robot and enjoyed seeing it blow up the house. JPF had no major difficulties programming the robot, very few times I had to intervene to warn him of this matter or the other. »

(FL, session 4)

Sessions overview – Group 9

Due to several absences, this group started the sessions several weeks after the remaining children. However, the children demonstrated being familiar with the operation of the environment, probably due to the fact that the computer is located within the main preschool activities room, rather than in isolation.

The teacher had poor expectations of AF, due to general cognitive problems in other preschool activities, but his initial performance in ToonTalk gave no indications of that, only in more complex actions was AF showing difficulties.

The teacher is again demonstrating greater attention to special details in the children's involvement with the process of programming, as shown by the third highlight, below.

Highlights – Group 9

« In the case of AF, I was already expecting his performance to fall short (he is a child with special educational needs, even though he's not declared as such). »

(FL, session 2)

« AF, contrary to my expectations, managed to fly and land the chopper, entered the house and started taking objects out of the toolbox and placing them on the house floor. This was entirely done without my teaching, not even a verbal explanation. I think that the fact that ToonTalk was installed in the PC at the activities room was a preponderant factor for this child having achieved, in the first session he attended, such a nice result. »

(FL, session 1)

« T (...) managed, almost without any help from me, to reach the end of the activity. The most important thing, for me, was the fact of T having understood why the images need to unswapped, during the activity. »

(FL, session 3)

Sessions overview – Group 10

This group of three-year olds was another odd assembly, because JPF had shown some skill in four sessions already (group 8) when he was paired with another three-year old, R, who had no previous ToonTalk experience. In her first session (this group's fifth), she simply watched JPF.

The teacher didn't demonstrate any particular interest in developing R's perspectives of computer use, as shown by the highlight below.

Highlights – Group 10

« She experienced some difficulties, namely while taking objects out of the toolbox, because to take out an object, she would take another instead. ***I didn't want to move on too much with R, since she's hardly ever present, and when she is she doesn't want to come to the computers.*** *»*

(FL, session 6, my emphasis)

Mondrões preschool

Computer-activities teacher profile

Name initials	RC
Age	25
Education	Professional training as Animator in Computing for Children. Second-year undergraduate student of Early Childhood Education.
Professional experience	Had previously worked two years as teacher/animator at a private computer-activities school for children, and also as a computer training teacher for adults. Afterwards, she had been working for three years as a computer-activities teacher at 5 preschools.
Computing skills	Power user (able to conduct computer training sessions). Able to customize and employ several programs in non-standard, programmatic ways (*e.g.*, game development using MS PowerPoint). Provided training and guidance on computer use in preschool contexts, for the ICEI project.
ToonTalk training	Attended two training sessions on the use of the environment tools and basic programming and had previous superficial contact with it.

The children

11 children: 4 girls, 7 boys; 1 three-year old, 4 four-year olds, 6 five-year olds.

Group	Child 1 (Age, Gender)			Child 2 (Age, Gender)		
1	L	5	Boy	R	4	Girl
2	RA	5	Boy	S	3	Girl
3	D	4	Boy	M	4	Boy
4	T	5	Boy	RF	4	Boy
5	SI	5	Girl	SM	5	Boy
6	P	5	Girl			

Sessions table

Group	Session	Date	Start	End	Time
1	1	9-Out-2002			1
1	2	16-Out-2002	9:00	9:50	1
1	3	23-Out-2002			1
1	4	30-Out-2005	9:15	9:45	1
2	1	9-Out-2002			2
2	2	16-Out-2002	10:30	10:45	2
2	3	23-Out-2002			2
2	4	30-Out-2005	9:45	10:15	2
3	1	9-Out-2002			3
3	2	16-Out-2002	10:45	11:05	3
3	3	23-Out-2002			3
3	4	30-Out-2005	10:15	10:45	3

Group	Session	Date	Start	End	Time
4	1	9-Out-2002			4
4	2	16-Out-2002	11:10	11:25	4
4	3	23-Out-2002			4
4	4	30-Out-2005	10:50	11:20	4
5	1	9-Out-2002			5
5	2	16-Out-2002	9:55	10:15	5
5	3	23-Out-2002			5
5	4	30-Out-2005	11:20	11:45	5
6	1	9-Out-2002			6
6	2	16-Out-2002	11:25	11:45	6
6	3	23-Out-2002			6

General introduction – all groups

For pairing the children, this teacher took in consideration not only their age, but also their previous experience with computers, since at least two children had used a computer regularly in the preschool during the previous two years. In the first session, all children tried out ToonTalk individually, major topics being the identification of the environment and basic navigation skills.

This teacher stopped providing reports on ToonTalk sessions after a few months, but she continued to use ToonTalk, albeit not with programming intent. She focused in ToonTalk activities where the environment was extensively customized by the children, in the context of extra-computer activities taking place in the preschool. Specifically, she started developing a "city of professions", each child customizing a ToonTalk house to match a specific profession. This included making freehand drawings in Paint and bringing them into the ToonTalk environment, but also building actual cardboard houses in the preschool, and even programming a physical floor turtle robot to move from one house to another, under several roles (policeperson, gardener, etc.) in the context of different activities.

Sessions overview – Group 1

The teacher took great care in allowing the children to acquire mouse-control over the environment, and also inquired the children on what they knew how to do, rather than just observe.

There is some indication of low teacher expectations, as shown by the slightly contradictory statement in the first highlight below, and also in the last session, when R showed pride in the houses she built, which surprised the teacher.

None of these children programmed robots, during the four sessions.

Highlights – Group 1

« What can be asserted is that at this moment (3rd session) R can manipulate the mouse. When I asked her if she still remembered how houses were built, R quickly explained it to me and then did it. Whenever she built a house, she would get the figurine up, left the house, and took off in the helicopter to see it. »

(RC, session 3)

« R took off in the helicopter and showed the houses to L... (I was surprised by this attitude of R.) L then wanted to do the same thing. »

(RC, session 4)

Sessions overview – Group 2

Both children present some difficulties in the control of the mouse, and the teacher has shown care, planning other computer activities to help her develop those skills.

The children enjoyed building houses and R even taught S how to do it and helped her. None of these children programmed robots.

Highlights – Group 2

« R built houses with my assistance (he has some trouble using the mouse). Then he told me he didn't want [to play] *anymore. »*

(RC, session 3)

« R is now perfectly capable of building houses and has been teaching S, who had no idea of how to do it (he helped her, placing his hand on top of hers). (...) *R still has some trouble to fully control the mouse, but he's managing it. »*

(RC, session 4)

Sessions overview – Group 3

These children found it difficult to control the mouse. But they managed to build houses in the third session, and teach a robot how to do it by the fourth session (a continuous house builder). They were surprised the large effects of their robot.

Highlights – Group 3

« D and M managed to build houses, and wanted to count them. After they did, I showed them how they could build a city. Both D and M managed to teach the robot, and went out into the street and up with the helicopter to see the development. They were very surprised with what they had done. I did notice that M was more surprised than did D. »

(RC, session 4)

Sessions overview – Group 4

The teacher provided some insight on her presentation of the house-building activity: she drew the child's attention to the fact that trucks move a little when dropped, and suggested trying out several things on its back to see if it moves again. When the child went out after the truck, she suggested using the helicopter to look for it, to see if the child would notice that the truck was nowhere to be seen, but that there was a new house available (with the robot and box inside).

They both managed to build robots to fill up a city, and enjoyed it. T asked to do it all over again, while D watched.

Highlights – Group 4

«T explained to R how to build a house, and at the same time built one. »

(RC, session 3)

«When T was in the helicopter, he started trying to go to the sea, but he couldn't go there. I told him that it was a bit complicated, since that sea was very deep and the game wouldn't let him go there, to avoid the danger of him falling into the water. »

(RC, session 3)

Sessions overview – Group 5

These two children were very skilled in controlling the mouse and had no trouble building houses and exploding them. The teacher presented suggestions but let them enjoy the counting activities they had spontaneously started while building houses. After having programmed a city builder robot, the children were a bit puzzled regarding the connection between the robot's operation and what had previously been done inside the thought bubble. The teacher employed a metaphorical explanation that she thinks was satisfactory for children.

Highlights – Group 5

«I tried suggestion that they developed a city, but none of them took the least notice, what they wanted was to build houses and count them in frenzy.

SM:"Do you want me to take a bomb out?"

SI: "Yes. Tuck it up, it can blow up!!!"

And in this manner they proceeded with building houses and counting them. »

(RC, session 3)

«They were both perplexed with how fast the construction workers built houses. They have shown difficulty in understanding the robot's thought:

SM – I'm a bit mixed up!?

SI – I am, too.

At this moment I explained them that when we want to do something, we must think about what we want to do, and so the robot also had to think... Since robots don't think by themselves, one has to go into their thoughts to teach them. I think they were convinced. »

(RC, Session 4)

Sessions overview – Group 6

Unlike the other children, who mostly started by building houses, P started by giving objects to birds, for them to take into their nests. She did build houses, but according to her teacher, she didn't associate the new houses with what she had done previously. However, the teacher had been mistaken, because in the following session, although P said that she didn't know what birds do, she then had the remarkable conversation with the teacher, presented below as the first highlight. As the conversation shows, P had perfectly understood the process, she just didn't want to bother saying "yes" to the teacher. After this conversation, she continued by building houses and giving things to birds, intertwining both tasks effortlessly.

Highlights – Group 6

« She told me that [in the previous session] *we had been building houses, and when I asked her what birds did, she told me she didn't know. Therefore, I decided to modify the session's goals, to do house-building. And I told her that we would be building houses. To my amazement, this was P's construction sequence: 1) took out a truck and a robot, placed the robot on the truck; 2) took out a nest and started giving things to the bird....*

RC – P, how does one build a house?

P – With a truck, a robot, and a box.

RC – And why did you take out a nest?

P – I felt like giving things to the bird for it to carry to the nest. »

(RC, Session 4)

Parada de Cunhos preschool

Computer-activities teacher profile

Name initials	CF
Age	23
Education	Bachelor in Teaching of Biology & Geology
Professional experience	This was her first job: computer-activities teacher at 3 preschools. She was also teaching at a highschool, for the first time.
Computing skills	Her degree included a course on applied computing and another on educational technologies. She received regular training and guidance on computer use in preschool contexts, within the ICEI project.
ToonTalk training	Attended two training sessions on the use of the environment tools and basic programming.

The children

14 children: 5 girls, 9 boys; 5 three-year olds, 6 four-year olds, 3 five-year olds.

Group	Child 1 (Age, Gender)			Child 2 (Age, Gender)		
1	B	3	Boy			
2	D	4	Boy			
3	R	4	Boy	G	4	Boy
4	M	4	Boy	DI	3	Boy
5	F	4	Boy			
6	MA	5	Boy			
7	MR	3	Girl	C	3	Girl
8	A	5	Boy			
9	AN	3	Girl			
10	CR	4	Girl			
11	CA	5	Girl			

Sessions table

Group	Session	Date	Start	End	Time
1	1	25-Out-2002	9:45	9:55	1
2	1	25-Out-2002	9:57	10:14	2
2	2	29-Nov-2005	15:51	15:54	2
3	1	25-Out-2002	10:16	10:31	3
3	2	4-Nov-2002	11:46	12:05	3
4	1	25-Out-2002	10:33	10:55	4
5	1	25-Out-2002	10:57	11:05	5
6	1	29-Nov-2005	14:15	14:55	6
7	1	25-Out-2002	11:47	12:05	7
7	2	4-Nov-2002	14:37	15:07	7
7	4	22-Nov-2002	11:18	11:41	7
8	1	25-Out-2002	14:20	15:05	8

Group	Session	Date	Start	End	Time
8	2	4-Nov-2002	10:14	11:05	8
8	3	15-Nov-2002			8
8	4	22-Nov-2002	10:00	10:25	8
8	5	29-Nov-2005	11:34	11:55	8
9	1	25-Out-2002	15:10	15:16	9
10	1	25-Out-2002	11:14	11:40	10
10	2	4-Nov-2002	10:10	10:42	10
10	3	15-Nov-2002			10
10	4	22-Nov-2002	10:31	11:12	10
11	1	4-Nov-2002	15:08	15:25	11
11	2	22-Nov-2002	14:17	14:31	11
11	3	29-Nov-2005	14:56	15:49	11

General introduction – all groups

This teacher was not experienced in dealing with preschool children, and felt uncomfortable using ToonTalk, but since she was juggling these sessions with teaching at a highschool in another county, there was little opportunity to provide her with more ToonTalk training. The ToonTalk sessions in this preschool were interrupted after a few months, but the computer is kept on throughout the day, and children can use it at any time.

Another issue was that many of these children were often absent from preschool, either for full days or part of a day. As a result, the computer-activities teacher was often forced to conduct single-child sessions. Also, some children would be missing preschool in the weekday scheduled for computer activities, sometimes several weeks in a row. For this reason, I present a summary of the first session, which for several chilren was the only recorded session, and then only per-group summaries for groups with whom further sessions were recorded.

Overview of the groups with a single recorded session

For most children, the single recorded session was the teacher's first, on October 25th, 2002. In three cases, the children experienced significant levels of difficulty in mouse skills: controlling the mouse movements was hard, as was clicking the mouse buttons only for a specific purpose, rather than haphazardly. In these cases, the teacher provided very short sessions, possibly daunted by the situation (group 1: 10 minutes; group 5: 8 minutes; group 9: 6 minutes). In these sessions, children simply landed the helicopter, entered a house, kneeled, picked up a few objects and dropped them, invariably failing to associate the wiggling with the focus of mouse clicks. This was also the case with the child M in group 4. It should be noted that the teacher attributed the difficulty of one child to her age (2¾ years) rather than to its lack of experience and practice in use of the mouse. This should be contrasted with the child AX, from the São Pedro Parque preschool, who demonstrated great mouse control, in spite of having quite a similar age.

The child DI in group 4 presented a higher level of mouse control, and the teacher unsuccessfully tried to show him how to build houses and blow them up.

One month later, the single recorded session for group 6 took place, with the child M. For this session, the teacher is keen to report how the child is using the program "without help", discovering the uses of the mouse buttons, and noticing the wiggling cues. He started questioning the teacher about the purpose of the various tools, and tried them out, and demonstrated great ease navigating around the ToonTalk environment, getting the figurine up to seek out stuff that was out of sight, entering and leaving houses, etc. – the session only ended when the teacher asked him to please allow another colleague to use the computer.

I believe that to this difference in results contributed both the greater experience of the teacher (both in conducting session with preschool children and with ToonTalk) and the fact that the child MA had the opportunity of a full month in contact with his colleagues who had already been using ToonTalk.

Highlights of the groups with a single recorded session

« I helped him to get up, but since he's constantly clicking, [the figurine] *is constantly kneeling. »*

(CF, Session 1, Group 1)

« M landed fast and kneeled without intending to. I helped him grab a few objects, but he didn't want try out further. »

(CF, Session 1, Group 4)

« I helped [DI] *grab objects and taught* [him] *how to build houses.* [He] *wanted to repeat* [what I did]*, but he no longer remembered which objects he needed. »*

(CF, Session 1, Group 4)

« F finds it very difficult to control the mouse, and since he's constantly clicking, he can't grab what he wants. »

(CF, Session 1, Group 5)

« AN can't manipulate the mouse, ***given her age*** *(2 years).* (...) *Since she can't use the mouse or click, she didn't want to play »*

(CF, Session 1, Group 9, my emphasis)

« Clicking ***without help****,* [MA] *found out how to land and lift off. He landed* ***unassisted*** *and* ***in a single*** *mouse-click, kneeled. He identified the hand and started taking objects from the tool box.* ***Unaided****, he found out that by pointing an object shakes, and that's the one grabbed by the hand. »*

(CF, Session 1, Group 6, my emphasis)

Sessions overview – Group 2

This child only had two short sessions, where he practiced the basic use of the environment.

Highlights – Group 2

« He didn't want to play anymore; instead he wanted to see what the mouse was made of. »

(CF, Session 1)

Sessions overview – Group 3

In two sessions, R & G practiced the manipulation of objects and the building of houses but were frustrated with their mouse-control difficulties.

Highlights – Group 3

« [R] kept on taking out many objects from the box and wanted to build many. I got muddled with all the objects scattered around and wasn't too successufl in building houses. I helped him with my hand. He didn't want to play any more. »

(CF, Session 2)

Sessions overview – Group 7

These two children also have many difficulties in mouse control, and feel a bit frustrated about it.

Highlights – Group 7

« MR and C are not capable of using the mouse, i.e., they are constantly moving it in a disoriented way. »

(CF, Session 2)

Sessions overview – Group 8

Unlike most of the other children, A has a nice level of mouse-control skills, and the difference this made in his appreciation of ToonTalk was visible. Regarding the programming of robots, the teacher pressed him on a programming suggestion unrelated to his current line of activity (he had been building houses) and he didn't react well (see the second highlight, below). The teacher insisted on this line of programming for three (!) more sessions, but A didn't really understand the process or cared about it. And he basically moved on exploring areas of ToonTalk which the teacher wasn't suggesting. The teacher kept on trying to push the idea of programming an exchanger robot, until eventually she gave up and, finally, proposed a programming activity connected to the house-building which, by then, was no longer the main focus of attention of A.

Highlights – Group 8

« I taught him how to build houses, and he wanted to try out. He was also successful and wanted to be the rest of the time building houses and checking the final result. »

(CF, Session 1)

« When he took out the robot, I suggested teaching it how to swap objects and showed him how to do it. I repeated it several times and asked him to do it on his own. But he didn't want to. »

(CF, Session 2)

« Without any guidance, he entered a house, kneeled and took out the notebook of drawings. He asked me how it was browsed. I explained him and showed him how to do it. He enjoyed seeing the drawings and took the one showing the toolbox. He wanted to take out many identical pictures and overlay them. I asked if he still remembered how to teach a robot about swapping objects. He said, "oh, I don't want to do that. I'd rather make houses." I then asked if he would prefer teaching the robot how to build houses. I preferred making a drawing. »

(CF, Session 5)

Sessions overview – Group 10

The child CR demonstrates some ease in manipulation of the environment, and already knew how to build houses in the first session (which took place in the afternoon of the first day). It is interesting that when she strayed away from the houses, the teacher couldn't remember how to call the helicopter and they both had to restart the program.

This child inquired about the purpose of the robot, but after explaining it, the teacher suggested the exchanger robot, unconnected with CR's current activities. Bur CR did kept on inquiring about other elements and the teacher used this to develop other activities (for instance, see the second highlight, below).

The teacher insisted on the exchanger idea over two more sessions, and CR eventually did it, but lacking interest and focus. However, CR did understand it well enough to be puzzled when she saw her robot repeat not only the intended actions but also her errors (see the third highlight, below).

Highlights – Group 10

« CR landed well, and wanted to enter a house but kept moving the mouse for too long and lost track of the site. Since she couldn't find the houses, she wanted to get back into the chopper. But she had also lost track of it. We had to exit Toon[Talk] *without saving and re-enter. She landed, entered a house and wanted to blow it up. Then she built a new one, all this without my help. »*

(CF, Session 2)

« She asked if the girl had no friends. I taught her how to browse the notebook and take out the drawing of a friend. She did it on her own. »

(CF, Session 2)

« Outside its thought, the robot repeated everything, including the "errors" and CR said: "This robot does confusing things!" »

(CF, Session 4)

Sessions overview – Group 11

This girl is often absent from the preschool, so she started using ToonTalk 15 days after the other children. But although she was five years old, she was much less at ease with the computer than usual, as shown by the first highlight, below. She did understood what was on the screen, but her limited motion control frustrated her (for an example covering both these statements, see the second highlight below). By the third session she was barely managing to manipulate the objects and move about the environment, and only with lots of help from the teacher. But she was eager to do different things and expressed what she wanted to achieve (see the third highlight, below), and even tried out different approaches with the help of the teacher.

Highlights – Group 11

« I let her at will to do whatever she wanted to with the mouse, and the result was that she tried to grab the objects by taking her other hand to the monitor. »

(CF, Session 1)

« CA no longer remembered the purpose of the mouse buttons. She found out by clicking randomly. However, she was no longer able to land the chopper. In an act of irritation, she moved the mouse vary fast and the chopper landed. »

(CF, Session 3)

« On foot, she saw that there were toys still not tucked in the toolbox and she wanted to tuck them in, even while standing up. I explained that one needs to bend down to get the toys and she tried out with one toy in the activities room [(outside the computer)]*. She tried several times to kneel* [in ToonTalk]*, but was always doing it away from the floor location of the toys. I helped her. »*

(CF, Session 3)

São Pedro Parque preschool

Computer-activities teacher profile

Name initials	PF
Age	23
Education	Bachelor in Sociology
Professional experience	Her first job had been as a computer-activities teacher at 5 preschools, in the previous year. This was also her current job.
Computing skills	Self-trained regular computer user. She received regular training and guidance on computer use in preschool contexts, within the ICEI project.
ToonTalk training	Attended two training sessions on the use of the environment tools and basic programming.

The children

21 children: 13 girls, 8 boys; 6 three-year olds, 8 four-year olds, 7 five-year olds.

Group	Child 1 (Age, Gender)			Child 2 (Age, Gender)		
1	A	3	Girl	D	3	Boy
2	M	3	Girl	R	3	Girl
3	MN	3	Girl	J	3	Boy
4	IN	4	Girl	C	4	Girl
5	L	4	Boy	MA	4	Girl
6	U	5	Girl	CA	5	Girl
7	CR	4	Boy	JG	4	Girl
8	IM	5	Girl	S	5	Boy
9	AM	5	Girl	V	5	Boy
10	AX	4	Boy	G	4	Boy
11	AL	5	Girl			

Sessions table

Group	Session	Date	Start	End	Time
1	1	10-Out-2002	9:08	9:28	0:20
1	2	15-Out-2002	9:56	10:19	0:23
1	3	24-Out-2002	14:10	14:29	0:19
1	4	29-Out-2002	10:11	10:41	0:30
1	5	5-Nov-2002	9:15	9:45	0:30
1	6	5-Dez-2002	14:37	14:57	0:20
1	7	23-Jan-2003	9:20	9:40	0:20
1	8	4-Fev-2003	10:25	10:45	0:20
1	9	13-Fev-2003	10:45	11:10	0:25
1	10	11-Mar-2003	15:05	15:28	0:23
1	11	27-Mar-2003	10:15	10:26	0:11
1	12	22-Mai-2003	10:18	10:37	0:19
1	13	5-Jun-2003	9:20	9:50	0:30
2	1	10-Out-2002	0:30	0:46	0:16
2	2	15-Out-2002	9:20	9:53	0:33
2	3	24-Out-2002	11:35	11:58	0:23
2	4	31-Out-2002	9:10	9:36	0:26
2	5	7-Nov-2002	9:20	10:40	1:20
2	6	5-Dez-2002	10:40	11:02	0:22
2	7	23-Jan-2003	11:00	11:20	0:20
2	8	4-Fev-2003	9:15	9:40	0:25
2	9	13-Fev-2003	9:10	9:44	0:34
2	10	18-Mar-2003	10:20	10:45	0:25
2	11	27-Mar-2003	11:15	11:36	0:21
2	12	20-Mai-2003	14:15	14:40	0:25
2	13	5-Jun-2003	11:10	11:46	0:36
3	1a	10-Out-2002	9:48	10:03	0:15
3	1b	10-Out-2002	15:06	15:30	0:24
3	2	17-Out-2002	14:15	14:46	0:31
3	3	24-Out-2002	14:35	15:00	0:25
3	4	31-Out-2002	10:27	10:50	0:23

Group	Session	Date	Start	End	Time
3	5	7-Nov-2002	9:50	10:18	0:28
3	6	5-Dez-2002	11:05	11:34	0:29
3	7	4-Fev-2003	11:00	11:26	0:26
3	8	18-Fev-2003	10:25	10:43	0:18
3	9	11-Mar-2003	15:30	15:54	0:24
3	10	27-Mar-2003	10:28	11:00	0:32
3	11	20-Mai-2003	14:55	15:20	0:25
3	12	3-Jun-2003	11:00	11:30	0:30
4	1	10-Out-2002	10:05	10:22	0:17
4	2	15-Out-2002	10:22	10:50	0:28
4	3	24-Out-2002	10:00	10:24	0:24
4	4	31-Out-2002	9:45	10:15	0:30
4	5	7-Nov-2002	9:15	9:46	0:31
4	6	5-Dez-2002	10:12	10:38	0:26
4	7	23-Jan-2003	9:45	10:00	0:15
4	8	4-Fev-2003	14:03	14:20	0:17
4	9	13-Fev-2003	9:45	10:06	0:21
4	10	18-Mar-2003	10:47	11:05	0:18
4	11	27-Mar-2003	9:40	10:14	0:34
4	12	22-Mai-2003	9:14	9:44	0:30
4	13	3-Jun-2003	14:05	14:35	0:30
5	1	10-Out-2002	10:25	10:45	0:20
5	2	17-Out-2002	11:15	11:40	0:25
5	3	24-Out-2002	11:00	11:35	0:35
5	4	29-Out-2002	15:30	16:00	0:30
5	5	5-Nov-2002	10:48	11:15	0:27
5	6	5-Dez-2002	15:32	15:56	0:24
5	7	4-Fev-2003	15:07	15:30	0:23
5	8	18-Fev-2003	10:45	11:07	0:22
5	9	18-Mar-2003	15:10	15:40	0:30
5	10	1-Abr-2003	14:32	14:46	0:14

Group	Session	Date	Start	End	Time
5	11	22-Mai-2003	9:46	10:06	0:20
5	12	5-Jun-2003	10:30	11:00	0:30
6	1	10-Out-2002	11:40	12:04	0:24
6	2	15-Out-2002	11:00	11:36	0:36
6	3	24-Out-2002	15:03	15:25	0:22
6	4	29-Out-2002	9:25	10:08	0:43
6	5	5-Nov-2002	9:50	10:15	0:25
6	6	5-Dez-2002	9:49	10:10	0:21
6	7	4-Fev-2003	11:30	11:45	0:15
6	8	18-Fev-2003	9:20	9:48	0:28
6	9	18-Mar-2003	9:10	9:35	0:25
6	10	1-Abr-2003	11:38	11:58	0:20
6	11	20-Mai-2003	15:25	15:49	0:24
6	12	3-Jun-2003	9:30	10:07	0:37
7	1	10-Out-2002	10:58	11:19	0:21
7	2	17-Out-2002	9:41	10:03	0:22
7	3	24-Out-2002	15:54	16:08	0:14
7	4	31-Out-2002	11:30	12:00	0:30
7	5	7-Nov-2002	10:44	11:10	0:26
7	6	3-Dez-2002	11:26	11:53	0:27
7	7	4-Fev-2003	11:48	12:00	0:12
7	8	18-Fev-2003	11:10	11:34	0:24
7	9	18-Mar-2003	11:40	12:04	0:24
7	10	1-Abr-2003	13:48	14:10	0:22
7	11	22-Mai-2003	11:20	11:35	0:15
7	12	5-Jun-2003	9:52	10:16	0:24
8	1	10-Out-2002	14:25	14:48	0:23
8	2	17-Out-2002	15:20	15:52	0:32
8	3	24-Out-2002	9:10	9:51	0:41
8	4	29-Out-2002	14:06	14:40	0:34
8	5	7-Nov-2002	14:15	14:46	0:31
8	6	5-Dez-2002	14:10	14:35	0:25
8	7	23-Jan-2003	10:09	10:28	0:19
8	8	4-Fev-2003	14:22	14:40	0:18
8	9	13-Fev-2003	15:40	16:00	0:20
8	10	18-Mar-2003	11:05	11:34	0:29
8	11	1-Abr-2003	14:16	14:45	0:29

Group	Session	Date	Start	End	Time
8	12	19-Mai-2003	14:40	15:10	0:30
8	13	3-Jun-2003	10:35	10:55	0:20
9	1	10-Out-2002	14:50	15:05	0:15
9	2	15-Out-2002	14:45	15:15	0:30
9	3	24-Out-2002	15:30	15:52	0:22
9	4	29-Out-2002	14:42	15:12	0:30
9	5	7-Nov-2002	11:34	11:59	0:25
9	6	5-Dez-2002	14:58	15:10	0:12
9	7	4-Fev-2003	9:50	10:20	0:30
9	8	13-Fev-2003	10:10	10:42	0:32
9	9	18-Mar-2003	11:40	12:03	0:23
9	10	1-Abr-2003	14:12	14:30	0:18
9	11	19-Mai-2003	15:12	15:35	0:23
9	12	5-Jun-2003	15:02	15:22	0:20
10	1a	15-Out-2002	11:40	12:02	0:22
10	1b	15-Out-2002	14:04	14:25	0:21
10	2	24-Out-2002	10:30	10:45	0:15
10	3	29-Out-2002	10:51	11:26	0:35
10	4	5-Nov-2002	10:17	10:45	0:28
10	5	5-Dez-2002	9:20	9:47	0:27
10	6	23-Jan-2003	10:30	10:40	0:10
10	7	4-Fev-2003	14:42	15:05	0:23
10	8	18-Fev-2003	10:00	10:20	0:20
10	9	18-Mar-2003	14:06	14:41	0:35
10	10	25-Mar-2003	15:23	15:45	0:22
10	11	19-Mai-2003	15:37	16:02	0:25
10	12	5-Jun-2003	14:35	15:00	0:25
11	1	31-Out-2002	14:07	14:34	0:27
11	2	5-Nov-2002	11:40	12:02	0:22
11	3	5-Dez-2002	15:12	15:28	0:16
11	4	4-Fev-2003	15:31	15:46	0:15
11	5	18-Mar-2003	15:42	15:57	0:15
11	6	25-Mar-2003	14:55	15:17	0:22
11	7	20-Mai-2003	15:25	15:49	0:24
11	8	3-Jun-2003	10:10	10:30	0:20

General introduction – all groups

This teacher conducted ToonTalk sessions until June and provided reports on them for the full year, not just for the first few months. She combined the ToonTalk activities with freehand drawing activities in Paint and other kinds of activities, aiming to improve children's mouse-using skills and provide some variety in computer use. She also included some elements of extra-computer activities in the ToonTalk activities, as for example the ordering of pictures from the Peter Pan story that had been used earlier in the preschool.

Session overview – Group 1

Since these were 3-year olds, the teacher went at a very careful pace, allowing them to get acquainted with the environment and with the control of the mouse. She avoided keeping them longer than necessary in the ToonTalk environment, providing it just as long as it kept them interested. She also made sure children were understanding verbal concepts such as "swap", by letting them practice it in boxes and images, before moving on to programming.

In programming, A managed to understand the concept of swapping, but in the generalization she only understood the visible effect of "swapping always", not the "swapping because of no constraint". For D, the whole concept of teaching how to swap was too abstract for his comfort.

Highlights – Group 1

« If I don't say anything, D stays "stunned" looking at the ToonTalk environment and playing with the mouse. Today he played with the mouse by making moise as if it was a toy car. »

(PF, Session 6)

« D didn't know how to explain the swap or what it was. I opted for the comparison box. D identified the identical images, but couldn't understand the swap. He managed to make the swap, but only to place identical images below the matching ones.

PF – "What is the swap, A?

A – "To swap is to pass!... over to the other side". A did the swap skillfully. »

(PF, Session 7)

« A said "To swap is exchanging money. He did the swap well."

She wouldn't leave the robot's thoughts.

PF – When you give it the box, what does he do?

A – He swapped.

PF – And why?

A – Because he's done. »

(PF, Session 11)

Session overview – Group 2

These children started out with no acquaintance with computers, to the point of shouting at the figurine, expecting it to react. The teacher tried to help them develop some autonomy regarding mouse control, helping them move about, pick up things and drop them. Some adequate level of control started to be visible by session 6 (see the second highlight, below). By session 8, they were already making houses and training the act of swapping things, albeit still requiring some assistance. However, even with enhanced control, sometimes children still forget that they're in control (see highlight from session 11, below). They put on the walls personally-meaningful images (see the third hihglight, below), and noticed several oddities about the environment (for instance, the doors of the houses never close when one's inside).

Highlights – Group 2

« As for the figurine, she tried commanding it by speaking to it: "go to the airplane, shucks!" »

(PF, Session 1)

« They're both more calm while working, and stopped clicking aimlessly. They realized that they need to turn the vacuum cleaner off in order for it to stop making noise. »

(PF, Session 6)

« M – Drop the box, hand. It's taking it for ever to drop the box, PF! (she's not realizing that she's the one controlling the hand).

R – The tree has eyes.

M – To see well.

R – I don't like trees with eyes. »

(PF, Session 11)

« R sought in the notebook. She constructed meaningful settings to decorate the wall. One of them was a boy, a plane, and an explosion. She said: "the boy is going to see it blowing up". »

(PF, Session 13)

Session overview – Group 3

These children, although of similiar age, have very distinct levels of computer expertise. J is very accustomed to using the mouse and computer games, while MN had never used a computer before, and found most events entertaining, laughing about almost anything, even trivial events such as page being printed.

J easily grasped how to use the magic wand and other environment tools, such as the vacuum cleaner (although sometimes trying to drag each item instead of picking and dropping – possibly since that's a common technique in CD-ROM titles). MN requires support, since she often clicks the mouse by accident, but perhaps from seeing J she is presenting a faster development than the other groups of three-year olds (groups 1 & 2), in terms of understanding (for instance, realizing she needs to turn the vacuum cleaner on & off), and in terms of motion control (she is able to control the mouse after seven sessions).

By session 7, J managed to swap two things and teach a robot how to do it, albeit without associating the entry box with the one were the swap show take place; he made a new box instead. In further sessions, he is quite capable of expressing the operation of the robot and understanding matching problesm (see the last highlight, below).

Highlights – Group 3

« While playing with the helicopter, he asked: "Is it crazy?" I answered: "no, you're the one moving it, aren't you?"

J: Yes. »

(PF, Session 4)

« I explained them that in order to have identical boxes we had to place the images in the same order. J explained: "yeah... bird/bird, mouse/mouse, yellow flower/yellow flower, pink flower/pink flower,tree/tree..." »

(PF, Session 9)

« MN is at will picking and dropping objects in ToonTalk. This was a victory! »

(PF, Session 10)

« When leaving the robot's thought, J asks: "Shall I give it this box?" – As he points to the starting box.

PF – Why does it stop?

J – Because he's finished doing this.

J says that he taught "Hugo" how to swap the places of things. J tried giving a box with several holes to the robot. Then he realized that he only had taught it with two holes, and therefore it wouldn't work out this way. »

(PF, Session 10)

Session overview – Group 4

These children had no trouble controlling the environment objects, and so after some acquaintance sessions, the teacher intended to introduce robot programming in the fourth session. However, some difficulties in environment navigation and use of tools delayed this until session 8. Instead, they explored the building of houses and their customization.

In programming, the teacher seems to have gotten so acquainted with the process, to the point of assuming the children aren't "testing the robot", right in the first programming session. She didn't consider the possibility of the simple concept of "using the robot" being still amiss. However, children realized in this first programming session the need for contents in the box (see the second highlight, below). However, in the following session the teacher things they still aren't ready.

Highlights – Group 4

« They don't leave the robot's thoughts. They say:"I think it learned!". They don't check to see if the robot learned what they taught it. »

(PF, Session 8)

« I asked why was it that the robot stopped after doing what they had taught it. C's answer was: "The box is empty." »

(PF, Session 8)

« I didn't move on with robot programming, because in terms of reasoning I don't think IN and C are ready; perhaps within a couple of sessions. »

(PF, Session 9 – cf. the two previous highlights)

Session overview – Group 5

This group started by exploring several tools and behaviors, such as the vacuum cleaner and the bomb. For the first sessions, they devoted a lot of time to freehand drawing in Paint, but then they resumed the exploration by using the house sensors.

In robot programming, they understood several elements of the process (for example, see the third highlight, below). But the teacher was not confident in their cognitive abilities, as is demonstrated by the last highlight, below. In that highlight, there are two problems. Firstly, there is a contradiction in saying that the children notices that two walls are identical and is not achieving abstraction. Secondly, the child obviously feels he doesn't need to check the roof, because he's sure about the operation of the small "make-believe" roof. But the teacher is mistaking this as "confusion" on the part of the child.

Highlights – Group 5

« L, after blowing up each house, would lift off to check that there one less house [in the city]. *»*

(PF, Session 1)

« MA required assistance to find the sensor notebook. Then, she managed to find the wall and decorate it, which gave her great joy, upon seeing on the wall the Santa Claus she had drawn. MA always lifts the figurine to check out what she did. »

(PF, Session 6)

« I asked MA what swapping is. She said: "it's putting one here and another there" – she pointed as she explained.

PF – What is the robot thinking?

MA: It's thinking that it's going to do the game again.

I explained that it only does the game if we give it an identical box.

PF – Why did it stop?

MA: Because it's thinking about another box.

Then MA tried swapping the objects and tested the robot again. »

(PF, Session 10)

---

*« L noticed that the wall is identical to the one in the notebook. The abstraction is still not being achieved. Only when I tell them can they understand that they need to use a "make-believe" wall. (**Session 11**)*

*L finds everything complicated. He won't check whether the name is on the roof, he says that "it's on the small one, therefore it is." (**Session 12**) »*

(PF, Sessions 11 & 12; the contradictions in the teacher's reasoning have been underlined)

Session overview – Group 6

One of the children in this group already knew how to use a house-builder robot, and used it in the first session. The teacher attributed this to the fact that children have free access to the computer, unaided. I may add that it also due to the fact that several children from this preschool have learned how to program a house-builder robot, in the sessions of the previous year. This is all the more significant because CA hadn't much experience with ToonTalk: in the second session, she didn't know how to get the figure to stand up. Both CA and the other child, U, demonstrated being skilled in the use of the mouse.

Along the first 7 sessions, the teacher assisted CA and U in the development of several customizations and environment activities, realizing that they had several basic difficulties, such as mixing the pump's configuration keys and the space bar, to turn it on and off; or locating a room sensor within a notebook. She also reports, in session 7, that they didn't understand "well" how to teach a robot, but provides no details of this.

In the 8th session, however, CA knew that to teach a robot she needed to go to the robot's thoughts, and then taught a robot how to put a picture on a frame, albeit not testing nor using it afterwards. U also demonstrated that she understood the process, as is shown in the 3rd and 4th highlights, below, albeit mixing part of what she recalled with her imagination. But she also twice expressed her desire to have more variety in computer use; she was fed up of playing almost only with ToonTalk (which hints that she wasn't using the computer besides the sessions with the computer-activities teacher).

Highlights – Group 6

« CA already knew how to use the "builder" robot, because ToonTalk is installed in the preschool computer and the children usually explore it freely, unassisted. – ***Session 1***

CA didn't know how to get the figurine to stand up. – ***Session 2*** *»*

(PF, Sessions 1 & 2)

« CA moves around with no need for my indications. She found the frame and set the drawing. To teach the robot, CA knew how to reach the thoughts "We must go up to the robot's thought." She learned how to teach it, but she didn't test to see if it learned. »

(PF, Session 8)

« I told them that a truck and a box were required. I also explained that they needed a robot to be the builder. (...) *I had to tell them, when they finished teaching the robot, that in order to see if it had learned, we had to give it a box identical to the one in its thoughts. They didn't use the original box.*

PF – Why did it stop?

U – Because it already build the house, and the box is empty.

I had to remind U to give a box with the required items to teach the robot. U wasn't going to use the original box. »

(PF, Session 9)

« U remembered using the magic wand in the thought bubble. I asked why the magic wand was being used. CA answered that it was in order for the robot to "do always the same". When testing the robot, CA doesn't use the original box.

I asked why was it that the robot they taught wasn't stopping. U answered that they had taught it some magic tricks. »

(PF, Session 10)

Session overview – Group 7

In this group, JG was the three-year old child J, from the ToonTalk research sessions the previous year. She now also played in ToonTalk at her home computer. Both JG and CR knew how to teach a robot to build houses, and that was precisely CR's initial activity, in the very first session. They demonstrated great ease in using the environment and its tools. Besides J and CR's ease in using the environment, and even in programming, they both get great joy in each further achievement, such as placing their names on the roof, and setting birds inside houses to "label" them without things on the roof. They also wanted to customize the helicopter, which isn't currently possible. CR was not as experienced as J, but managed to follow along in most situations, albeit showing some confusion from time to time in programming, as evidenced by the fourth highlight, below.

Highlights – Group 7

« CR made a robot to build houses. Then he made 3 more. They already know how to get up and sit down without any difficulty.

CR: "I'm going to make many houses."

JG used the magic wand: "ah! Got it! More trucks!"

They used the bomb/grenade. I told them: "Are we going to blow up all the houses?" Them: "Yes, because some build themselves afterwards!"

They took drawings out of the notebook and stored them in boxes. When picking the spaceship, J said: "We must make this smaller!" And she fetched the air pump. J did a "box of stuff" and saved it in her notebook. »

(PF, Session 1)

« PF – "If we join the boxes..."

CR – "I know, I know!... The mouse comes and joins them!" »

(PF, Session 3)

« JG learned right away how to put her name on the roof. Upon seing the result, she started stomping her feet in joy and said: "I'm making another one!" »

(PF, Session 4)

« PF – How do I program the robot?

JG – I go to the thoughts.

PF – To know if the robot learned how do I do?

JG: I leave the robot's thought.

JG employs the original box to test the robot. Since they had copied the box, they asked if they should wait for the robot to "do always the same". I explained that they could stop the robot.

CR couldn't remember how to get to the robot's thoughts. He then taught the robot how to build a house, since he didn't manage to teach it the swapping game. After he had left the robot's thoughts, I asked CR: What is the robot thinking?

CR: It's thinking about doing what I taught it. »

(PF, Session 9)

« I started by asking what would be necessary to teach the robot how to place the drawing on the wall.

JG – I need the drawing and the wall.

We placed both things in the box, to teach the robot. JG taught without my assistance. »

(PF, Session 11)

Session overview – Group 8

There was a huge skill difference between the two children in this group. S already knew how to make a "builder robot", while for IM the first session was also her first contact with a mouse. The teacher also reports that IM has "educational needs". This situation is reflected in the teacher's goals for the second session (first highlight, below).

This resulted in slow exploration of ToonTalk details with IM, under the support of S, for several sessions. S would explore small bits of ToonTalk. For instance, he noticed the behavior of birds in the third session. IM was responding well to this support (see the second highlight, below).

In robot programming, S managed to program a robot to decorate the house and understood the benefit of it as "being faster" than placing thing on the house himself. IM experienced several difficulties but in session 10 did refer that the robot stops because its box is empty. And in the following session she managed to explain in adequate detail what is happening (third highlight, below).

In session 12, the teacher realized that IM wanted to leave the computer because she wasn't understanding what was being requested of her, and by letting her play freely, IM only stopped playing after doing several other activites by her own initiative.

Throughout the sessions, S was constantly demontrating ease of use on ToonTalk, but the time devoted to IM meant that he didn't receive as much support for progress as he might have had in a more balanced group.

Highlights – Group 8

*« **Predetermined objectives:***

to develop concentration – w/ IM;

to develop his relationship with co-workers – w/ S. »

(PF, Session 2)

« IM said she had to be at the computer: "I must work very much, in order to learn." IM likes to work, and to experiment, and tries to make things on her own. I had to her to "let me" assist her, because she was saying "I can't do it", but was avoiding help. »

(PF, Session 4)

« IM – It's doing the same because I taught it how to do. – and she continues the explanation, at my request – It stopped because it wanted to stop. Because it only swaps the way I taught it. »

(PF, Session 11)

« When ordering more houses to be built, S has no problems in using the magic wand to make many trucks and boxes. To teach the [name-on-the-roof] *robot, S said he had to make "the same thing.". And he taught the robot correctly. To test it, he gave the robot the box he had originally prepared, but he didn't go out to check if the name was indeed on the roof. »*

(PF, Session 13)

Session overview – Group 9

The five-year old AM had never played with a computer, so she required activities to train her mouse skills. V was more experienced, and the teacher wanted him to be a strong support for AM. It should be noted that AM by the 3rd session still didn't manage to identify the houses as such in the top-down view.

V had little trouble using the mouse, but overly fantasized the ToonTalk environment (*e.g.*, the second highlight, below). As for AM, the teacher constantly reports that she is "distracted". However, the teacher also reports her as focused when there is a purpose for an activity, beyond ToonTalk itself (see the third highlight below for an example).

As for robot programming, V programmed a roof-naming robot in session 7 and a wall-decorating robot in session 8. In session 9 he was already demonstrating a nice understanding of the process (see the fourth highlight, below), but his over-fantasizing of the environment was constatly getting in the way of a more effective use of ToonTalk. AM didn't manage to understand the process of programming (see for instance the fifth highlight, below).

Highlights – Group 9

« She made 2 twin birds, which she named "the bird brothers", and then I helped her save a box with them in her notebook. »

(PF, Session 1)

« As soon as V entered the ToonTalk house, he said, "I'm going to blow up the house." However, he forgot that in order to blow up the house he had to hold the bomb in the figurine's hand.

PF – "What can we do to find out whose is each house?"

V – "We can give money to the man and stay there!"

I explained that also in drawings the name allows us to know who made them. »

(PF, Session 4)

« Today, AM was very distracted, she only focused when I told her that we could make gifts (with the boxes) for her to offer. »

(PF, Session 6, when Christmas was nearing)

« PF – Why did it stop?

Instead of answering, V filled up the box again, matching the stuff the robot had in its head. »

(PF, Session 9)

« PF – Why did it stop, AM?

AM – I don't know. »

(PF, Session 10)

Session overview – Group 10

AX is the younger brother of D, who took part in the sessions of the previous year. He reportedly used ToonTalk in the preschool, without adult supervision, since he was 2 ½ years old. During these sessions, he turned 4. In the very first session, he demonstrated to the teacher how to program a robot to build houses (see the first highlight below). G was also acquainted with ToonTalk, and also knew how to make a house-builder robot, but demonstrated less at ease with the vacuum cleaner and magic wand (see the second highlight, below). Although AX didn't demonstrate any

difficulty using these tools, he did become surprised when he learned how to use the enlarging tool, in session 4. AX did have a somewhat confusing way of interpreting his programs, but managed to catch the implications of what he was told quite quickly, and even simplified the teacher's descriptions (regarding these situations, see the third highlight, below). Both AX and G cooperated with ease, even proposing to each other to employ birds to carry things around.

As sessions progressed, the teacher noticed that although AX understood the need to compose an entry box for the robot, he had the tendency to give it an empty box, since he got used to the fact that this would take him into the robot's thoughts. The teacher also reported some confusion between his actions and the robot's actions, when playing in the thought bubble, but his answers to queries show otherwise. But yet another teacher comment is that AX forgets to exit the robot's thoughts, and this may provide a clue to solving these contradictions (see the fourth highlight, below).

G missed several sessions, and didn't get as much assisted practice as AX. And programming for him never reached the point of becoming a "natural" activity, as it is for AX, who programmed robots for most of the tasks he usually did, such as decorating a house, branding a roof, etc. AX even taught G how to teach robot (see the fifth highlight, below).

Highlights – Group 10

« The city that AX has since the school year 2001/2002 is filled with many houses. On his own, he went to the robot's memory.

PF – "What are you doing, AX?"

AX – "I'm teaching the robot how to make houses. Look at this..." »

(PF, Session 1a)

« Back into ToonTalk, he [G] *made a builder robot (he already knew how to).* (...) *As for the vacuum cleaner and the wand, he still experiences some difficulties in understanding that he must hold them in the hand in order to use them. »*

(PF, Session 1b)

« PF – What uses can the boxes have?

AX – "To put things in."

Eu – "Then, if this is our travelling case, what will you put inside?"

AX – "I'll take with me a robot, a truck, a bomb, a number, a truck, and scales. There: it's done."

He entered the robot's thoughts: "I'm going to teach the robot how to explode the box..."

PF – What are you teaching to it, anyway?

AX: How to make houses.

(...)

AX: the birdies carry the things to the nest, so their little children can eat.

I explained to AX that if he gives an empty box to the robot, what he may teach him it'll do always in sequence.

AX: it'll keep on doing it, is it? »

(PF, Session 2)

« In the robot's thoughts, AX doesn't realize that he's now controlling the robot, and continues to act as himself[340]*.* (...) *AX forgets to exit from the robot's thoughts.* (...) *PF: And if the box gets empty, what happens?; AX: "It won't have anything to work with." »*

(PF, Session 9)

« AX taught G how to make a swap and how to teach the robot. AX told G to give the box to the robot's hands, to see if it learned. G still forgets to give the box to the robot to teach it. Inside the robot's thoughts, G remembered to use the magic wand, but he doesn't leave the thoughts. »

(PF, Session 10)

[340] *I.e.*, as if he was still controlling the programmer figurine.

Session overview – Group 11

This girl started taking part in the ToonTalk session two weeks after the other children. She demonstrated ease in doing the basic actions, and even some complex actions such as building houses and putting pictures on the roof (see the first highlight, below).

In programming, she was showed consideration for the necessary items, and demonstrated ease in programming (see the second highlight, below). She was also very keen on noticing that the robot was using the vacuum cleaner, when she hadn't taught it to do that, and on the reddening of a mismatched image (see the third highlight, below).

Highlights – Group 11

« She likes to build houses and exits, so that, in the helicopter, can see the houses she built: "Look at the houses I sent to be made," she said while pointing.

Without any instructions from me, she wanted to put her name on the roof. She only asked how to change pages in the notebook. »

(PF, Session 2)

« To teach the robot how to make houses, AL thought she just needed a box and a truck, because she already had a robot (the one being taught). She mentioned having to exit the robot's thoughts. She doesn't use the original box, but learned to copy the box, so that the robot doesn't stop. AL –"We have to teach things, and to avoid stopping, we must teach it to make tricks with the magic wand." »

(PF, Session 5)

« AL: Why does the tree turn red?

PF – Because it has already used the tree[341].

(...)

AL taught the robot again and remembered to give it the box to see if it had learned. She says that she didn't teach it to vacuum, but it does vacuum[342]*. »*

(PF, Session 6)

[341] This remark by the teacher is actually quite confusing, or even plain wrong. The tree had turned red because the swap had been made and in the tree's position there was now a different image.
[342] She's probably right. ToonTalk robots automatically vacuum any objects left on the floor at the end of a cycle.

São Vicente de Paula preschool

Computer-activities teacher profile

Name initials	G
Age	28
Education	Bachelor in Forestry Engineering
Professional experience	Her first job had been as a computer-activities teacher at 5 preschools, in the previous year. This was also her current job.
Computing skills	Experienced computer user, having attended both an Office course and two courses on Computer-Aided Design. She received regular training and guidance on computer use in preschool contexts, within the ICEI project.
ToonTalk training	Attended two training sessions on the use of the environment tools and basic programming.

The children

22 children: 14 girls, 8 boys; 0 three-year olds, 12 four-year olds, 10 five-year olds.

Group	Child 1 (Age, Gender)			Child 2 (Age, Gender)		
1	IT	5	Girl	ACS	4	Girl
2	F	5	Boy	M	5	Boy
3	IF	4	Girl	MG	4	Girl
4	AT	4	Girl	CP	4	Girl
5	FM	4	Boy	D	4	Boy
6	FS	5	Boy	C	5	Girl
7	FL	4	Boy	DI	4	Boy
8	B	5	Girl	CA	5	Girl
9	BB	5	Girl	L	5	Boy
10	J	5	Girl	MA	4	Girl
11	A	4	Girl	AC	4	Girl

Sessions table

Group	Session	Date	Start	End	Time
1	1	28-Out-2002	14:30	14:50	0:20
1	2	5-Nov-2002	9:40	10:00	0:20
2	1	28-Out-2002	14:50	15:10	0:20
2	2	5-Nov-2002	10:00	10:20	0:20
3	1	28-Out-2002	15:15	15:40	0:25
3	2a	4-Nov-2002	14:41	15:00	0:19
3	2b	4-Nov-2002	15:00	15:20	0:20
4	1	28-Out-2002	15:40	16:00	0:20
5	1	29-Out-2002	9:30	9:50	0:20
5	2a	4-Nov-2002	14:15	14:40	0:25
5	2b	5-Nov-2002	10:40	11:00	0:20
6	1	29-Out-2002	9:50	10:10	0:20
6	2a	4-Nov-2002	15:20	15:40	0:20
6	2b	4-Nov-2002	15:40	16:00	0:20

Group	Session	Date	Start	End	Time
7	1	29-Out-2002	10:10	10:30	0:20
7	2a	4-Nov-2002	14:41	15:00	0:19
7	2b	4-Nov-2002	15:20	15:40	0:20
8	1	29-Out-2002	10:30	10:50	0:20
8	2	5-Nov-2002	9:20	9:40	0:20
9	1	29-Out-2002	10:55	11:20	0:25
9	2a	4-Nov-2002	15:40	16:00	0:20
9	2b	5-Nov-2002	10:40	11:00	0:20
10	1	29-Out-2002	11:20	11:40	0:20
10	2a	4-Nov-2002	15:00	15:20	0:20
10	2b	5-Nov-2002	9:40	10:00	0:20
11	1	29-Out-2002	11:40	12:00	0:20
11	2a	4-Nov-2002	14:15	14:40	0:25
11	2b	5-Nov-2002	11:20	11:45	0:25

Sessions overview – all groups

This teacher didn't provide reports for the first sessions, and only two reports in total. She didn't use static grouping, but rather joined the children in a random manner for each session. Therefore, my label of "groups", regarding the reports presented in Appendix VIII, is merely for the convenience of the reader, in light of the approach used by the other teachers. The approach she used for the sessions was quite different from what had been intended: she had predetermined goals for each session, to be achieved by all children. If the children were unable to reach them on their own, the teacher would "help" them, to the point of demonstrating the actions herself, if necessary.

These sessions weren't helpful for the most part. But they did provide some data: some children, which had not enough mouse control to perform some activities on their own, had sometimes understood the processes well enought to help out their colleagues; and children had a variety of guesses for how a bird would behave when its nest was copied, but the concept of taking things to its nest wasn't questioned even in such a case (except for a pair of children who

thought the bird would die if its nest was copied). And how easily a professional (albeit one with little early-childhooh education training) fell into the option of being too rigid in an environment where perhaps she wasn't comfortable.

Highlights – all groups

« A problem I found was that to start up the robot, one must give it a blue box; but the one that was on the room floor is always placed behind the robot's thoughts, and it's hard for the girls to find it to give it to the robot. »

(G, Group 1, Session 1)

« Then what will the bird do, if I copied the nests? (...) *they said it would place the objects in all the nests. »*

(G, Group 2, Session 2)

« MG said, after making the copy, that the bird would take the object only to one nest, because she only had one birdie. »

(G, Group 3, Session 2b)

« FS thought that the bird would always place the object in a single nest. »

(G, Group 6, Session 2)

« DI's answer to the question [about a bird and several nests]*: "I'll see", and he had to give an object to the bird, to see what it would do. »*

(G, Group 7, Session 1)

« B helped CA in the steps where she was doing it wrong. »

(G, Group 8, Session 2)

« AC doesn't speak much; I had to do everything on her behalf. She doesn't want to stay at the computer for long. »

(G, Group 11, Session 2b)

Senhora da Pena preschool

Computer-activities teacher profile

Name initials	MF
Age	37
Education	Bachelor in Early Childhood Education
Professional experience	She was a Preschool teacher for 5 years (1993-1998), but then interrupted her educational career and was a secretary at the Ministry of Justice for four years. In this year, she resumed her educational career as preschool computer-activities teacher at 5 preschools.
Computing skills	Her degree included a course on basic computing. Received regular training and guidance on computer use in preschool contexts, within the ICEI project.
ToonTalk training	Attended two training sessions on the use of the environment tools and basic programming.

The children

12 children: 6 girls, 6 boys; 4 three-year olds, 4 four-year olds, 4 five-year olds.

Group	Child 1 (Age, Gender)			Child 2 (Age, Gender)		
1	JO	4	Boy	F	3	Boy
2	A	4	Girl	L	4	Boy
3	JG	3	Boy	C	3	Girl
4	P	5	Boy	FI	5	Girl
5	JA	5	Girl	CA	5	Girl
6	E	4	Boy	S	3	Girl

Sessions table

Group	Session	Date	Start	End	Time
1	1	17-Out-2002	14:15	15:45	1:30
2	1	10-Out-2002	10:00	10:30	0:30
2	2	17-Out-2002	14:15	15:45	1:30
2	3	28-Nov-2002	14:45	15:30	0:45
2	4	13-Jan-2003	10:45	11:15	0:30
2	5	10-Fev-2003	14:15	14:45	0:30
3	1	18-Out-2002	10:00	10:45	0:45
3	2	21-Out-2002	10:20	11:10	0:50
3	3	28-Nov-2002	9:45	10:00	0:15
3	4	28-Nov-2002	15:30	15:45	0:15
3	5	13-Jan-2003	10:00	10:30	0:30

Group	Session	Date	Start	End	Time
4	1	21-Out-2002	9:45	10:15	0:30
4	2	10-Fev-2003	15:00	15:30	0:30
5	1	21-Out-2002	11:15	11:45	0:30
5	2	13-Jan-2003	11:30	12:00	0:30
6	1	21-Out-2002	11:45	12:00	0:15
6	2	28-Nov-2002	11:00	11:45	0:45
6	3	13-Jan-2003	14:30	15:00	0:30
6	4	10-Fev-2003	10:30	11:15	0:45
7	1	28-Nov-2002	10:05	11:00	0:55

Sessions overview – all groups

This teacher opted to let children explore the environment at will, and explained small bits of it to the children, as necessary, but her reports don't detail this support. However, one can notice that children are acquiring a high degree of control over the environment, and are able to use ToonTalk's elements for their own purposes.

But the sessions didn't progress from this. The teacher did not explore programming constructs with the children. It is almost as if she was content with the children's ease and delight in the use of the environment for their purposes, *per se*. There are hints of trying to blend the ToonTalk activities with the activities taking place in the preschool room, but this was mostly at the level of the children's make-believe playfulness in ToonTalk, rather than actual development of activities. For instance, the only time one sees reports of images entering or exiting ToonTalk in this process are the placement of the children's photographs inside ToonTalk and the selection of ToonTalk pictures for the children's coat hangers in the preschool room. But these few instances did show how powerful they can be to connect a child to the virtual environment.

These sessions did provide insights on how powerful a virtual environment can be to allow children to employ their imagination and expressive abilities, and how they can create strong emotional ties with it. But the sessions also provided the insight of how it is possible for a trained early childhood teacher to leave the environment use at that, being content with having it as a sort of virtual doll house, not taking advantage of any complex constructs.

The highlights, in the following page, provide several examples of these comments.

Highlights – all groups

*« Children still haven't explored the game **completely**; therefore it is still hard to devise a strategy for specific activity-goals. »*

(MF, Group 1, Session 2, my emphasis)

« JO said that the air pump was a gas-station pump. F would like for the image dolls to leave the truck and play with him. »

(MF, Group 1, Session 3)

---

*« With the birds, they wanted to duplicate them and the nests, and **hand numbers to the birds so they could learn how to count** – there was a mess, but they had fun. Each bird was a boy, so there would have to be many birds. »*

(MF, Group 1, Session 5, my emphasis; there were no counting programs before or after this event, nor events of birds carrying objects)

« L wanted to move the vacuum cleaner around and called it a fat fish, "I'm going to take it for a walk." »

(MF, Group 2, Session 2)

« A took out boxes, nests, and robots. She likes to see the birds getting out of the nests, and then she makes "pretend" games, as if she was playing with dolls on the floor. She talks with herself and to the "dolls" – the bird and the robot – giving them orders. Every once in a while, she rides the helicopter to go shopping. »

(MF, Group 2, Session 3)

« They placed two scales on the room floor. After they both played, L said that those were for flour and for olive oil, that his granny had one just like them. (...) *These are very small children, and yet **they have appropriated the game, and brought many of their livelihoods into the small screen, as if they were part of it in a make-believe world**. »*

(MF, Group 2, Session 4; my emphasis)

---

*« He loves blowing up houses (his father is an explosives technician in construction sites, and he did mentioning doing as his father does). He took out a robot and **worked on the thoughts** with boxes. »*

(MF, Group 4, Session 1, my emphasis; the teacher is failing to see the need for a purpose, in order for this "work on the thoughts" to be meaningful.)

« After the activity [of placing their picture on a frame]*, P wanted to build houses so he could then blow them up to open space for a road to France. »*

(MF, Group 4, Session 2; again, this interest of the child has not been explored)

« JA and CA took out a nest from the box; the bird hatched and with the magic wand they copied several nests. Then they went to the notebook and brought a flower to give to the bird, which unfolded into several birds, to place the flower in the copied nests. They did this several times, with different objects. »

(MF, Group 5, Session 4; there was no follow up activity with birds)

« E and S like to experiment with all the objects in the toolbox. The trucks are their favorite. And E is already able to place boxes, robots, and objects on the trucks, to build many houses, "for when I return from Switzerland." »

(MF, Group 5, Session 1)

Annex VIII
—
Full reports from the sessions conducted by computer-activity teachers in preschools

Mateus preschool

Group 1

Session 1

Início: 10h 45m **Fim:** 11h10m

PL (age 4) /LF (age 4)

Objectivos Prévios: Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

Esta primeira actividade consistiu emas criança, de forma livre explorarem o ambiente do Toon Talk. Pilotarem o Helicóptero, faze-lo aterrar junto às casas, entrarem com o boneco na casa e explorarem as diferentes ferramentas que se encontramao seu dispor. O LF conseguiu "pilotar" o Heli até pousar junto às casas, conseguiu também, embora com alguma dificuldade entrar na casa. Já dentro desta e, com a caixa de ferramentas ao seu dispor, o LF teve algumas dificuldades em agarrar objectos e a aspirar. Esta dificuldade deveu-se ao facto de ele não ter assimilado a "regra" de que para se agarrar algum objecto ou aspirar, tem de primeiro ver se o objecto pretendido está a mexer, como lhe foi por mim transmitido e exemplificado.

A PL sentiu algumas dificuldades em fazer descer o heli, pois confundiu por diversas vezes as teclas (descer/subir), teve ainda dificuldades em o colocar junto às casa e em concretizar a aterragem propriamente dita, pois como o heli se apresentava ao mesmo nível das casas, ela pensava que este já se encontrava em terra.

Após ter efectuado a aterragem, ela saiu com o boneco e tentou chegar à casa. Esta foi mais uma tarefa de difícil concretização, uma vez que, por diversas vezes, ela fez, sem ser sua intenção, sentar o boneco.

Quando entrou, tive de a ajudar para que ela conseguisse agarrar os objectos e experimentar trabalhar com alguns deles.

Lista de elementos de registo (ficheiros) associados:

Observações:

O LF identificou de imediato o heli e disse que as casas eram o aeroporto.

Session 2

Início: 09h 40m **Fim:** 10h06m

PL (age 4) /LF (age 4)

Objectivos Prévios: Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

Esta terceira sessão consistiu na realização de trocas de objectos nas caixas, e se possível, na programação do robô para efectuar essas mesmas trocas.

Assim, e passando à caracterização do desempenho das duas crianças que compõem este grupo, tenho a referir que a PL sentiu muitas dificuldades durante o desenrolar desta actividade. Para ela fazer o percurso desde a saída do helicóptero até à casa com o boneco foi de facto bastante complicado, pois foram várias as vezes em que fez o boneco sentar sem o querer. Já dentro da casa, mais propriamente no chão desta, a Patrícia não conseguiu, de forma satisfatória, realizar as tarefas mais simples tais como: agarrar, aspirar, trocar objectos, dado que aquando da realização de qualquer uma destas tarefas, ela nunca conseguiu obter o que pretendia.

Em Relação ao desempenho do LF, posso dizer que foi idêntico ao da Patrícia. As dificuldades sentidas foram exactamente as mesmas, não havendo por isso nada a acrescentar.

Resta-me apenas referir que não avancei para a troca de caixas e por sua vez, para a programação do robô, pelo simples facto de não considerar conveniente introduzir novas tarefas, sem que estas estejam presentes nas crianças.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

Início: 10h 15m **Fim:** 10h 37m

PL (age 4) /LF (age 4)

Objectivos Prévios: Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A quarta sessão consiste em realizar algumas tarefas como o aspirar e o colocar e trocar objectos do interior das caixas e posteriormente programar o robô para trocar ou aspirar os conteúdos das caixas. No entanto, irão ser as crianças a escolher o que querem ensinar ao robô (aspirar, copiar, trocar), uma vez que já tiveram a oportunidade de realizar este tipo de tarefas.

Assim, e passando à descrição do desenrolar da sessão realizada com o grupo nº 1, posso referir que tanto a PL como o LF, e embora tenham sentido algumas dificuldades na realização da actividade, conseguiram, com o meu auxílio, concretizar a programação do robô. A PL ensinou o robô a aspirar um camião da caixa e ainda a fazer cópias de uma caixa. Na primeira ela retirou uma caixa e um camião da mala de ferramentas e colocou o ultimo no interior da caixa. Em seguida, ela agarrou no aspirador e aspirou o camião do interior da caixa (as duas primeiras vezes que ela realizou esta tarefa, não conseguiu aspirar somente o camião, ela aspirou a caixa na totalidade da primeira vez, e da segunda aspirou o camião, mas esqueceu-se de desligar o aspirador e a caixa também foi aspirada. Após ter experimentado no chão da casa, a patrícia deu a caixa ao robô e programou-o para fazer o que ela esteve a treinar. As copias das caixas ela já as fez no pensamento do robô. A caixa utilizada para entrar no pensamento do robô foi a mesma, por sua opção.

O LF só ensinou o robô a aspirar. Ele realizou primeiro no chão da casa e só depois entrou no pensamento do robô para o ensinar. A caixa era individual e continha um camião. As dificuldades sentidas pelo LF foram as mesmas que a patrícia sentiu, ou seja, não foi a programação do robô que lhe causou problemas, mas sim a tarefa de aspirar.

Lista de elementos de registo (ficheiros) associados:

Observações:

As dificuldades sentidas por estas crianças, não se prendem tanto com a programação do robô propriamente dita, mas sim com a realização de tarefas como o aspirar, o agarrar, o copiar o objecto certo.

Session 4

Início: 09h 35m **Fim:** 10h 01m

PL (age 4) /LF (age 4)

Objectivos Prévios: Avaliar o desempenho da criança

Desenrolar e Resultados Gerais:

A PL sentiu dificuldades em fazer pousar o heli e em se deslocar para a casa sem fazer sentar o boneco no chão da rua. Sentiu ainda dificuldades em agarrar objectos, bem como em dar a caixa ao robô para entrar no seu pensamento. Já dentro do pensamento, a PL ensinou somente o robô a retirar objectos da mala de ferramentas. Não se lembro como se fazia para sair do pensamento, nem como fazer o teste para verificar se o robô aprendeu algo. Assim, todo o desenrolar da actividade foi feito com um constante auxílio dado por mim.

O LF retirou alguns objectos da mala de ferramentas (camião, caixas, letras, números e um ninho) e, limitou-se a deslocá-los de um lado para o outro. Não fez portanto, programação do robô, o que não quer dizer que não o saiba fazer, mas, e por sua própria vontade, não avançou para a programação.

Lista de elementos de registo (ficheiros) associados:

Observações:

O LF perguntou porque razão jogamos sempre o mesmo jogo.

Group 2

Session 1

Início: 10h 30m **Fim:** 10h55m

PR (age 4) / JPM (age 4)

Objectivos Prévios: Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

O JPM, embora o tivesse tentado, não conseguiu parar o heli junto das casas, tendo por isso optado por trabalhar com o boneco no exterior. Comas diversas ferramentas, ele teve algumas dificuldades em as manusear, no entanto, não me parece que sejam dificuldades de grande importância, dado que elas se prendem mais com a sua destreza manual e não com as características do jogo. Quando se levantou, após ter estado a trabalhar livremente, ele não conseguiu encontrar o heli, tendo sido eu a chamar o mesmo através do teclado.

O PR, embora tivesse tido dificuldades em aterrar juntos às casas, acabou por concretizar o seu objectivo. Andou bem com o boneco até ao momento em que quis entrar em casa. Aí teve bastantes dificuldades em levar o boneco a abrir a porta, não tendo conseguido concretizar tal tarefa. Assim, o PR, trabalhou com a sua caixa de ferramentas no exterior.

No que se refere ao manuseamento dos diversos objectos, ele sentiu muitas dificuldades em tirar partido dos mesmos, nomeadamente do aspirador, da bomba.

Lista de elementos de registo (ficheiros) associados:

Observações:

Ambas as crianças identificaram o heli como sendo um avião.

O JPM teve distraído após ter realizado a sua tarefa.

A mesma criança não gostou do jogo.

Session 2

Change in group composition due to absentees. This session matches Session 2 in group 7.

Início: 11h 24m **Fim:** 11h 49m

PR (age 4) / LE (age 5)

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOONTALK.

Desenrolar e Resultados Gerais:

O LE não teve dificuldade alguma em realizar esta actividade. Não necessitou de nenhum tipo de ajuda, eu só lhe dei algumas dicas para ele fazer este ou aquele tipo de tarefa, como o aspirar objectos do interior das caixas, copiar objectos, rebentar com as casas, etc.

O PR, sentiu mais dificuldades, nomeadamente no agarrar, aspirar e copiar, pois esqueceu-se frequentemente de que o objecto com que quer trabalhar tem de estarem movimento.

As dificuldades sentidas por esta criança prendem-se com o facto de esta não conseguir permanecer muito tempo concentrada no que está a realizar.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

Início: Fim:

PR (age 4) / JPM (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

Neste grupo faltou um elemento, o PR, por essa razão faço apenas referência ao JPM.

O JPM conseguiu fazer a troca de objectos dentro das caixas com bastante facilidade, ele retirou duas caixas da mala de ferramentas, tirou um camião e uma bomba da mesma e colocou esses objectos em cada um dos compartimentos da caixa, e digo caixa porque embora ele tivesse tirado duas ele juntou a segunda à primeira, o que origina, como sabemos, uma só caixa com dois compartimentos. Depois trocou de um lado para o outro o camião e a bomba, sem grandes dificuldades. De seguida, e após ele ter realizado esta tarefa por diversas vezes, sugeri que o JPM

ensinasse ao robô a fazer o que ele tinha estado a fazer. Então, e após ter explicado e exemplificado, que temos de entrar no pensamento do robô, que está na sua cabeça, e ensina-lo a realizar a troca de objectos nas caixas, o JPM consegui com alguma, mas pouca ajuda minha, ensinar o robô a trabalhar sozinho.

Lista de elementos de registo (ficheiros) associados:

Observações:

O JPM perguntou ao início, qual a razão pela qual se joga sempre o mesmo jogo.

Session 4

Início: 09h 40m **Fim:** 10h 2m

PR (age 4) / JPM (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O PR após ter navegado com o helicóptero, ele pousou-o e dirigiu-se para casa. Dentro da casa ele, com a minha indicação, começou por tirar duas caixas da mala de ferramentas, um camião e uma bomba. Colocou esses mesmos objectos nas caixas e esteve a trocá-los de posição. Depois, pegou nessa mesma caixa e deu-a ao robô, tendo assim entrado no pensamento deste. Aí ele ensinou o robô a trocar os objectos de sítio e a volta-los a colocar na posição correcta. Em seguida retirou uma balança da mala de ferramentas para saber quando é que o robô termina a sua tarefa. (a colocação da balança foi feita por minha indicação). Terminado o ensinamento, o PR saiu do pensamento do robô e foi verificar se o robô estava ou não a trabalhar como ele o ensinara. No fim resolveu que tinha de rebentar com as casas e foi o que ele fez, com bastante contentamento.

O JPM teve grandes dificuldades em navegar e pousar o helicóptero, principalmente em pousar junto às casas. Tive de colocar a minha mão sobre a dele, para que este conseguisse pousar junto às casas. Já dentro da casa, o JPM quis somente rebentar com as casas. Ele rebentou com uma, saiu da casa entrou no heli e foi ver o resultado, rebentou com a outra e fez o mesmo, quando só faltava uma casa para rebentar decidiu que não queria mais jogar.

Lista de elementos de registo (ficheiros) associados:

Observações:

O JPM perguntou ao início da sessão anterior o porquê de estarmos sempre com o mesmo jogo. Este aborrecimento poderá estar a influenciar o seu desempenho e o seu interesse nas sessões do Toon Talk.

Session 5

Início: 10h10m **Fim:** 10h30m

PR (age 4) / JPM (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

Nesta sessão de toon talk, o PR e o JPM quiseram trabalhar em conjunto no toon talk. Durante toda a sessão estas duas crianças estiveram a trabalhar em sintonia ajudando-se mutuamente. Teria sido bom que ambas tivessem optado por ensinar o robô a trabalhar, no entanto não o fizeram, pois optaram por fazer cópias de camiões, balanças e caixas, bem como em aspirar os mesmos. Quando se cansaram de fazer cópias e de aspirar, resolveram rebentar com as casas, o que lhes deu um grande gozo.

Apesar de não terem avançado muito, penso que esta sessão correu bem, pelo facto de ter possibilitado um desempenho totalmente feito em conjunto por estas duas crianças, durante o qual eu pude observar a perfeita sintonia existente entre estas duas crianças.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 6

Início: Fim:

PR (age 4) / JPM (age 4)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O PR, após eu ter explicado a actividade e o que queria com ela, começou logo por dar a caixa com os respectivos objectos ao robô e ensinou-o a trocar os lugares. Pegou no objecto do lado direito e colocou-o no chão e depois pegou no outro e colocou-o no espaço deixado vago pelo anterior, tendo de seguida colocado o primeiro objecto no local vazio.

Saiu do pensamento do robô e veio ver o resultado. De seguida, como o seu companheiro faltou, ele desfez a troca novamente através do robô. Ele percebeu o que estava a fazer e o porquê de o fazer.

Lista de elementos de registo (ficheiros) associados:

Observações:

O JPM faltou.

Session 7

Início: 00 h00m **Fim:** 00h00m

PR (age 4) / JPM (age 4)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O PR esteve a ensinar ao JPM o que aprendeu na sessão anterior. Não teve grandes dificuldades em demonstrar como se fazia. O JPM não entendeu muito bem todas aquelas trocas que o PR realizou, mas com a minha explicação ele conseguiu realizar a actividade sem grandes dificuldades. No entanto, e como o PR quis explicar ao JPM como se fazia, eu não lhes expliquei como copiar as imagens para o ToonTalk, tendo optado por o fazer antes do PR começar a sua explicação.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 3

Session 1

Início: 11h10m **Fim:** 11h30m

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

A AR sentiu bastantes dificuldades em realizar todas as tarefas do jogo, desde o pilotar até ao agarrar dos objectos. Esta criança, por diversas vezes trocou os comandos de pilotagem do heli, o que lhe dificultou a sua aterragem, não conseguiu passear com o boneco, pois estava constantemente a sentá-lo sem querer. Após conseguir entrar na casa escolhida (a que estava à sua frente), ela brincou com o aspirador, com a bomba, tirou objectos da caixa de ferramentas, mas sempre com bastantes dificuldades, tendo tido, durante toda a sessão, uma grande ajuda minha.

No que diz respeito ao RJ, as anotações por mim tomadas são as mesmas que tirei para a criança anterior. As dificuldades sentidas por este menino, foram as mesmas sentidas pela AR.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Início: 09h40m **Fim:** 10h05m

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOONTALK.

Desenrolar e Resultados Gerais:

Ambas as crianças mostraram que ainda se recordavam do que se tinha estado a fazer na sessão anterior. Sentiram as mesmas dificuldades que haviam já sentido na semana passada, no entanto, essas não foram notadas com tanta intensidade, nomeadamente no que se refere ao RJ, fruto da atenção com que esta criança esteve na realização da sessão anterior.

Lista de elementos de registo (ficheiros) associados:

Observações:

O RJ esteve sempre interessado no desenrolar do "trabalho" da AR, tendo ajudado esta aquando da construção de casas. A AR queria construir casas mas, esqueceu-se que tinha de colocar uma caixa no camião. O RJ, desde logo, alertou a sua colega para esta falha, e explicou a solução.

Session 3

Início: 10h10m **Fim:** 10h35m

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A AR mal se sentou começou logo a jogar, fazer casas e mais casas sem ajuda. Ela começou a colocar nos camiões caixas e robôs e eu perguntei se ela sabia o que estava a fazer, tendo ela respondido que estava a construir casas. Após ter construído varias casas, sugeri que ela trocasse objectos dentro das caixas, tarefa que ela realizou sem grande dificuldade. Em seguida expliquei-lhe que o robô também podia fazer o mesmo e que para isso bastava ensina-lo. Então ela perguntou como se fazia e eu expliquei. Após esta minha explicação, a AR deitou "mãos ao trabalho" e programou o robô sem que para isso eu tivesse que a estar constantemente a ajudar.

O RJ só quis fazer casas e rebentar com as mesmas, não consegui levá-lo a realizar as tarefas que a AR realizou. No entanto, e de acordo com o desempenho do RJ no tipo de tarefa que realizou, penso que não irá ter grandes dificuldades em programar o robô.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 4

Início: 10h06m **Fim:** 10h35m

AR (age 4)/ RJ (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A AR tentou, assim que aterrou o helicóptero, programar o robô por três vezes consecutivas. Eu esperei para ver se ela dava conta do porquê de não conseguir realizar tal tarefa, no entanto tive de intervir para lhe explicar que não se consegue programar o robô no chão da rua. Assim, ela entrou na casa com alguma dificuldade, pois não conseguia acertar com a porta, e de imediato retirou um robô da caixa de ferramentas, duas caixas vazias que encheu uma com o nº1 e a outra com a letra A, juntou as duas e o rato fez a colagem das mesmas. Depois ela deu ao robô as respectivas caixas, ou melhor a caixa com dois compartimentos e programou o robô a aspirar o conteúdo da caixa. Saiu e veio testar o robô. Após verificar o seu funcionamento, ela programou outro robô a aspirar de uma caixa um camião. Esta segunda programação foi inteiramente feita por ela sem o meu auxílio.

Quanto ao RJ, ele programou o robô a retirar da mala uma caixa vazia, uma letra A e a colocar essa letra na caixa. Depois levou o robô a aspirar a letra.

Esta criança não quis estar muito tempo no jogo, pelo que se realizou esta tarefa.

Lista de elementos de registo (ficheiros) associados:

Observações:

A AR quando teve de dar a caixa ao robô para entrar no seu pensamento, ela utilizou a caixa inicial, ao contrário das outras crianças, que vão sempre buscar uma nova caixa à mala de ferramentas.

Session 5

Início: 09h35m **Fim:** 10h05m

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A AR ensinou o robô a copiar caixas com o nº 1 e a aspirar as mesmas. Para entrar no pensamento, a AR deu ao robô uma caixa com a letra A. Em seguida entrou no pensamento e retirou uma caixa vazia e um caixa de nº da mala de ferramentas, colocou a última dentro da primeira e agarrou na varinha mágica e fez 3 cópias. Em seguida pegou no aspirador e aspirou-as. Ao aspirar, a AR esqueceu-se de parar o aspirador, no entanto conseguiu aspirar só o pretendido, pois não existia mais nada, no seu caminho, que pudesse ser aspirado acidentalmente.

Após ter testado o robô, dando-lhe uma caixa igual à que o robô tinha no seu pensamento, caixa essa que voltou a retirar da mala de ferramentas, a AR construiu casas, retirando um camião, uma caixa e um robô da mala de ferramentas.

O RJ, esteve a ensinar um robô a construir casas, e outro a destruir uma casa. Para isso, ele deu ao robô uma caixa com uma balança e já dentro do pensamento, retirou um camião, uma caixa e um robô da mala de ferramentas e programou um robô a construir casas. Veio testar esse mesmo robô, teste esse que não conseguiu realizar à primeira porque deu uma caixa com um robô e não a caixa que estava no pensamento do robô. No entanto, após eu ter explicado e lembrado, ele conseguiu fazer o teste.

No segundo robô, o RJ deu uma caixa com um camião, entrou no pensamento e retirou uma bomba da mala de ferramentas e rebentou com o pensamento. Realizou o teste sem cometer o erro anterior.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 6

Início: 10h07m **Fim:** 10h31m

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Avaliar o desempenho da criança.

Desenrolar e Resultados Gerais:

A AR começou por construir várias casas, depois retirou duas caixas da mala de ferramentas e colou-as. No buraco do lado esquerdo colocou a letra A e no buraco do lado direito o nº 1. Agarrou essa mesma caixa e deu-a ao robô. Entrou no pensamento, retirou um ninho da mala de ferramentas, pousou-o, retirou mais alguns objectos e, desses objectos deu alguns ao pássaro. Depois quis sair do pensamento, mas não se recordava como fazer. Esperei um pouco para ver se ela se lembrava, mas conseguiu sair com o auxílio do RJ que se aprontou a ajudar a sua colega. Quando veio testar o robô, também não se conseguiu lembra como se fazia. Para conseguir realizar essa tarefa eu tive de lhe explicar os procedimentos que devia tomar. No entanto, esperei que ela se conseguisse lembrar e/ou que o RJ a apoiasse. Para acabar rebentou com algumas casas.

O RJ começou por construir casas, em seguida retirou uma caixa e o nº 1 da mala de ferramentas, colocou o último dentro da primeira e agarrou a caixa e colocou-a nas mãos do robô (teve dificuldade em agarrar na caixa por completo, isto é, ao tentar agarrar a caixa ele agarrou o nº.). Já no pensamento, ensinou o robô a retirar alguns objectos

da mala de ferramentas. Quando saiu do pensamento o RJ não testou o robô, tive de ser eu a perguntar-lhe se o robô trabalhava bem ou não. Só ai é que ele se lembrou de testar o robô. No entanto não conseguiu fazer o teste sem a minha ajuda, pois não se recordava como o fazer. Tive de lhe explicar novamente como proceder para testar o robô. Depois saiu da casa, entrou no helí e terminou a sua prestação.

Lista de elementos de registo (ficheiros) associados:

Observações:

Ambas as crianças disseram que não queriam trabalhar no Toon Talk, aquando do início da actividade.

Session 7

Início: Fim:

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O RJ não quis fazer a actividade proposta. Embora eu tivesse tentado encaminha-lo para o que tinha planeado, não consegui persuadi-lo da sua vontade de rebentar com as casas para depois ver os trabalhadores a reconstrui-las.

Lista de elementos de registo (ficheiros) associados:

Observações:

A AR faltou

Session 8

Início: Fim:

AR (age 4) / RJ (age 4)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O RJ conseguiu realizar a troca no chão da casa, mas não conseguiu ensinar o robô, embora eu tenha exemplificado várias vezes. Ele entrou no pensamento do robô com a caixa mas depois ele fez a troca e muitas mais coisa, como o copiar, o aspirar, enfim fez de tudo sem se aperceber de que tudo o que fazia o robô também iria fazer. O resultado, como é evidente foi uma tremenda "salgalhada" sem princípio nem meio nem fim.

Lista de elementos de registo (ficheiros) associados:

Observações:

A AR faltou.

Group 4

Session 1

Início: 10h20m **Fim:** 10h40m

L (age 4) / MT (age 4)

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

Neste grupo de duas crianças, a que se destacou mais, pela positiva, foi a MT, pois ela conseguiu, desde logo, realizar com alguma destreza, o percurso com o heli e também todas as outras tarefas que se lhe apresentaram pela frente, como foi o caso de entrar em casa, fazer sentar o boneco, tirar objectos da caixa de ferramentas, e trabalhar com alguns dos instrumentos tais como, o aspirador e a varinha mágica.

O L sentiu bastantes dificuldades desde o início, pois trocou por diversas vezes as teclas para fazer descer/subir o heli, não conseguiu realizar o percurso (do heli até ao interior da casa) com o boneco, uma vez que estava constantemente a clicar e consequentemente o boneco sentava-se, no interior da casa e já com a caixa de ferramentas ao seu dispor não conseguiu agarrar nos objectos por si escolhidos (nem por uma vez o L agarrou o objecto escolhido) e tira-los para fora da caixa, aspirou sempre o que queria e o que não queria, o que lhe provocou algum transtorno.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Início: 10h05m **Fim:** 10h26m

L (age 4) / MT (age 4)

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOONTALK.

Desenrolar e Resultados Gerais:

A MT conseguiu lembrar-se do que aprendeu na 1ª sessão, o que originou a que neste dia, ela tenha trabalhado facilmente com as ferramentas do TOON TALK (aspirador, varinha mágica e bomba de encher). Esta criança fez descer o heli, entrou em casa, e fez sentar o boneco sem praticamente ter pedido ajuda. Para trabalhar com as diversas ferramentas a MT precisou de alguma ajuda, nomeadamente para encher alguns objectos tirados da caixa de ferramentas, uma vez que, esta tarefa não tinha ainda sido explorada pela criança. Em tudo o resto, a MT esteve sempre com alguma à vontade e compenetrada no que fazia.

O L teve muitas dificuldades em realizar esta sessão. Não se lembrou do que tinha aprendido, e mesmo com a minha ajuda para o fazer recordar, ele não conseguiu trabalhar com as diferentes funcionalidades do jogo. Continua a trocar o botão que leva o heli no sentido descendente com o que o leva no sentido ascendente, dificultando assim a sua pilotagem. Para tirar um objecto da caixa de ferramentas, recolhe dois ou três diferentes dos escolhidos e só depois, com bastante insistência minha para que estivesse com atenção ao movimento do objecto, é que ele conseguiu tirar o escolhido. No que respeita a aspirar e a fazer copias as dificuldades prenderam-se com o mesmo facto, ou seja, a criança não dá atenção ao movimento do objecto escolhido.

Lista de elementos de registo (ficheiros) associados:

Observações:

A MT ajudou o L durante o tempo em que este esteve a realizar a actividade, contudo esta sessão foi interrompida pelo facto de a MT se ter fartado de estar na sala e ter dito que queria vir embora. O L perante a atitude da Teresa foi de "arrasto" com ela.

Session 3

Início: 10h40m **Fim:** 10h55m

L (age 4) / MT (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A terceira sessão realizada por este grupo, não se desenrolou como estava à espera pois, a MT só quis encher e esvaziar objectos com a bomba de encher, no entanto conseguiu realizar esta tarefa com bastante facilidade.

O L esteve constantemente a construir e a destruir casas, e embora estivesse a faze-lo com muita satisfação, ele não conseguiu permanecer muito tempo no computador.

Lista de elementos de registo (ficheiros) associados:

Observações:

A MT quando tentou encher os objectos pela primeira vez, utilizou a bomba de destruição, mas após eu ter explicado qual a diferença entre uma e outra, ela não trocou mais as bombas .

Session 4

Início: 10h40m **Fim:** 110h10m

L (age 4) / MT (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

Mais uma vez a MT não fez programação do robô. Ela esteve somente a fazer cópias de camiões e de balanças, embora o tivesse feito por bastante tempo. Ela, apesar do meu incentivo, não quis programar o robô. Eu programei-o para fazer cópias, pois era o que ela estava a fazer, e nem assim ela quis tentar. Na próxima sessão terei de a "forçar" a entrar na programação.

O L fez a mesma coisa que a MT. Não sei se foi por ter visto a sua colega a fazer a actividade, ou se também não o quis por vontade própria. No entanto penso que foi por influência. Na próxima sessão vou começar com o L a fazer a programação, para tentar desta forma, influenciar a MT, e evitar que ele não seja influenciado por ela.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 5

Início: 00h00m **Fim:** 00h0m

L (age 4) / AC (age 4) – this matches session 6b of group 5.

Objectivos Prévios:

Avaliar o desempenho da criança.

Desenrolar e Resultados Gerais:

O L começou por retirar objectos ao acaso da mala de ferramentas e a desloca-los de um lado para o outro. Depois, com a indicação da AC, ele deu uma caixa ao robô e entrou no pensamento. Aí ensinou o robô a retirar uma caixa da mala de ferramentas, um camião, que colocou dentro da caixa e um ninho. Depois saiu do pensamento e testou o robô. (toda a tarefa foi realizada pelo L com a indicação da AC).

A AC esteve somente a trabalhar no chão (da rua), retirando para o mesmo diversos objectos, tendo-se limitado a transportá-los de um lado para o outro. Este facto supeendeu-me bastante, pois ela esteve constantemente a ajudar o L.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 6

Início: 00h00m **Fim:** 00h0m

L (age 4) / AC (age 4) – this matches session 7b of group 5.

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O L fez a troca dos sinais no chão da casa e posteriormente ensinou o robô a realizar a mesma tarefa. No chão da casa ele retirou os dois sinais da caixa e colocou-os no chão. Depois pegou no que estava no lado esquerdo e colocou-o no lado direito e o que estava no lado direito colocou-o no lado esquerdo, tendo depois desfeito essa troca para dar a caixa ao robô. Dentro do pensamento do robô ele realizou a tarefa com uma pequena diferença. Retirou-o o sinal do lado esquerdo e colocou-o no chão e depois pegou no do lado direito e colocou-o logo no lado esquerdo da caixa. Em seguida pegou no outro sinal e colocou-o no respectivo lugar. Entretanto ele ia desfazer o que havia feito, pois foi o que fez no chão da casa. Então tive de lhe explicar porque razão ele tinha desfeito a troca no chão da casa. Ele compreendeu a razão, e o objectivo da actividade.

A AC colocou objectos nas caixas, retirou esses objectos para colocar outros, aspirou, copiou, e para finalizar rebentou com a casa.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 7

Início: 00h00m **Fim:** 00h0m

L (age 4) / AC (age 4) – this matches session 8b of group 5.

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O L realizou a tarefa de imediato no pensamento do robô sem grandes dificuldades. No entanto, ao contrário de algumas crianças, este não conseguiu copiar sozinho os sinais e o camião.

A AC fez a actividade com o meu auxílio. Ela fez as trocas no chão da casa, ensinou o robô a destrocar o que o L tinha trocado, mas sempre com a minha ajuda, pois não se lembrava sequer de como se fazia para entrar no pensamento do robô.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 5

Session 1

Início: 09h40m **Fim:** 10h13m

AA (age 5) / AC (age 4)

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

Ambas as crianças conseguiram identificar o heli, mas não identificaram as casas como tal, quando vistas de cima.

A AA teve dificuldades em se aperceber se o heli tinha ou não pousado, pois para ela este já estaria no chão desde o momento em que se encontra ao mesmo nível das casas e, embora eu tenha feito referência e alertado para o facto de o heli só se encontrar pousado quando não se encontrasse com as pás em movimento e com o motor silenciado, ela continuou a ter dificuldades em se aperceber se de facto o heli se encontrava ou não parado. Por esta razão eu insisti na navegação para que a AA se apercebesse de tal facto.

No que diz respeito ao agarrar de objectos, a AA não teve grandes dificuldades em entender que para realizar tal tarefa, necessitava de ver o objecto pretendido a abanar. No entanto, e como gostou de retirar objectos da caixa e de os colocar no chão, a certa altura eram tantos em tão pouco espaço, que a levou a sentir bastantes dificuldades para os agarrar, como é evidente.

Em relação ao outro membro do grupo, a AC, posso referir que esta sentiu muitas dificuldades em todo o tipo de tarefa que fez, na navegação, no deslocamento do boneco, no agarrar, no aspirar.... Penso no entanto que as dificuldades se devem só à não compreensão das "regras" do jogo, mas também ao seu menor desenvolvimento, ao nível da motricidade fina, pois tem bastantes dificuldades em trabalhar com o rato.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Início: 10h30m **Fim:** 10h53m

AA (age 5) / AC (age 4)

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOONTALK.

Desenrolar e Resultados Gerais:

A AA já trabalhou de forma satisfatória no jogo. Ela conseguiu navegar sem dificuldade com o heli, entrar na casa e brincar com algumas ferramentas como o aspirador e a varinha mágica.

Sem querer, a AA entrou no pensamento do robô, o que a assustou um bocado, pois ficou um pouco confusa por não entender onde se encontrava. Eu expliquei o que se estava a passar e demonstrei como programar o robô para dar uma caixa ao pássaro. Ela gostou do que aprendeu a fazer, não tendo tido grande dificuldade em realizar esta tarefa após a minha explicação.

A AC, continua a ter bastantes dificuldades na realização de qualquer tipo de tarefa. Não consegue aterrar com o heli, não consegue agarrar os objectos pretendidos, não interiorizou que para aspirar tem de permanecer com o aspirador na mão, embora eu tivesse explicado e exemplificado, por diversas vezes, como fazer para realizar tal tarefa.

Dadas as dificuldades sentidas por esta criança, e apesar de ela ter visto a exemplificação que fiz com a AA para a programação do robô, eu achei por bem, não tentar avançar com a AC, por razões obvias.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

Início: 09h45m **Fim:** 10h28m

AA (age 5) / AC (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A AA começou por tirar alguns objectos da mala de ferramentas, letras, números e caixas.

Ela colocou uma caixa com o nº 2 e tentou colocar uma letra nessa mesma caixa. Não deu resultado. Eu expliquei o porquê de não dar para fazer isso e ela em seguida tirou uma caixa vazia e colocou a letra A e depois colocou mais duas vezes a mesma letra. Em seguida copiou, com a varinha mágica essa caixa e aspirou tudo. Em seguida, expliquei como se poderia fazer para ensinar o robô a trabalhar sozinho. Ela gostou da ideia, e após a minha explicação ela começou logo a trabalhar. Então deu uma caixa vazia ao robô, entrou no pensamento e ensinou o robô a tirar uma caixa da mala de ferramentas, retirar um camião e coloca-lo lá dentro, depois tirou outra caixa e colocou dentro desta uma balança e trocou os conteúdos das caixas. Saiu do pensamento e veio ver o robô a trabalhar. Quando deu a caixa vazia ao robô este começou a trabalhar, no entanto apareceu uma mensagem emitida pelo marciano de que o robô havia perdido o rasto a muita coisa e que ele não esperava trabalhar com tanta coisa. No entanto, e apesar do erro, a AA conseguiu ver o robô a trabalhar.

A AC começou também por retirar objectos da mala de ferramentas e a coloca-los dentro de caixas, depois aspirou tudo o que havia retirado e colocado no chão da casa e pegou numa caixa vazia e deu-a ao robô. Entrou no pensamento deste e programou-o não para fazer trocas, porque ela não o quis, mas sim para o ensinar a aspirar. Então ensinou o robô a aspirar uma letra, um nº e um camião. Saiu do pensamento e veio ver o robô a trabalhar.

Por fim a AA decidiu aspirar a casa toda e sair.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 4

Início: 10h 40m **Fim:** 11h00m

AA (age 5) / AC (age 4)

Objectivos Prévios: Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

A AA quis ensinar o robô a fazer casas. Eu expliquei-lhe que era muito parecido ao que ela tinha feito na sessão anterior, só mudava dentro do pensamento do robô, no qual ela tinha de retirar um robô, uma caixa e um camião da mala de ferramentas, e em seguida colocar a caixa vazia no camião e agarrar o robô e introduzi-lo na caixa. A explicação dada por mim foi somente oral, ou seja, eu não exemplifiquei.

A AA conseguiu realizar esta tarefa com alguma facilidade, embora demorasse algum tempo para a concluir, uma vez que ela parava para pensar no que eu tinha dito. Como se pode adivinhar, a cidade ficou repleta de casas.

Por fim, quis rebentar com algumas (muitas) casas, tendo parado somente quando se cansou.

Lista de elementos de registo (ficheiros) associados:

Observações:

A AC faltou.

Session 5

Início: 14h 10m **Fim:** 14h36m

MA (age 5) / AA (age 5) – this matches session 5 in group 6.

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O MA a primeira coisa que fez foi rebentar com a casa (já não se lembrava como se fazia). Posteriormente entrou noutra casa e foi ensinar o robô. Ele já não sabia qual era o robô. Eu disse-lhe qual era o robô e ele deu uma caixa vazia e entrou no pensamento. No pensamento, o MA confundiu a mão que o robô mexe com a mão que lhe permite agarrar os objectos. Após ter entendido qual a mão que agarra (através da minha explicação), tirou duas caixas vazias da mala de ferramentas e colocou em cada uma delas dois camiões, tendo de seguida trocado os respectivos lugares. Saiu do pensamento do robô, verificou se o robô trabalhava ou não, e em seguida rebentou com a casa.

O MA realizou esta actividade praticamente sem o meu auxílio.

A AA deu ao robô uma caixa com a letra A e já no pensamento ela agarrou na letra e escreveu o seu nome (com a minha ajuda). Depois retirou três caixas da mala de ferramentas e colocou o seu nome, noutra colocou o nº um e na última a letra A. Posteriormente trocou o nome com o nº e veio verificar o resultado. No fim rebentou com a casa.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 6a

Início: 09h40m **Fim:** 10h00m

MA (age 5) / AA (age 5)

Objectivos Prévios:

Avaliar o desempenho da criança.

Desenrolar e Resultados Gerais:

A AA mal entrou na casa quis rebentar com a mesma. Como já vem sendo hábito este tipo de acção, eu perguntei-lhe se ela ainda se lembrava como se ensina o robô a trabalhar e porque não ensina-lo a construir casas. Ela aceitou o meu desafio e deu ao robô uma caixa com um camião (eu perguntei o porquê de ser um camião e não outra coisa qualquer para verificar se ela já estava a pensar que para se construir casas precisamos de um camião, no entanto foi uma escolha ao acaso ou seja não tinha essa intenção). Já dentro do pensamento, ela retirou um camião, uma caixa (que colocou prontamente no camião) e um robô que colocou dentro da caixa transportada pelo camião. Verificou se o robô construiu alguma casa, facto de que se apercebeu de imediato, pois estavam na mesma 3 casas, quando só deveriam existir duas pois tinha já rebentado com uma. Para finalizar, e como não podia deixar de ser, a AA rebentou com todas as casas que existiam.

O MA seguiu exactamente os mesmos passos da AA, rebentou com uma casa, foi programar o robô para construir casa, a única diferença foi ter dado ao robô uma caixa com uma letra para entrar no seu pensamento. No

entanto quando veio verificar se o robô trabalhava ou não, o MA não consegui lembrar-se de como se fazia para testar o mesmo. Tive de lhe dizer como se fazia, ou melhor, lembrar, pois ele já sabia faze-lo.

No fim o MA não rebentou com a casa, ele preferiu pilotar o heli por algum tempo.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 6b

Início: 00h00m **Fim:** 00h0m

L (age 4) / AC (age 4) – this matches session 5 of group 4.

Session 7a

Início: 00h00m **Fim:** 0h00m – this matches session 7 of group 6.

MA (age 5) / AA (age 5)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

Ambas as crianças realizaram a tarefa "manualmente", numa primeira fase, e depois ensinaram o robô a fazer a mesma correspondência.

A primeira criança a trabalhar foi o MA. Este teve de ensinar o robô a trocar o objecto do compartimento da direita para o compartimento da esquerda, de modo a colocar o sinal dos peões mais perto do boneco que está na folha. Conseguiu, embora com alguma dificuldade, pois não se recordava como ensinar o robô.

A AA não conseguiu realizar a tarefa de forma autónoma. Fui eu quem praticamente fez a actividade, pois embora ela tivesse estado atenta, não conseguiu lembrar-se da maior parte das acções para que pudesse realizar a dita tarefa.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 7b

Início: 00h00m **Fim:** 00h0m

L (age 4) / AC (age 4) – this matches session 6 of group 4.

Session 8a

Início: 00h00m **Fim:** 0h00m – this matches session 8 of group 6.

MA (age 5) / AA (age 5)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

Ambas as crianças conseguiram realizar a tarefa, embora com alguma dificuldade, nomeadamente a AA. No entanto esta teve uma prestação bastante mais positiva do que na sessão anterior. A actividade foi idêntica a anterior, mas desta vez, foram as crianças a copiar os bonecos (sinais; camião) para o ToonTalk.

A ordem da realização da actividade foi a mesma da sessão anterior, para que assim, a AA, que teve muitas dificuldades da última vez, observasse com atenção para que a sua prestação melhorasse.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 8b

Início: 00h00m **Fim:** 00h0m

L (age 4) / AC (age 4) – this matches session 7 of group 4.

Group 6

Session 1

Início: 09h55m **Fim:** 10h24m

MA (age 5) / M (4 anos)

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

Ambas as crianças identificaram o heli e as casas.

O MA teve algumas dificuldades em fazer deslocar o heli, de forma a que este aterrasse perto das casas. Após eu lhe explicar como o fazer, ele seguiu as minhas instruções e conseguiu começar a pousar o heli junto às casas. No entanto, o MA voltou a sentir algumas dificuldades em aterrar, uma vez que para ele bastava que o heli se encontrasse ao mesmo nível das casas. Após o ter deixado tentar, por diversas vezes, chegar à conclusão de que o heli para aterrar tem de ter as pás paradas e não se pode ouvir o barulho do seu motor trabalhar, e dado que não o conseguiu fazer por si, expliquei e exemplifiquei a forma de o fazer, a partir daí, o MA conseguiu por duas vezes realizar a aterragem.

Dentro da casa não sentiu grandes dificuldades em agarrar objectos, aspirar e em fazer inúmeras cópias de camiões. Após ter posto a casa numa confusão total, cheia de camiões e de outros objectos espalhados pelo chão, o MA sentiu grandes dificuldades em agarrar o que quer que fosse.

O M teve menos dificuldades em descer junto às casas e em aterrar com o heli, fruto talvez da atenção com que estava aquando da minha explicação.

Sentiu bastantes dificuldades para entrar em casa, foi constante o sentar e o levantar do boneco, durante o percurso compreendido entre o heli e a casa, a aspirar e a fazer cópias, pelo facto de trocar as teclas destas funções.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Início: 10h55m **Fim:** 11h22m

MA (age 5) / M (age 4)

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOONTALK.

Desenrolar e Resultados Gerais:

O MA sentiu dificuldades em agarrar objectos, não conseguiu interiorizar que para tal feito, tem de ter o objecto escolhido a mexer, teve também alguns problemas ao aspirar, foram várias as tentativas para aspirar, no entanto ele fazia-o sempre com o aspirador no chão, mesmo após eu ter explicado como se tem de proceder. Para além destas dificuldades, o MA teve os mesmos problemas da sessão anterior, embora de forma mais aligeirada.

O M surpreendeu-me pela positiva. Foi só lembrar como se faziam as diversas tarefas, para que ele começasse a trabalhar sozinho sem ligar a ninguém. Conseguiu aspirar objectos retirados da caixa de ferramentas, fazer cópias, e explodir com as casas, o que para ele foi deveras divertido.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

Início: 10h30m **Fim:** 10h57m

MA (age 5) / M (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O MA, após ter entrado na casa só se interessou em construir e destruir casas, tendo somente interrompido esta "destruição maciça" para, numa das casas, retirar alguns objectos e aspirá-los. Em seguida rebentou com a casa. Não quis fazer mais nada, embora eu tivesse tentado levá-lo a fazer algo de diferente, não o consegui, pois não quero obrigá-lo a fazer algo que no momento não lhe apetece, pois se o fizesse poderia estar a por em risco todo o seu desempenho e interesse pelo jogo no futuro.

O M, já não se lembrava de quase nada do que havia aprendido na sessão anterior. Como tal, e dado que mesmo após eu o ter lembrado de como se faziam certas e determinadas tarefas, o M continuou a sentir algumas dificuldades na sua realização. Assim sendo optei por deixar o M a aspirar, copiar, retirar objectos da mala de ferramentas, para que deste modo, continue o seu processo de habituamento ao ambiente do Toon Talk.

Lista de elementos de registo (ficheiros) associados:

Observações:

Foi muito estranho para mim o desempenho do M, não por ele se ter esquecido de como realizar esta ou aquela tarefa, mas sim a dificuldade sentida após o ter lembrado de como se fazia. Espero porém que seja fruto de um mau dia, pois na sessão anterior o M surpreendeu-me pela positiva.

Session 4

Início: 11h15m **Fim:** 11h35m

MA (age 5) / M (age 4)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O MA, retirou uma caixa da mala de ferramentas, uma letra e colocou esta última na caixa. Pegou na caixa e deu-a ao robô para entrar no pensamento. Já lá dentro, ele ensinou o robô a aspirar objectos do interior das caixas. Para tal, retirou um camião e uma caixa, introduziu o primeiro dentro desta e aspirou-o. Veio ver o robô a trabalhar, mas esqueceu-se de dar uma caixa idêntica à do pensamento do robô. Eu alertei o Miguel para este facto e ele conseguiu ver o trabalho do robô.

Depois, ensinou o robô a aspirar uma bomba da caixa. Tanto na primeira como na segunda programação, tive de auxiliar a criança, pois ele sentiu algumas dificuldades, nomeadamente, no agarrar nos objectos, entrar no pensamento e agarrar objectos com o robô. No entanto, essa minha ajuda foi menor na segunda programação.

O M, e embora eu lhe tenha explicado e exemplificado como se programa o robô, não se mostrou interessado em realizar tal tarefa. Fez somente algumas cópias de objectos e rebentou com muitas casas.

Fiz uma programação do robô com ele, após este ter estado a brincar livremente no jogo, para ver se o conseguia levar a interessar-se mas não consegui, pois logo que acabamos de programar ele quis ir embora.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 5

Início: 14h 10m **Fim:** 14h36m

MA (age 5) / AA (age 5) – this matches session 5 of group 5.

Session 6

Session 7

Início: 00h00m **Fim:** 0h00m – this matches session 7a of group 5.

MA (age 5) / AA (age 5)

Session 8

Início: 00h00m **Fim:** 0h00m – this matches session 8a of group 5.

MA (age 5) / AA (age 5)

Group 7

Session 1

Início: 11h25m **Fim:** 11h55m

IX (age 4) / LE (age 5)

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

Logo de início, sem eu dizer nada, ambas as crianças identificaram o heli e as casas.

O LE conseguiu com alguma facilidade navegar com o heli e aterrar junto às casas. No interior das mesmas ele tirou facilmente os objectos pretendidos e colocou-os no chão da casa onde posteriormente os aspirou e copiou.

Ele colocou dentro de uma caixa a letra A e em seguida colocou mais letras A em cima da primeira. Aí eu sugeri que colocasse o n.º 1 em cima da letra e de imediato ele notou na alteração que surgiu, ou seja a letra A passou a letra B.

Notei que esta criança tem alguma facilidade em trabalhar com o programa.

A IX teve algumas dificuldades em fazer subir e descer o heli, devido ao facto de se enganar constantemente nas teclas do rato. Em relação ao interior da casa, a I teve alguns problemas em interiorizar as "regras" para recolher, aspirar e copiar objectos. Para sair de casa ela fez várias tentativas, uma vez que não conseguia dar com a porta.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Change in group composition due to absentees. This session matches Session 2 in group 2.

Início: 11h 24m **Fim:** 11h 49m

PR (age 4)/LE (age 5)

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOONTALK.

Desenrolar e Resultados Gerais:

O LE não teve dificuldade alguma em realizar esta actividade. Não necessitou de nenhum tipo de ajuda, eu só lhe dei algumas dicas para ele fazer este ou aquele tipo de tarefa, como o aspirar objectos do interior das caixas, copiar objectos, rebentar com as casas, etc.

O PR, sentiu mais dificuldades, nomeadamente no agarrar, aspirar e copiar, pois esqueceu-se frequentemente de que o objecto com que quer trabalhar tem de estarem movimento.

As dificuldades sentidas por esta criança prendem-se com o facto de esta não conseguir permanecer muito tempo concentrada no que está a realizar.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

No records.

Session 4

No records.

Session 5

Início: 15h11m **Fim:** 15h39m

IX (age 4) / LE (age 5)

Objectivos Prévios: Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O LE ensinou o robô a construir casas. Ele deu uma caixa com um ninho ao robô, entrou no pensamento e agarrou numa caixa, depois agarrou num camião, pousou-o colocou nele a caixa, agarrou um robô e colocou-o na caixa transportada pelo camião. Em seguida veio verificar se o robô aprendeu ou não a construir casas. Após ter verificado o robô, o LE quis ensinar outro robô a destruir casas. Então, deu uma caixa com uma bomba ao robô e entrou no pensamento, agarrou numa bomba e fez explodir a mesma. Saiu do pensamento do robô, deu a mesma caixa ao robô que tinha que tinha usado para entrar no seu pensamento, e foi verificar se o robô rebentou com alguma casa. Como só rebentou com uma, ele fez questão de ir rebentar com algumas outras casas.

Lista de elementos de registo (ficheiros) associados:

Observações:

A IX faltou.

Session 6

Início: 15h11m **Fim:** 15h39m

FG (age 5) / LE (age 5) – this matches session session 7 of group 8.

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O LE realizou a tarefa sem grandes dificuldades. Ele fez as trocas no chão da casa, e depois deu as caixas (na posição inicial) ao robô, entrou no seu pensamento e ensinou-o a realizar a troca de forma a que o sinal do lado esquerdo fosse colocado no lado direito de forma a que este fica-se mais perto da imagem colocada na folha. O objectivo da actividade foi bem compreendido por esta criança.

O FG não quis realizar a tarefa no chão da casa, tendo por isso tido mais dificuldade em a realizar logo no pensamento do robô. No entanto com alguma ajuda minha, ajuda essa verbal, ele conseguiu destrocar o que o LE havia feito. O objectivo da actividade foi também por ele percebido. Se a actividade tivesse sido realizada primeiramente por esta criança, certamente teria tido muitas mais dificuldades em a concretizar, por essa razão o coloquei a fazer após o LE.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 7

Início: Fim:

FG (age 5) / LE (age 5) – this matches session 8 of group 8.

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O LE foi novamente ao primeiro a "trabalhar". Nesta sessão expliquei como se fazia para se copiar os bonecos para o ToonTalk. Embora não conseguisse fazê-lo de imediato, o LE à terceira tentativa copiou os desenhos e foi ele que fez o cenário para a actividade. Após estar tudo no lugar ele fez a troca dos objectos no chão da casa e posteriormente deu as caixas na posição inicial (porque eu o alertei para tal facto) e ensinou o robô. A actividade foi realizada com sucesso, apesar de o facto de ter de copiar os desenhos o ter confundido um pouco no início.

O FG teve mais dificuldade em concretizar a actividade, facto esse que está certamente relacionado com a falta de atenção durante a prestação do LE e também a falta de vontade de a realizar, pois a sua vontade era rebentar com as casas.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 8

Session 1

Início: 11h00m **Fim:** 11h18m

FG (age 5) / JPF (age 3) – this matches session 1 of group 10.

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk).

Desenrolar e Resultados Gerais:

Em relação à identificação do heli e das casas, ambas as crianças disseram que era um helicóptero, no entanto, no que respeita às casas o JPF disse que eram malas e o FG disse que eram três tijolos. Quando se aproximaram delas deram conta que de facto se tinham enganado.

O FG navegou tanto com o helicóptero como com o boneco com alguma facilidade. Dentro da casa ele brincou com a caixa de ferramentas, da qual tirou algumas caixas, camiões e balanças, para posteriormente aspirar. Esta criança não teve curiosidade em trabalhar com as outras ferramentas, apesar de eu ter tentado incentivar para que o fizesse. Por último, resta-me referir que o FG esteve pouco tempo no computador, por sua vontade.

O JPF teve bastantes dificuldades realizar as diversas tarefas, nomeadamente na recolha de objectos da caixa de ferramentas, no aspirar e no copiar. Também esta criança permaneceu pouco tempo no computador. Penso que este facto se deve à hora a que eles foram por mim chamados, pois coincide com altura do recreio, no qual eles andam nos baloiços, nos escorregas, etc.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Início: 11h45m **Fim:** 12h15m

FG (age 5) / JPF (age 3) – this matches session 2 of group 10.

Objectivos Prévios:

Identificar e trabalhar com algumas ferramentas do TOON TALK.

Desenrolar e Resultados Gerais:

Nesta sessão o JPF conseguiu, após eu o ter lembrado, realizar as tarefas com bastante facilidade. Ele recolheu, aspirou e copiou objectos com bastante à vontade, o único problema foi a quantidade de objectos que ele tirou para o chão da casa que a transformou num "campo de batalha". No fim ele quis explodir as casas, tarefa que lhe deu um enorme gozo e que repetiu vezes sem conta.

O FG teve mais dificuldades. Ele não se conseguia lembrar de nada do que tinha feito na sessão anterior, e após eu o ter relembrado, continuou a sentir dificuldades em realizar algumas das tarefas, nomeadamente o recolher objectos, aspirar e o copiar.

Lista de elementos de registo (ficheiros) associados:

Observações:

A facilidade demonstrada pelo JPF na realização das tarefas é de facto algo que me surpreendeu, dado que esta criança tem algumas dificuldades em realizar desenhos no Paint e jogos no computador.

Session 3

Início: 11h30m **Fim:** 12h07m

FG (5 anos) / JPF (3 anos) – this matches session 3 of group 10.

Objectivos Prévios: Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O FG entrou na casa e tirou alguns objectos que em seguida aspirou. Posteriormente, voltou a tirar alguns objectos da mala de ferramentas e fez cópias com a varinha mágica. Após ele ter realizado algumas tarefas que já tinha realizado na sessão anterior, eu disse-lhe que poderíamos ensinar o robot a trabalhar sozinho. Ele mostrou-se interessado e então expliquei e exemplifiquei como se faz para que ele tentasse programar o robot.

Foi o que ele fez. Pegou numa caixa, a essa juntou outra e colocou numa um camião e noutra uma balança. Em seguida pegou nessa mesma caixa e deu-a ao robot. Desta forma entrou no pensamento e ensinou o robot a aspirar os objectos inseridos nas caixas. Em seguida o robot retirou um camião e uma balança da mala de ferramentas e colocou-os com a mesma disposição nas caixas. O FG saiu do pensamento do robot e testou-o para verificar se o robot aprendeu o que ele lhe ensinou.

O FG teve algumas dificuldades em programar o robot, dificuldades essas que foram ultrapassadas com a minha ajuda.

O JPF esteve todo o tempo a observar o desenrolar da actividade realizada pelo FG, o que lhe facilitou um pouco a sua aprendizagem.

Esta criança quis logo ensinar o robot. Pegou numa caixa colocou um camião pegou noutra caixa e colocou uma letra. Em seguida juntou as duas caixas e deu-as ao robot. Já dentro do pensamento, ensinou o robot a aspirar o conteúdo das caixas e a colocar novamente os objectos nas mesmas. Após ter programado o robot, o JPF verificou se o robot tinha ou não aprendido tudo o que lhe foi ensinado. Após ter feito a verificação, o JPF quis rebentar com a casa.

O JPF conseguiu realizar esta tarefa, embora tivesse sentido algumas dificuldades, no entanto essas dificuldades foram menores que as sentidas pelo FG.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 4

Início: 09h40m **Fim:** 10h12m

FG (age 5) / JPF (age 3) – this matches session 4 of group 10.

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô

Desenrolar e Resultados Gerais:

O FG sentiu grandes dificuldades na realização da tarefa. Não conseguiu entrar na casa à primeira vez, confundiu a tecla F1 com a tecla Esc quando quis fazer desaparecer o marciano e não se lembrou de como se faz para sentar o boneco.

Quando entrou na casa, foi logo buscar uma bomba e rebentou com a mesma. Na outra casa, ele já foi para a programação do robô. Retirou uma caixa e uma letra da mala de ferramentas e colocou a letra no interior da caixa. Em seguida aspirou a letra. Após ter treinado cá fora ele deu uma caixa com a letra A ao robô e programou o robô a aspirar de uma caixa a letra A. Teve muitas dificuldades em programar o robô, e como tal estive constantemente a apoiá-lo.

Depois programou outro robô a aspirar um camião, mas teve na mesma grandes dificuldades. Essas dificuldades também se fizeram sentir aquando da parte de testagem do robô, pois das duas vezes ele cometeu o mesmo engano, não tendo dado uma caixa idêntica à que o robô tinha no pensamento.

O JPF entrou no pensamento do robô e ensinou-o a aspirar de uma caixa um nº, em seguida testou o robô tendo depois rebentado com a casa.

Já noutra casa o JPF entrou no pensamento do robô e pegou numa bomba e fê-la rebentar. Testou o robô e gostou de o ver a rebentar com a casa.

O JPF não teve grandes dificuldades em programar o robô, poucas foram as vezes que tive de intervir Para o alertar para este ou aquele aspecto.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 5

Início: 10h05m **Fim:** 10h25m

FG (5 anos)

Objectivos Prévios: Avaliar o desempenho da criança.

Desenrolar e Resultados Gerais:

O FG entrou na casa e esteve a aspirar alguns objectos por ele retirados da mala de ferramentas. Depois, por minha influência, programou um robô a aspirar objectos. Ele deu uma caixa com um camião ao robô, para entrar no seu pensamento, e já dentro deste, retirou um camião, um ninho e uma balança e aspirou tudo. Em seguida veio verificar o robô. Deu uma caixa com uma balança ao robô e este não funcionou. Ele não conseguiu decifrar o problema, então tive de o levar a descobrir o problema. Perguntei se ainda se lembrava o que é que o robô tinha no seu pensamento e, aí ele lembrou-se que o robô pede sempre o que quer para poder trabalhar. Então, deu a caixa certa e o robô funcionou.

Para terminar, o FG andou a rebentar casas e pilotou um pouco o heli.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 6

Início: 15h11m **Fim:** 15h39m

FG (age 5) / LE (age 5) – this matches session session 6 of group 7.

Session 7

Início: **Fim:**

FG (age 5) / LE (age 5) – this matches session 7 of group 7.

Group 9

Session 1

Início: 14h40m **Fim:** 15h06m

AF (age 5) / T (age 5)

Objectivos Prévios:

Introdução ao TOON TALK (breve apresentação do ambiente do Toon Talk)/ Trocar caixas/objectos no seu interior/programar o Robô.

Desenrolar e Resultados Gerais:

A T entrou na casa e retirou alguns objectos e fez cópias dos mesmos. Em seguida aspirou o chão da casa e deu ao robô uma caixa com uma balança. (Teve algumas dificuldades em agarrar objectos e em trabalhar com a varinha mágica e com o aspirador, nomeadamente para os fazer funcionar, pois esquecia-se que os tem de ter na mão para eles trabalharem). No pensamento ela ensinou o robô a fazer cópias de camiões, retirou um camião, agarrou na varinha mágica e fez 3 cópias seguidas. Em seguida testou o robô e rebentou com a casa.

O AF, ao contrário do que esperava, conseguiu pilotar e fazer aterrar o helí, entrou na casa e começou a retirar objectos da mala de ferramentas e a colocá-los no chão da casa. Tudo isto foi feito sem eu lhe ter ensinado nada nem sequer lhe disse como se fazia. Penso que o facto do ToonTalk estar instalado no PC da sala de actividades foi um factor preponderante para que esta criança tenha conseguido, na primeira sessão a que está presente, obter tão bom resultado.

Lista de elementos de registo (ficheiros) associados:

Observações:

A actividade desenvolvida com o AF teve como objectivo inicial, "introduzir a criança" no ambiente do toon talk, no entanto esta criança demonstrou ter já um conhecimento deste programa, facto pelo qual eu permiti que esta avançasse até onde avançou sem restrições da minha parte, uma vez que quando me preparava para começar a explicar o ambiente de trabalho, já ela estava a pilotar o helí, tendo eu optado por ver até onde ela chegava.

Session 2

Início: 00h00m **Fim:** 00h00m

AF (age 5) / T (age 5)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

Tanto o AF como a T não conseguiram realizar a tarefa proposta. No caso do AF, eu já esperava que a sua prestação não fosse a melhor (é uma criança com necessidades educativas especiais, embora não esteja declarado como tal). Desde o momento em que percebi que ele não conseguiria, mudei a actividade, uma vez que ele tem vindo a melhorar nas actividades anteriores. Assim esteve a trabalhar com as diversas ferramentas (aspirador, varinha mágica, bomba) e a ensinar o robô a realizar tarefas como copiar, ou retirar um objecto da caixa. Foram sempre tarefas que englobassem uma só acção do robô para que a criança percebesse o que lhe estava a ensinar.

A T fez também o mesmo género de actividade. No entanto ela levou para o pensamento do robô uma caixa com dois compartimentos para trocar os objectos nela inseridos. Acabou por fazer mais ou menos a actividade programada, sem que fosse objectivo fazer a correspondência dos objectos da caixa com a folha, pois ela não se mostrou interessada em a realizar dessa forma, embora eu, disfarçadamente a tenha tentado "empurrar para a actividade programada".

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

Início: 00h00m **Fim:** 00h00m

AF (age 5) / T (age 5)

Objectivos Prévios:

Fazer corresponder a imagem da folha com a localização dos bonecos (carro, peão).

Desenrolar e Resultados Gerais:

O AF, com bastante dificuldade, realizou a actividade programada. Primeiro fez a troca dos objectos "manualmente", e aí correu tudo bem, depois ensinou o robô a trocar da mesma forma esses objectos de acordo com a imagem que estava na folha (note-se que a actividade proposta para esta criança foi a actividade proposta para a sessão anterior a qual ele não conseguiu realizar), aqui tive de lhe explicar mais uma vez o objectivo da troca dos sinais e ajudá-lo a realizar essa mesma troca. Penso que ele não percebeu o objectivo da actividade, embora tivesse dito que sim.

A T realizou a mesma actividade, mas conseguiu, praticamente sem a minha ajuda, chegar ao fim da actividade. O mais importante para mim foi o facto da T ter entendido o porquê de destrocar a localização das imagens, durante a actividade.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 10

Session 1

Início: 11h00m **Fim:** 11h18m

FG (age 5) / JPF (age 3) – this matches session 1 of group 8.

Session 2

Início: 11h45m **Fim:** 12h15m

FG (age 5) / JPF (age 3) – this matches session 2 of group 8.

Session 3

Início: 11h30m **Fim:** 12h07m

FG (age 5) / JPF (age 3) – this matches session 3 of group 8.

Session 4

Início: 09h40m **Fim:** 10h12m

FG (age 5) / JPF (age 3) – this matches session 4 of group 8.

Session 5

Início: 10h40m **Fim:** 11h08m

R (age 3) / JPF (age 3)

Objectivos Prévios:

Trocar caixas/objectos no seu interior/programar o Robô.

Desenrolar e Resultados Gerais:

O JPF deu uma caixa com um nº ao robô, entrou no pensamento. Aí ele retirou uma caixa e um camião da mala de ferramentas, colocou o camião no interior da caixa o camião e pegou no aspirador e aspirou a o camião da caixa. Saiu do pensamento e testou o robô. Depois de ter visto o resultado, foi buscar outro robô, deu-lhe uma caixa com uma bomba, e já no interior do pensamento retirou uma bomba da mala de ferramentas e acionou-a.

A R esteve presente, no entanto não quis trabalhar no Toon Talk.

Lista de elementos de registo (ficheiros) associados:

Observações:

A R nunca esteve presente nas sessões anteriores do Toon Talk. As restantes sessões do JPF estão no anterior grupo ao qual pertencia.

Session 6

Início: 11h25m **Fim:** 11h55m

R (age 3) / JPF (age 3)

Objectivos Prévios:

Desenrolar e Resultados Gerais:

O JPF começou logo por entrar no pensamento do robô, dando uma caixa com um camião ao robô. Depois ensinou-o a aspirar objectos que foram previamente retirados da mala de ferramentas. Depois agarrou num ninho, e por fim pegou numa bomba e accionou-a. Fora do pensamento ele fez logo explodir a casa sem sequer ter testado o robô

Posteriormente o JPF andou a construir e a destruir casas.

A R esteve a pilotar o heli e dentro da casa esteve a agarrar alguns objectos.

Sentiu algumas dificuldades, nomeadamente a retirar os objectos da mala de ferramentas, pois para tirar um determinado objecto, ela tirava outro. Não quis avançar muito com a R, uma vez que ela nunca está presente, ou quando está não quer vir para os computadores.

Lista de elementos de registo (ficheiros) associados:

Observações:

Mondrões preschool

First session, all groups:

Esta sessão serviu para apresentar o TT às crianças, assim como alguns conceitos base da sua funcionalidade:

1. Formar grupos de pares (A formação destes grupos tiveram como critério a sua idade e os anos de projecto (ICEI), existe 1 par que já é o 3º ano que trabalha com o computador);
2. Começámos por escrever os nomes de cada par, depois escolhemos qual a aparência do boneco do jogo; já no jogo perguntei (o que é aquilo ali em baixo?) ao início fixaram os olhos e não deram resposta, mas assim que (com a minha ajuda) começámos a descer as sugestões variaram entre casas (para a maioria) no entanto houve uma criança que achava que a casa do meio era uma pá, no entanto o seu colega disse logo que não, "aquilo são casas"; houve também outra que rapidamente disse que estava num helicóptero e que aquilo era fogo. Quando lhes perguntei como se chamava o transporte, todos disseram que era um helicóptero.
3. Quando aterrámos saímos do helicóptero, e eu sugeri que tentassem sozinhos, prontamente o fizeram, no entanto os que estão pela 1ª vez no jardim não conseguiram fazer sem a minha ajuda.
4. Voltaram a aterrar e entrámos na casa, depois tentaram tirar coisas da caixa (os que estão pela 1ª vez não conseguiram fazer sozinhos);
5. Expliquei como se levantavam e depois como tinham tanta coisa no chão falei que teríamos de limpar. Nesse momento começámos a trabalhar com o aspirador.
6. Saímos da casa e fomos para o helicóptero e depois terminámos.

Todas as crianças experimentaram individualmente.

General session goals (common to all groups)

Session 2

- Identificar o transporte;
- Subir e descer no helicóptero
- Entrar na casa;
- Tirar objectos da caixa;
- Aspirar;
- Voltar a entrar no helicóptero.

Session 3

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estivemos a fazer da última vez?
  - Construir casas.

Session 4

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estivemos a fazer da última vez?
  - Construir casas. Identificar de quem é a casa.
- Ensinar um Robô a construir uma cidade.

Group 1

Session 2

GRUPO: (1) L + R

TEMPO: 9h00 às 9h50

Nº DE SESSÃO: 2ª

OBJECTIVO: Conseguir fazer o que está proposto nos objectivos gerais

ACTIVIDADE: O L conseguiu: aterrar; entrar na casa e tirar algumas coisas da caixa, no entanto precisou de ajuda para aspirar. A R necessitou de ajuda para o cumprimento dos objectivos gerais.

Session 3

GRUPO: (1) L + R

TEMPO:

Nº DE SESSÃO: 3ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- Construir casas;
- Identificar a quem pertence cada casa (colocando um objecto à sua escolha dentro da caixa)

ACTIVIDADE:

O L faltou.

Não conseguiu descer o helicóptero sem ajuda, depois de uma explicação conseguiu fazer sozinha. Entrou sozinha na casa e conseguiu baixar o boneco.

Retira objectos da caixa sem grande dificuldade. O que se pode concluir é que neste momento (3ª sessão) a R consegue manipular o rato.

Quando lhe perguntei se ainda se recordava como se construíam casas a R rapidamente me explicou e depois executou. Sempre que construía uma casa, levantava o boneco, saía e subia no helicóptero para as ver. Com a minha sugestão entrou nas casas que construía e verificava que estava lá dentro o Robô e a caixa que lhe era dada (com o objecto que ela tinha escolhido)

Session 4

GRUPO: (1) L + R

TEMPO: 9h15 às 9h45

Nº DE SESSÃO: 4ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estive-mos a fazer da ultima vez?
    - Construir casas. Identificar de quem é a casa.

ACTIVIDADE:

O L tem um controlo de rato muito bom. Na construção da casa foi a R que apontava no ecrã o que tinha de tirar da caixa. Depois já conseguiu fazer sem ajuda.

A R conseguiu fazer tudo sem ajuda, no entanto às vezes tem dificuldade em tirar objectos da caixa, mas esforça-se por fazer sozinha.

A R subiu no helicóptero e mostrou ao L as casas... (fiquei surpreendida com a atitude da R)

O L também quis fazer o mesmo.

Group 2

Session 2

GRUPO: (2) RA + S

TEMPO: 10h30 às 10h45

Nº DE SESSÃO: 2ª

OBJECTIVO: Conseguir fazer o que está proposto nos objectivos gerais

ACTIVIDADE: A S precisou de ajuda para o cumprimento dos objectivos gerais (foi o RA que a ajudou, apesar das dificuldades que tem).

Estiveram a tirar objectos da caixa.

O RA apresentou mais dificuldades quando foi a vez dele. (necessitou da minha ajuda)

Session 3

GRUPO: (2) RA + S

TEMPO:

Nº DE SESSÃO: 3ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- Construir casas.
- Identificar a quem pertence cada casa (colocando um objecto à sua escolha dentro da caixa).

ACTIVIDADE:

A S é uma criança que apesar de ter 3 anos, os seus períodos de concentração que ainda são muito reduzidos. Em relação à motricidade fina ainda está muito pouco desenvolvida. Nesta primeira fase a S irá ter um acompanhamento com actividades do ICEI.

O R esteve a ajudar a S a construir uma casa (ao que a S comentou "Olha o camião foi à Vila").

O R esteve a construir casas com alguma ajuda minha (tem dificuldade na manipulação do rato). Depois disse-me que não queria mais.

Session 4

GRUPO: (2) R + S

TEMPO: 9h45 às 10h15

Nº DE SESSÃO: 4ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estivemos a fazer da ultima vez?
    - Construir casas. Identificar de quem é a casa.

ACTIVIDADE:

O R está perfeitamente capaz de construir casas e esteve a ensinar a S, que não sabia fazer (ajudou-a, colocando a sua mão sobre a dela).

A S mantém-se sempre desatenta dificultando a ajuda do R.

O R ainda tem alguma dificuldade em controlar o rato na totalidade, mas vai conseguindo.

A tentativa de ajudar a S foi repetida, mas em vão.

Group 3

Session 2

GRUPO: (3) D + M

TEMPO: 10h45 às 11h05

Nº DE SESSÃO: 2ª

OBJECTIVO: Conseguir fazer o que está proposto nos objectivos gerais

ACTIVIDADE:

O D apresentou dificuldades em todo o cumprimento dos objectivos gerais. O M está mais autónomo, atingiu os objectivos gerais.

Tem dificuldade em sair e andar com boneco.

Session 3

GRUPO: (3) D + M

TEMPO:

Nº DE SESSÃO: 3ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- Construir casas.

ACTIVIDADE:

O D e o M ainda têm alguma dificuldade na manipulação do rato, o que se nota no tirar objectos da caixa e no sair da casa com o boneco.

No entanto os dois construíram casas e subiram no helicóptero para as ver.

Não se mostraram interessado em continuar.

Session 4

GRUPO: (3) D + M

TEMPO: 10h15 às 10h45

Nº DE SESSÃO: 4ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estive-mos a fazer da ultima vez?
  - Construir casas. Identificar de quem é a casa.
- ensinar um Robô a construir uma cidade.

ACTIVIDADE:

O D e o M conseguiram construir casas, e quiseram contá-las.

Depois mostrei como se poderia construir uma cidade.

Quer o D, quer o M, conseguiram ensinar o robô, saindo para a rua e subindo no helicóptero para ver a construção.

Mostraram-se muito surpreendidos com o que tinham feito. Reparei que o M ficou mais surpreso do que o D.

Group 4

Session 2

GRUPO: (4) T + RF

TEMPO: 11h10 às 11h25

Nº DE SESSÃO: 2ª

OBJECTIVO: Conseguir fazer o que está proposto nos objectivos gerais;

Construir casas

ACTIVIDADE:

O RF faltou. A actividade foi realizada individualmente.

O T está autónomo conseguiu realizar os objectivos gerais.

Construir casa (quando tiramos um camião, ele anda um bocadinho; quando colocamos alguma coisa sobre ele, o camião volta a andar mais um bocado) então sugeri que ele experimentasse os objectos da caixa ou 2 objectos para ver se ele andava mais (com algumas sugestões minhas colocou uma caixa e um robô); depois fomos ver para onde teria ido o camião, como não o encontrava subiu no helicóptero (minha sugestão) e lá estava outra casa, aterrou e foi ver se estava lá o robô. Fez outra casa colocando na caixa uma letra e depois subiu no helicóptero para ver, desceu e entrou na casa para saber se estava lá o robô que tinha colocado no camião. Quis explorar toda a ilha, então entrou no helicóptero e aterrou muito longa das casas, queria encontrar o mar.

Session 3

GRUPO: (4) T + RF

TEMPO:

Nº DE SESSÃO: 3ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- Construir casas;
- Consolidar a construção de casas;
- Identificar a quem pertence a casa;
- Ensinar o Robô a fazê-las;
- Guardar no caderno, com o nome.

ACTIVIDADE:

O T explicou ao R como se construía uma casa, em simultâneo construiu uma casa. O R quando tentou fazer a casa mostrou muita dificuldade em manusear o rato, então sugeri que o T o ajudasse. O T quando estava no helicóptero começou a querer ir ao mar então tentou, mas não conseguiu, eu disse-lhe que era um pouco complicado, pois aquele mar era muito fundo e que para o menino não cair na água o jogo não deixava ir para o mar.

Session 4

GRUPO: (4) T + RF

TEMPO: 10h50 às 11h20

Nº DE SESSÃO: 4ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estive-mos a fazer da ultima vez?
  - Construir casas. Identificar de quem é a casa.
- ensinar um Robô a construir uma cidade.

ACTIVIDADE:

O T voltou a mostrar ao R como se fazia uma casa. Em seguida o R fez sem dificuldade.

Exemplifiquei e expliquei como se poderia construir uma cidade e cada um fez o mesmo, divertiram-se muito com o ver a construir as casas.

O T quis novamente ir até ao mar, mas como não conseguia entrar na água acabou por se lembrar da minha explicação e até disse "Pois é não se pode ir nadar, porque este mar é muito fundo. No entanto continuou a explorar as casas e quando levantou voo viu que a ilha estava repleta de casas. Pediu-me para repetir a construção.

O D quis ficar só a ver o que o T fazia.

Group 5

Session 2

GRUPO: (5) SI + SM

TEMPO: 9h55 às 10h15

Nº DE SESSÃO: 2ª

OBJECTIVO: Conseguir fazer o que está proposto nos objectivos gerais;

Construir casas

ACTIVIDADE: No ecrã apontaram para o ícone do TT: e o SM clicou e entrou. Os dois estão autónomos conseguiram realizar os objectivos gerais. Construíram casas o procedimento utilizado foi igual ao utilizado com o T. A SI percebeu logo que o camião com o robô e uma caixa construia casas "O Robo fez a casa!!!" (SI)

Session 3

GRUPO: (5) SI + SM

TEMPO:

Nº DE SESSÃO: 3ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- Construir casas.
- Ensinar o Robô a fazê-las;
- Guardar no caderno, com o nome.

ACTIVIDADE:

Os dois recordavam-se de tudo o que tínhamos feito na última sessão (construir casas) e começaram a fazê-lo ora um ora o outro (quando não estavam com o rato davam sugestões ao que estava a fazer).

Toda a sessão foi passada em: tira camião, mete robô, mete caixa, levanta, saí da casa sobe no helicóptero e conta as casas.

Tentei sugerir em construir uma cidade, mas nenhum me ligou, queriam era construir casas e conta-las freneticamente.

SM: "Queres que tire uma bomba?"

SI: "Sim."

SI: "arruma a bomba que pode explodir!!!"

Continuaram a construir casas e a contá-las.

Session 4

GRUPO: (5) SI + SM

TEMPO: 11h20 às 11h45

Nº DE SESSÃO: 4ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- O que estive-mos a fazer da ultima vez?
  - Construir casas. Identificar de quem é a casa.
- ensinar um Robô a construir uma cidade.

ACTIVIDADE:

Cada um construiu uma casa.

Eu comecei por explicar e exemplificar como se poderia construir uma cidade, ensinando o robô.

O SM executou e esteve dentro do helicóptero a ver construir.

A SI fez o mesmo.

Ambos se mostraram perplexos com a rapidez com que os trabalhadores construíam as casas.

Mostraram dificuldade em compreender o pensamento do robô:

SM - estou um bocadinho baralhado!?

SI - eu também.

Neste momento expliquei-lhes que quando queremos fazer alguma coisa temos de pensar no que queremos fazer, por isso o robô também tinha de pensar...

Como os robôs não pensam sozinhos é preciso ir ao pensamento para lhes ensinar.

Acho que ficaram convencidos.

Group 6

Session 2

GRUPO: (6) P

TEMPO: 11h25 às 11h45

Nº DE SESSÃO: 2ª

OBJECTIVO: Conseguir fazer o que está proposto nos objectivos gerais;

Construir casas

ACTIVIDADE:

Muito autónoma, realizou tudo o que estava proposto nos objectivos gerais. Tirou várias coisas da caixa, no entanto aos pássaros deu objectos para eles levarem para o ninho.

Construiu 3 casas, mas teve alguma dificuldade em perceber que as construía, apesar de numa das caixas ter colocado um objecto à sua escolha (minha sugestão)

Session 3

GRUPO: (6) P

TEMPO:

Nº DE SESSÃO: 3ª sessão

OBJECTIVO:

- Descer o helicóptero;
- Sair e entrar na casa;
- Abaixar;
- Tirar objectos da caixa;
- Explorar os pássaros.

ACTIVIDADE:

Descobri que a P consegue ler.... (Eu quero esta com cabelo, apontando com o cursor no botão que dizia com cabelo)

Disse-me que tínhamos estado a construir casas e quando lhe perguntei o que faziam os pássaros respondeu-me que não sabia.

Então resolvi modificar os objectivos da sessão para a construção de casas.

E disse-lhe então vamos construir casas.

Para meu espanto a sucessão de construção da Patrícia foi:

1. tirou um camião;
2. colocou o robô no camião;
3. tirou um ninho e começou a dar coisas ao pássaro....
4. RC[343] - como se constrói uma casa?
5. P – com um camião um robô e uma caixa.
6. RC – e porque tiraste um ninho?
7. P – apeteceu-me dar coisas ao pássaro apara ele levar para o ninho.

Deixei a P continuar a fazer o que queria.

Ex:

1. tira camião;
2. coloca o robô no camião;
3. tira ninho;
4. tira uma caixa e coloca-a no chão;
5. coloca o pássaro na caixa;
6. dá objectos ao pássaro;
7. muda o pássaro de sitio e volta a dar-lhe objectos;
8. coloca uma caixa no camião;
9. repete
10. tira camião
11. coloca-lhe um robô;
12. tira ninho;
13. brinca com o pássaro (dá objectos ao pássaro)
14. coloca caixa no camião;
15. repetiu...

Passado algum tempo disse-me que queria ir contar as casas...

E foi.

Deixei a P explorar à sua vontade. Fiquei pasma com a sua capacidade de abstracção e concentração.

[343] The teacher.

Parada de Cunhos preschool

Groups with a single recorded session

Group 1

Data: 2002/10/25 **Início:** 9h 45m

Término: 19h 55m

Nome: B

Objectivos:

- descer/levantar helicóptero;
- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos.

Desenrolar e resultados gerais:

O B pousou rapidamente. Ajoelhou-se sozinho e conseguiu agarrar alguns objectos, espontaneamente, isto é, sem se aperceber o que estava a agarrar e sem se aperceber que o objecto pisca ao apontar. Não consegue controlar o rato (mão), e acabou por perder a caixa dos brinquedos. Ajudei a levantar mas visto estar constantemente a clicar, está constantemente a ajoelhar-se.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 4

Data: 2002/10/25 **Início:** 10h 33m

Término: 10h 55m

Nome: M / DI

Objectivos:

- ajoelhar-se/levantar-se;
- levantar/pousar helicóptero;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos;

Desenrolar e resultados gerais:

O M pousou rápido e ajoelhou-se sem intenção de o fazer.

Ajudei a agarrar alguns objectos, mas não quis experimentar mais.

O DI quis entrar no helicóptero, levantar e pousar de novo. Entrou dentro da casa, já sabe que clicando ajoelha-se. Ajudei a agarrar objectos e ensinei a construir casas. Quis repetir mas já não se lembrava do quais objectos eram necessários. Voltei a repetir mas sozinho não conseguiu fazer.

Ensinei também a rebentar as casas. Quis fazer sozinho mas também não conseguiu.

Lista de elementos de registo (ficheiros) associados:

Observações:

O DI experimentou construir casas e rebentá-las, contudo, não foi um objectivo proposto mas que se proporcionou.

Group 5

Data: 2002/10/25 **Início:** 10h 57m

Término: 11h 5m

Nome: F

Objectivos:

- notar que ao apontar o objecto treme;
- agarrar/largar objectos;
- copiar objectos.

Desenrolar e resultados gerais:

O F já sabe pousar e voltar a entrar no helicóptero. Como está sempre a clicar, está sempre a ajoelhar-se. Sabe levantar-se mas logo a seguir volta a ajoelhar-se. Sem se aperceber de que o objecto pisca quando se aponta, agarrou alguns e largou. Tentei ensinar a copiar objectos mas não quis repetir.

Lista de elementos de registo (ficheiros) associados:

Observações:

O F tem muita dificuldade em controlar o rato e como está constantemente a clicar, não consegue agarrar o que deseja.

Group 6

Data: 2002/11/29 **Início:** 14h 15m

Término: 14h 55m

Nome: MA **Idade:** 5 anos

Objectivos:

- descer/levantar helicóptero;
- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos.

Desenrolar e resultados gerais:

O MA identificou só o heli.

Clicando sem ajuda, descobriu como pousava e levantava.

Pousou sem ajuda e num acto de clicar com o rato, ajoelhou-se. Identificou a mão e começou a tirar objectos da caixa das ferramentas. Sem ajuda, descobriu que ao apontar o objecto treme e é aquele que a mão agarra. Tirou os objectos todos e experimentou várias vezes o agarrar/largar os objectos.

Sugeri que voltasse a arrumar os objectos para "arrumar a casa". Como não conseguia arrumar as letras, porque já tinha várias juntas ensinei a aspirar. Quis experimentar. Perguntou o que fazia a bomba de dar ar. Mostrei e também experimentou. Perguntou o que fazia a varinha mágica. Mostrei e experimentou muitas vezes, copiando muitas bombas de dar ar.

Perguntei se era capaz de tirar caixas azuis e arrumar alguns objectos dentro.

Como tirou muitas caixas, acabou por perder a localização de algumas delas. Pôs-se a pé para as procurar melhor e tudo isto sem ajuda. Quis mexer muito tempo com o rato e acabou por perder de vez a sua localização. Ajudei a encontrá-las. Ajoelhou-se e colocou os objectos tirados, dentro de casa caixa.

Perguntei se conseguia entrar no heli. Fê-lo bem.

Perguntou o que havia dentro das casas. Sugeri que visse ele próprio.

Pousou, entrou e apercebeu-se de que a cor do chão era diferente. Pediu para que a menina subisse ao andar de cima. Expliquei que a casa ainda não tinha escadas mas que mais tarde iríamos fazer umas.

Tive que pedir para dar o lugar ao colega.

Lista de elementos de registo (ficheiros) associados:

Observações: Foi a 1ª vez que o MA jogou no Toon.

Group 9

Data: 2002/10/25 **Início:** 15h 10m

Término: 15h 16m

Nome: AN

Objectivos:

- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;

Desenrolar e resultados gerais:

A AN não consegue manusear o rato, atendendo à idade que tem (2 anos).

Ajudei a clicar, baixando/levantando o heli.

Como não consegue manusear o rato nem clicar não quis jogar.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 2

Session 1

Data: 2002/10/25 **Início:** 9h 57m

Término: 10h 14m

Nome: D

Objectivos:

- descer/levantar helicóptero;
- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos.

Desenrolar e resultados gerais:

Ajudei o D a pousar o helicóptero. Como está sempre a clicar, ajoelhou-se logo. Ajudei a levantar-se.

Moveu muito rápido o rato e perdeu a caixa dos brinquedos. Soube levantar-se e ajudei a procurá-la.

Não quis jogar mais porque preferiu ver como é feito o rato.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Data: 2002/11/29 **Início:** 15h 51m

Término: 15h 54m

Nome: D **Idade:** 4 anos

Objectivos:

- notar que ao apontar o objecto treme;
- agarrar/largar objectos.

Desenrolar e resultados gerais:

Pousou.

Pedi para tirar objectos da caixa das ferramentas.

Tirou alguns mas com dificuldade, porque agarra objectos não pretendidos.

Pedi para tirar caixas azuis e arrumar lá dentro alguns objectos à escolha. Tirou as caixas mas não quis jogar mais (porque estava na hora de arrumar).

Lista de elementos de registo (ficheiros) associados:

Observações:

O D ainda não se apercebe muito bem que ao apontar o objecto treme. É capaz de agarrar melhor os objectos que são maiores (letra, número, caixa azul, camião).

Group 3

Session 1

Data: 2002/10/25 **Início:** 10h 16m

Término: 10h 31m

Nome: R / G

Objectivos:

- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos;
- aumentar/diminuir objectos;
- copiar objectos.

Desenrolar e resultados gerais:

O R pousou perto das casas. Saiu do helicóptero e entrou dentro das casas. Sabe ajoelhar-se sozinho e consegue levantar-se sem ajuda. Mostrei como se aumenta/diminui objectos, mas não quis experimentar. Ensinei a copiar objectos e experimentou uma vez e preferiu andar de helicóptero, para cima e para baixo.

O G pousou, entrou numa casa e sem se aperceber ajoelhou-se e tirou alguns objectos da caixa. Conseguiu agarrar alguns objectos, mas não se apercebeu que ao apontar o objecto pisca. Não quis jogar mais.

Lista de elementos de registo (ficheiros) associados:

Observações:

Pedi ao R para dar o lugar ao colega. O G não quis jogar mais.

Session 2

Data: 2002/11/04 **Início:** 11h 46m

Término: 12h 05m

Nome: R / G

Objectivos:

- notar que ao apontar o objecto treme;
- agarrar/largar objectos;

Desenrolar e resultados gerais:

O R pediu para construir casas porque tinha visto o A (grupo 8) a construir.

Ensinei mas teve dificuldades em agarrar os objectos pretendidos. Ainda não se apercebe do piscar do objecto apontado e acaba por agarrar objectos não desejados. Foi tirando muitos objectos da caixa e quis construir muitas casas. Atrapalhou-se com a quantidade de objectos e não foi muito bem sucedido da construção. Ajudei com a minha mão. Não quis jogar mais. O G pediu para entrar no heli mas já não sabia como levantar o menino. Relembrei a tecla. Também não era capaz de manusear o rato para entrar no heli. Ajudei a entrar no heli e sem querer voltou a pousar. Sem querer entrou em casa e voltou a sair. Não quis jogar mais.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 7

Session 1

Data: 2002/10/25 **Início:** 11h 47m

Término: 12h 5m

Nome: MR / C

Objectivos:

- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos;

Desenrolar e resultados gerais:

A MR pousou rápido mas sem ser perto das casas. Não consegue manusear o rato e por isso tem muita dificuldade em agarrar objectos.

Ensinei a copiar objectos mas não consegui que fizesse sozinha.

Não quis continuar a jogar.

A C ajudei a entrar no heli., ajudei a pousar. Mexeu muito tempo o rato e perdeu o local do heli. Tivemos que sair do Toon sem gravar e voltar a entrar.

Tem também muita dificuldade em clicar e mexer o rato ao mesmo tempo.

Ajudei a ajoelhar-se e a levantar-se. Ainda não consegue agarrar objectos sem ajuda.

Lista de elementos de registo (ficheiros) associados:

Observações:

Perdeu por várias vezes o local do heli e das casas.

Session 2

Data: 2002/11/04 **Início:** 14h 37m

Término: 15h 07m

Nome: MR / C

Objectivos:

- notar que ao apontar o objecto treme;
- agarrar/largar objectos.

Desenrolar e resultados gerais:

A C não sabe pousar. Ajudei e num clicar constante ajoelhou-se. Pediu para levantar uma vez que já não se lembrava. Como está sempre a mexer com o rato, perdeu o local do heli. Ajudei a encontrar porque pediu para entrar no heli. Ajudei e sem se aperceber do que fazia pousou perto das casas. Voltou a perder o local por estar sempre a mexer com o rato. Desapontada não quis jogar mais.

Tive que sair do Toon porque não conseguimos encontrar o heli.

A MR pousou, ajoelhou-se, voltou a levantar-se e quis entrar novamente no heli mas tal como a colega já não o encontrava. Ajudei. Preferiu estar sempre a pousar, sair do heli, e voltar a entrar. Sugeri ajoelhar e tirar alguns objectos, mas não fui bem sucedida.

Lista de elementos de registo (ficheiros) associados:

A MR e a C não são capazes de manusear o rato, isto é, estão constantemente a mexer com ele de maneira desorientada.

Observações:

Session 3

Data: 2002/11/22 **Início:** 11h 18m

Término: 11h 41m

Nome: MR **Idade:** 3 anos

Objectivos:

- pousar o heli perto das casas;
- agarrar/largar objectos;
- tirar objectos da caixa das ferramentas e colocá-los em caixas.

Desenrolar e resultados gerais:

A MR começou logo a clicar com o botão de pousar mas sem estar com atenção onde pousava.

Ajudei a pousar perto delas, entrou na casa e deixei-a explorar. Tirou alguns objectos e propus-lhe para arrumar aqueles objectos nas caixas azuis. Teve alguma dificuldade em arrumá-los cada um na sua caixa, porque estas estavam perto umas das outras. Separei as caixas e já foi mais bem sucedida. Quis experimentar a colocar um objecto em cima de outro na mesma caixa mas não resultou.

A MR ainda tem alguma dificuldade em agarrar o objecto pretendido, porque não repara no tremer do objecto.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 8

Session 1

Data: 2002/10/25 **Início:** 14h 20m

Término: 15h 5m

Nome: A

Objectivos:

- ajoelhar-se/levantar-se;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos;
- tirar caixas da caixa dos brinquedos;
- construir casas;
- aspirar/cuspir objectos;
- copiar objectos.

Desenrolar e resultados gerais:

O A já pousa sem ajuda e perto das casas. Ajoelha-se sem ajuda e já tira objectos da caixa. Ensinei a aspirar/cuspir objectos. Experimentou sozinho e foi bem sucedido.

Ensinei a construir casas e quis experimentar. Também foi bem sucedido e quis ficar o resto de tempo a construir casas e a verificar o resultado final (entrar no heli. E ver as casas construídas).

Lista de elementos de registo (ficheiros) associados:

Observações:

O A tem ainda alguma dificuldade em se aperceber de que o objecto pisca ao apontar. Não ensinei a copiar objectos por vontade da criança, que preferiu construir casas.

Session 2

Data: 2002/11/04 **Início:** 10h 14m

Término: 11h 05m

Nome: A

Objectivos:

- notar que ao apontar o objecto treme;
- ensinar o robô a trocar objectos.

Desenrolar e resultados gerais:

O A quis descer o heli mas já não sabia como se fazia (clicava sempre no botão da direita).

Pousou em cima das casas (muitas) que já tinha construído na sessão anterior.

Entrou e já não sabia ajoelhar-se. Quis construir mais casas e entrar no heli para ver o resultado. Voltou a pousar, entrou numa casa e quis tirar os objectos da caixa. Quando tirou o robô sugeri ensiná-lo a trocar objectos e mostrei como se fazia. Repeti várias vezes e pedi para fazer sozinho. Não quis. Pediu ajuda para arrumar todos os objectos e como teve dificuldades em arrumar o pássaro pediu para o ajudar a aspirar. Tirou muitos (5) camiões da caixa para construir muitas casas. Contudo, teve dificuldades ao colocar cada caixa no respectivo camião, atendendo à proximidade deles (a caixa era colocada dentro de outra caixa noutro camião). Fiz-lhe notar o piscar dos objectos, mas mesmo assim tive que ajudar com a minha mão. Quis ir de heli para poder contar quantas casas tinha feito. Não as conseguiu contar porque eram muitas.

Pousou novamente e quis construir uma no local onde pousou (jardim). Expliquei que só se pode construir casas quando estiver dentro de uma

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 3

This session is mentioned in the record of session 4, but no record of it was kept.

Session 4

Data: 2002/11/22 **Início:** 10h

Término: 10h 25m

Nome: A **Idade:** 5 anos

Objectivos:

- ensinar o robô a trocar objectos.

Desenrolar e resultados gerais:

O A pousou o heli perto das casas.

Perguntei se ainda se lembrava do que tinha feito na semana passada. Como já não se lembrava relembrei que ensinámos o robô a trocar objectos. Perguntei se ainda se lembrava como se fazia e respondeu que sim. Pedi então para me mostrar e disse-lhe que me ía ensinar como se fazia. Embaraçou-se dizendo que já não se lembrava. Sugeri que lhe repetisse os procedimentos. A sugestão não foi bem recebida e preferiu construir casas. Construiu uma casa e depois de a construir, começou por iniciativa própria a tirar duas caixas. Perguntei o que ia fazer e respondeu que ía ensinar o robô na troca dos objectos. Contudo, não avançou mais porque não se lembrava. Ajudei com a minha mão e fizemos. Quando saímos do pensamento do robô, não o quis testar e quis ver a casa que tinha construído.

Voltou a descer e quis explodir as casas. Insisti para que visse o que andava a fazer no robô dentro de casa. Entrou nas casas uma de cada vez e explodiu-as. Quis entrar no heli para ver que já não havia casas. Depois de entrar no heli e levantar disse desolado: oh fizeram-nas outra vez… Resolveu pousar e entrar numa casa. Ajoelhou-se e folheou o caderno dos desenhos. Tirou o heli a voar e quis colocar ao lado o heli a pousar mas colocou-o tão perto que este quase desapareceu ao ser martelado pelo rato. Expliquei que não poderia por tão perto porque pode haver um acidente. Não quis jogar mais.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 5

Data: 2002/11/29 **Início:** 11h 34m

Término: 11h 55m

Nome: A **Idade:** 5 anos

Objectivos:

- ensinar o robô a trocar objectos.

Desenrolar e resultados gerais:

O A pousou o heli perto das casas.

Sem ser orientado, entrou numa casa, ajoelhou-se e tirou o caderno dos desenhos. Pediu como se folheava. Expliquei e mostrei como se fazia. Gostou de ver os desenhos e tirou o da caixa das ferramentas. Quis tirar muitos desenhos iguais e colocá-los sobrepostos.

Perguntei se ainda se lembrava como se ensinava o robô a trocar objectos. Disse: "oh não quero fazer isso. Quero antes fazer casas." Perguntei se então preferia ensinar o robô a construir casas. Preferiu fazer um desenho.

Propus que fizesse um desenho à escolha e depois podíamos colocá-lo na casa onde estava.

Concordou e fez um desenho no Paint (casa e árvore). Copiei-o para o Toon.

Viu o resultado mas pediu para sair (estava na hora de arrumar para sair).

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 10

Session 1

Data: 2002/10/25 **Início:** 11h 14m

Término: 11h 40m

Nome: CR

Objectivos:

- notar que ao apontar o objecto treme;
- aumentar/diminuir objectos;
- construir casas;
- rebentar casas;
- copiar objectos.

Desenrolar e resultados gerais:

A CR pousou perto das casas. Entrou, ajoelhou-se. Já agarra bem os objectos mas tem alguma dificuldade em aperceber-se que piscam ao apontar.

Ensinei a copiar e experimentou sozinha com sucesso. Ensinei a rebentá-las e experimentou uma vez. Preferiu construir muitas casas.

Ensinei a folhear o caderno e procurar objectos para colocar dentro das caixas. Preferiu continuar a construir casas. Sempre que construía algumas casas, levantava no heli, para ver o resultado final.

Lista de elementos de registo (ficheiros) associados:

Observações:

Manuseia bem o rato.

A CR já sabe construir casas sem ajuda e agarra bem nos objectos.

Session 2

Data: 2002/11/04 **Início:** 10h 10m

Término: 10h 42m

Nome: CR

Objectivos:

- notar que ao apontar o objecto treme;
- ensinar o robô a trocar objectos.

Desenrolar e resultados gerais:

A CR pousou bem, quis entrar dentro de uma casa mas como mexeu muito tempo o rato perdeu o local. Como não encontrava o local das casas quis entrar no heli. Mas também perdeu o local. Tivemos que sair do Toon sem gravar e entrar outra vez. Pousou, entrou dentro de uma casa e quis rebentá-la. De seguida construiu uma nova e isto tudo sem qualquer ajuda.

Quis aumentar o aspirador mas não sabia como. Ajudei.

Perguntou o que fazia o robô e daqui sugeri ensiná-lo a trocar objectos. Mostrei como se fazia e pedi para repetir mas não foi capaz sozinha. Voltei a fazer com a mão dela mas não quis fazer sozinha. Preferiu construir casas e de seguida rebentá-las.

Perguntou se podia desenhar uma casa. Fizemos no Paint, e tentei copiar mas não resultou.

Perguntou se a menina não tinha amigos. Ensinei a folhear o caderno e tirar o desenho de um amigo. Fez sozinha.

Voltei a sugerir ensinar o robô mas não fui bem sucedida. Preferiu construir casas e rebentar.

Lista de elementos de registo (ficheiros) associados:

Desenho de uma casa no Paint.

Observações:

Session 3

This session is mentioned in the record of session 4, but no record of it was kept.

Session 4

Data: 2002/11/22 **Início:** 10h 31m

Término: 11h 12m

Nome: Carina **Idade:** 4 anos

Objectivos:

- ensinar o robô a trocar objectos.

Desenrolar e resultados gerais:

A Carina pousou perto das casas e perguntei se ainda se lembrava o que tinha feito na semana passada. Disse que sim e quis fazer sozinha mas já não se lembrava. Relembrei e começou por fazer sozinha, fora do pensamento do robô, indo buscar desenhos ao caderno (árvore e heli). Já no pensamento do robô, foi buscar desenhos diferentes (pessoa tonta e flor). Deixei fazer para no fim lhe explicar o sucedido. Expliquei que o temos que colocar os mesmos objectos para que o robô os reconheça, senão não sabe o que fazer.

Já não sabia como se saía do pensamento do robô. Relembrei.

Fora do pensamento, o robô repetiu tudo, inclusive "erros" e disse a Carina: "Este robô faz coisas confusas!"

Não quis repetir, por causa da confusão e esta foi a sua justificação. Preferiu folhear o caderno dos desenhos e tirar alguns desenhos, colocando-os em caixas. Teve um pouco de dificuldade em colocar os desenhos cada um na sua caixa, devido à proximidade destas. Assim, alguns dos desenhos ficavam sobrepostos e ficavam "cortados".

Pediu para sair.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 11

Session 1

Data: 2002/11/04 **Início:** 15h 8m

Término: 15h 25m

Nome: CA

Objectivos:

- notar que ao apontar o objecto treme;
- agarrar/largar objectos.

Desenrolar e resultados gerais:

Praticamente foi hoje a 1ª sessão para a CA. Mostrei como pousava e levantava o heli. Contudo, como não consegue manusear o rato foi uma tarefa muito difícil controlar os seus movimentos.

Deixei-a à vontade para fazer o que quisesse com o rato e o resultado foi o querer agarrar os objectos com a outra mão no monitor.

Com a minha mão a segurar ajoelhou-se e não conseguiu agarrar os objectos, visto estar sempre desatenta.

Lista de elementos de registo (ficheiros) associados:

Observações:

A CA tem muitas dificuldades em se concentrar e está sempre distraída. Não consegue manusear o rato, isto é, clicar e arrastar ao mesmo tempo e não consegue mexê-lo devagar. Na maior parte das vezes ajuda com a outra mão para o segurar o que acaba por não resultar muito bem. Tento ajudar mas é preciso fazer muita pressão na sua mão para mexer o rato devagar.

Falta muitas vezes.

Session 2

Data: 2002/11/22 **Início:** 14h 17m

Término: 14h 31m

Nome: CA **Idade:** 5 anos

Objectivos:

- pousar o heli perto das casas;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos;

Desenrolar e resultados gerais:

Para a CA esta foi a segunda sessão do Toon.

Como é uma criança com muitos problemas de concentração as dificuldades ainda se agravam mais.

Ajudei a pousar o heli e pedi para clicar à vontade com o botão que desejasse para ver o resultado. Distrai-se facilmente acabando por não escutar o que lhe é dito.

Ajudei a pousar e pediu como se entrava novamente para o heli.

Mostrei como se fazia, mas sozinha já não foi capaz.

Ajudei com a minha mão. Quis entrar em casa mas como tem muitas dificuldades em manusear o rato não foi capaz sozinha, isto é, sem ajuda manual.

Lista de elementos de registo (ficheiros) associados:

Observações:

A CA esteve ainda mais distraída porque as crianças de Gravelos lhes fizeram uma visita.

Session 3

Data: 2002/11/29 **Início:** 14h 56m

Término: 15h 49m

Nome: CA **Idade:** 5 anos

Objectivos:

- pousar o heli perto das casas;
- notar que ao apontar o objecto treme;
- agarrar/largar objectos;

Desenrolar e resultados gerais:

A CA já não se lembrava o que faziam os botões do rato.

Descobriu clicando à sorte. Contudo, já não conseguia pousar o heli.

Num acto de irritação, mexeu muito rápido o rato e o heli pousou.

Clicou e ajoelhou-se logo. Pedi para tirar alguns objectos da caixa das ferramentas. Com muita dificuldade conseguiu, após várias tentativas, agarrar alguns e colocá-los fora.

Pedi para tirar caixas azuis e colocar lá dentro alguns objectos. Fez devagar e algumas vezes agarrou os objectos que não queria.

Expliquei que só pode agarrar o objecto quando ele tremer.

Pedi novamente para tirar os objectos das caixas e colocá-los outra vez, na caixa das ferramentas.

Quis arrumar os passarinhos mas não conseguia. Mostrei como se aspiravam.

Pedi para se levantar mas já não sabia como. Relembrei.

Quis ajoelhar-se e levantar-se muitas vezes.

De pé, viu que haviam brinquedos por arrumar na caixa das ferramentas e quis arrumá--los mesmo de pé.

Expliquei que era preciso baixar-se para conseguir pegar nos brinquedos e experimentou com um brinquedo na sala de actividades.

Tentou várias vezes ajoelhar-se mas ajoelhava-se longe do local onde estavam os brinquedos. Ajudei.

Perguntou o que tinha o caderno. Mostrei-lhe o caderno dos desenhos e ensinei a folhear. Gostou de ver os desenhos e quis tirar alguns. Perguntei qual menina queria. Disse: “Não quero a doida porque não sabe o que faz. Quero a boa!”

Tirou vários desenhos.

Lista de elementos de registo (ficheiros) associados:

Observações:

A CA ainda tem algumas dificuldades em aperceber-se que os objectos ao apontar tremem.

Distrai-se facilmente e falta muitas vezes.

São Pedro Parque preschool

Group 1

Session 1

Nome1	A (3 anos)		
Nome2	D (3 anos)		
Hora inicial	9h08	**Hora final**	9h28
Objectivos prévios			
Identificar ambiente do ToonTalk. "Apresentação" do rato. Identificar sons e objectos.			
Desenrolar e resultados gerais:			
Ambos identificaram o helicóptero e as casas. O D aprendeu rápido a "sentar e levantar", repetindo a operação diversas vezes. O D e a A estiveram atentos durante toda a sessão. A A também repetiu o "sentar/levantar" diversas vezes.			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
O D é muito calado mas quando cativado aprende facilmente. A A tem uma grande energia, é muito divertida e gosta de experimentar todos os espaços do infantário.			

Session 2

Nome1	A (3 anos)		
Nome2	D (3 anos)		
Hora inicial	9h56	**Hora final**	10h19
Objectivos prévios			
Rever ambiente do ToonTalk. Aperfeiçoar a motricidade fina. Explorar e conhecer os comandos do Paint.			
Desenrolar e resultados gerais:			
A A já consegue dirigir os movimentos e sentar-se. A A quis experimentar muitas cores do Paint. O D consegue sentar e levantar o boneco sem dificuldades. Sentou e levantou o boneco diversas vezes, dentro e fora da casa.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint.			
Observações			
Exploraram as cores. Às 10h13 o Daniel perdeu a concentração.			

Session 3

Nome1	A (3 anos)		
Nome2	D (3 anos)		
Hora inicial	14h10	**Hora final**	14h29
Objectivos prévios			
Ambientar-se ao ToonTalk. Desenvolver motricidade fina. Trabalhar as cores. Noções: "vazia vs. cheia"; "dentro vs. fora".			
Desenrolar e resultados gerais:			
A A está a desenvolver rapidamente a capacidade de movimentação com o rato. Conseguiu perceber que agora é a mão dela que controla a mão do ToonTalk. Confunde o Esc (levantar) e o clic de sentar. Gosta de mexer muito com o rato a ver o helicóptero mexer-se.			
Lista de elementos de registo (ficheiros) associados:			
Caixa de coisas no caderno do ToonTalk. Desenho do Paint.			
Observações			
A A tem especial interesse por letras. Pede sempre para escrever o nome. O mesmo se passa em outras actividades, por exemplo a desenhar manualmente faz muitas aproximações a letras.			

Session 4

Nome1	A – 3		
Nome2	D – 3		
Hora inicial	10:11	**Hora final**	10:41
Objectivos prévios			
Fazer o desenho de abóboras no Paint, a pedido da educadora, porque estão a trabalhar sobre abóboras. Trabalhar as cores. Aprender o que é necessário para fazer casas no ToonTalk.			
Desenrolar e resultados gerais:			
O D gosta de ver o desenho sair na impressora: "vai sair, vai sair!". Para a A ainda é difícil controlar o traço no Paint. Para aterrar, no ToonTalk a A ainda não consegue fazê-lo à primeira. Mas quando lhe disse "levanta com o outro botão" ela foi logo carregar em "Esc", lembrando-se do "levantar o boneco". As 1ªs vezes que a A se sentou, o D carregou em "Esc" para ver o boneco levantar-se.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos no Paint – Abóboras.			
Observações			
Por problemas com a instalação do ToonTalk ainda não pude trabalhar com os desenhos do Paint no ToonTalk. Ao D parece não fartar o "sentar e levantar" o boneco e fazer casas: "outra vez! Outro camião, outra casa e outro robô!".			

Session 5

Nome1	A – 3		
Nome2	D – 3		
Hora inicial	9:15	**Hora final**	9:45
Objectivos prévios			
Trabalhar o controlo "liga/desliga" com o "space". Tirar coisas da caixa para aspirar.			
Desenrolar e resultados gerais:			
A A não identifica as casas. O D apontou-lhe as casas. A A já controla os movimentos do rato. A A não quer a ajuda do D para trabalhar com o ToonTalk. O D continua a querer ver o boneco a levantar-se continuamente. A A disse que queria pôr uma flor no telhado, "como a AL... e uma bola". A A já consegue tirar coisas da caixa e pousar no chão sem ajuda, mas não consegue trabalhar o rato e o "space" juntos. Carrega no rato, tentando "ligar/desligar". O D não consegue controlar o movimento da mão. Nota: parece-me que o "som" poderá ajudá-los muito a trabalhar.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
A A presta muita atenção ao trabalho desenvolvido também com os colegas. Por vezes a A pega no rato e levanta-o, querendo ver isso no ecrã. O D fica fascinado a trabalhar, embora precise constantemente de ajuda.			

Session 6

Nome1	A – 3		
Nome2	D – 3		
Hora inicial	14h37	**Hora final**	14h57
Objectivos prévios			
Usar desenho do Paint para decorar a casa ToonTalk. Usar caixas com coisas dentro como se fossem prendas para oferecer.			
Desenrolar e resultados gerais:			
A voar a A ainda não vão sozinha ter a uma casa. O D explicou-lhe onde estão as casas. A A pega bem nos objectos mas não encontra o caderno dos sensores nem percebe que é nele que tem que procurar a "parede faz-de-conta". O D também ainda não tem autonomia para usar cadernos.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint e cidade ToonTalk.			
Observações			
Se eu não disser nada, o D fica "espantado" a olhar para o ambiente ToonTalk e brinca com o rato. Hoje brincou com o rato a fazer ruído como se fosse um carrinho de brincar.			

Session 7

NOME: A

IDADE: 3 anos

DATA: 23-01-03

HORA DE INÍCIO: 9h20

HORA DE TERMINUS: 9h40

OBJECTIVOS PRÉVIOS: foi actividade livre para aferir conhecimentos do toontalk

DESENROLAR E RESULTADOS GERAIS: conseguiu sentar- se e começou a tirar coisas da caixa. Com ajuda conseguiu colocar uma foto numa moldura mas não conseguiu encontrar a moldura. A A começou a largar números em cima de outros. Perguntei-lhe o que estava a acontecer. Depois de ver mais algumas vezes, concluiu: "saiu outro número".

Session 8

GRUPO: A e D

IDADE: 3 anos

DATA:4-02-03

HORA DE INÍCIO: 10h25m

HORA DE TERMINUS: 10h45m

OBJECTIVOS PRÉVIOS: fazer caixa das trocas. Perceber a troca e conseguir fazer a troca.

DESENROLAR E RESULTADOS GERAIS: a A já consegue controlar bem os movimentos. Percebeu/ sabe o que é a troca, fazendo o exercício com destreza e ajudando o D a perceber a troca.

O D conseguiu estar concentrado mas não percebeu logo o valor da troca. Depois de repetir algumas vezes persistiram dúvidas. O D ainda precisa de ajuda para manipular o rato.

OBSERVAÇÕES: trabalhando com animais o D fica muito concentrado, nem que seja só o passarinho.

Session 9

GRUPO: A e D

IDADE: 3 anos

DATA: 13- 02- 03

HORA DE INÍCIO: 10h45m

HORA DE TERMINUS: 11h10m

OBJECTIVOS PRÉVIOS: trocas de imagens nas caixas, usando os desenhos feitos no paint alusivos à visita à azenha de Guiães.

DESENROLAR E RESULTADOS GERAIS: o D não sabia explicar a troca, nem o que era. Optei pela caixa de comparação. O D identificou as imagens iguais mas não percebeu a troca. Conseguiu fazer a troca mas para colocar as imagens iguais umas por baixo das outras.

Eu- "O que é a troca, A?

A - " trocar é passar!... para o outro lado". A A fez a troca com destreza. Depois perguntou se queria decorar a parede do quarto. Obtive um entusiasmado: "SIIMM!!"

OBSERVAÇÕES: a A esteve a colocar o nome e desenho na parede pois já percebe e faz bem a troca.

Trabalhamos com música, o que me tem revelado maiores níveis de concentração.

Session 10

GRUPO: D e A

IDADE: 3 anos

DATA: 11-03-03

HORA DE INÍCIO: 15h05m

HORA DE TERMINUS: 15h28m

OBJECTIVOS PRÉVIOS: Objectivo de sessão já enunciado

DESENROLAR E RESULTADOS GERAIS: O D, para não variar, gosta de sentar e levantar o boneco consecutivamente.

Quando apontei para onde queria que ele levasse o boneco, colocou o dedo no ecrã a tentar levar o boneco. Depois desatou a andar à roda com o boneco.

O D ainda não é autónomo a pegar em objectos do toontalk.

Não conseguiu deduzir os objectivos da actividade sem que lhos explicasse; ou seja, não começou a colocar as imagens na mesma sequência. Depois de alguma insistência acabou por perceber os objectivos da sessão mas precisa de ajuda na sua concretização.

A A coloca as imagens sem dificuldade nas caixas mas também teve dificuldade em perceber os objectivos do jogo, não colocando as imagens na sequência certa.

Session 11

NOMES: D e A

IDADE: 3 anos

DATA: 27-03-03

HORA DE INÍCIO: 10h15m

HORA DE TERMINUS: 10h26m

OBJECTIVOS PRÉVIOS: trabalhar a noção de troca

DESENROLAR E RESULTADOS GERAIS: a A disse "A troca é trocar dinheiro. Fez bem a troca. Não saía do pensamento do robô.

Eu- Quando lhe dás a caixa o que é que ele faz?

A – Trocou.

Eu- E porquê?

A- Porque já acabou.

Disse-lhe que para o robô não parar de trocar, qualquer coisa, tínhamos de lhe limpar a caixa pois assim ele trocaria o que quer que fosse. A A deu- lhe caixas com outras coisas e percebeu que ele continuava a trocar.

OBSERVAÇÕES: o D chegou só ao fim da manhã.

Session 12

NOMES: A e D

IDADE: 3 anos

DATA: 22- 05-03

HORA DE INÍCIO: 10h18m

HORA DE TERMINUS: 10h37m

OBJECTIVOS PRÉVIOS: construir com eles a caixa do que vão precisar para decorar a entrada com o nome e a parede com um desenho.

Desenho/ Parede/ Nome/ Entrada

Usar a caixa para os objectivos enunciados.

DESENROLAR E RESULTADOS GERAIS: o D já identifica bem as casas mas não aterrou em frente a nenhuma. O D faz o boneco sentar - se muitas vezes, mesmo quando não o quer fazer, mas sabe bem como o levantar.

Ao olhar a caixa o seu gosto por animais é notório: "olha um ovinho!".

Não tem abstracção para a actividade proposta e a A falou da caixa de trocas e quis fazer uma. Colocou, na caixa dupla, uma parede e um pássaro. Ao enganar - se, reparou que o que deu ao pássaro ele levou para o ninho.

Session 13

NOMES: A e D

IDADE: 3 anos

DATA: 05- 06-03

HORA DE INÍCIO: 9H20m

HORA DE TERMINUS: 9h50m

OBJECTIVOS PRÉVIOS: colocar nome no telhado. Construir casas. Fazer caixa de troca.

DESENROLAR E RESULTADOS GERAIS: a A não pousou numa casa. O D fez logo -ESC- e disse "não é assim!". Depois a A teve dificuldades em entrar na casa e pediu ajuda. Já não se lembravam do que é preciso para fazer uma casa. Mas a A já tem destreza para agarrar os objectos que quer agarrar.

O D percebeu que o robô tem de levar a mala para poder ir embora. O D já trabalha sozinho, precisando apenas de algumas indicações. Já dominam a troca, o que foi conseguido por sessões anteriores, culminando em êxito nesta.

OBSERVAÇÕES: não colocaram nome no telhado pois ainda não têm a abstracção necessária.

Group 2

Session 1

Nome1	M (3 anos)		
Nome2	R (3 anos)		
Hora inicial	9h30	**Hora final**	9h46
Objectivos prévios			
Apresentação do "rato". Promover a comunicação, sobretudo para a Rita que é bastante inibida.			
Desenrolar e resultados gerais:			
A R desatou a rir ao ver o boneco de apresentação e ao ouvir o som inicial "Tóim!!". Não identificaram o helicóptero. Só depois de lhe dizer que voava é que disseram que era um avião. Também não identificaram as casas. A R gostou de andar a voar com o "avião". Quanto ao boneco, tentou dar-lhe ordens falando com ele: "vai prò avião, fogo!". As duas repetiram ordens.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
São duas boas comunicadoras quando juntas. A R é mais inibida no dia-a-dia do jardim. Após 15 minutos começaram a ficar distraídas.			

Session 2

Nome1	M (3 anos)		
Nome2	R (3 anos)		
Hora inicial	9h20	**Hora final**	9h53
Objectivos prévios			
Relembrar os comandos do ToonTalk. Explorar o ambiente do ToonTalk e os comandos do Paint.			
Desenrolar e resultados gerais:			
A R já conseguiu baixar o avião e gosta de brincar com o rato, movimentando o avião e o boneco. Depois de entrar em casa a intenção é fazer "Pause" (aprender a interromper o jogo). Conseguiram perceber que foram fazer um desenho para guardar no caderno do ToonTalk. Quiseram desenhar as suas caras.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint.			
Observações			
Depois de fazerem o desenho, levantaram o boneco, após voltarem ao ToonTalk. Voltei a explicar, tal como na 1ª sessão, que o boneco, acabada a brincadeira, volta para o helicóptero e vai embora.			

Session 3

Nome1	R (3 anos)		
Nome2	M (3 anos)		
Hora inicial	11h35	**Hora final**	11h58
Objectivos prévios			
Desenvolver capacidade de trabalhar o rato. Noção de "fora vs. dentro"; "cheia vs. vazia".			
Desenrolar e resultados gerais:			
Já não se lembravam de como aterrar. A R sentou-se no jardim. Depois de entrar em casa a R quis andar com o boneco: "posso mexer um bocadinho?" Quando queria tirar uma caixa sem ajuda, não mexia a mão e, como tal disse: "olha, não sai!!!" A R não consegue arrastar objectos mas já sabe clicar. Quando íamos a sair eu disse: agora vamos levantar o helicóptero. M: "não é! é o avião! Esqueceste-te?"			
Lista de elementos de registo (ficheiros) associados:			
Caixas de coisas no caderno do ToonTalk. Desenhos do Paint.			
Observações			
A R fala com o computador como se este lhe obedecesse.			

Session 4

Nome1	R – 3		
Nome2	M – 3		
Hora inicial	9:10	**Hora final**	9:36
Objectivos prévios			
Identificação de cores e formas. Desenvolver a motricidade fina.			
Desenrolar e resultados gerais:			
A R e a M ainda têm muitas dificuldades em levar o rato para onde quer. M: Depois vamos fazer desenho! Sim? A M e a R perceberam bem o jogo e identificaram bem a sequência. Contudo, ainda não têm qualquer autonomia no ambiente ToonTalk; isto é, precisam de ajuda para pegar e largar objectos e para fazer movimentos com a mão e com o helicóptero.			
Lista de elementos de registo (ficheiros) associados:			
Caixa com objectos, no caderno ToonTalk.			
Observações			
Preparei o ambiente com uma caixa com quatro objectos diferentes que guardamos no caderno. No chão estavam dispersos objectos iguais. Quis que construíssem uma caixa para colocar na mesma ordem os objectos dispersos. A R e a M dispersam-se facilmente.			

Session 5

Nome1	R – 3		
Nome2	M – 3		
Hora inicial	9:20	**Hora final**	10:40
Objectivos prévios			
Trabalhar o controlo do rato. Iniciar a exploração do aspirador.			
Desenrolar e resultados gerais:			
A R explica à M como mexer o avião e pergunta "onde estão as casas?" e responde apontando: "estão ali" (apontou os telhados). A M já consegue tirar coisas da caixa de ferramentas mas não os coloca fora da caixa. M: "a mão nã via o que está ali!" (Mostrando que tem necessidade de ver o boneco.)			
Lista de elementos de registo (ficheiros) associados:			
Observações			
A R e a M ainda estão a explorar o controlo do rato. Não explorámos o aspirador, pois a R e a M ainda não têm qualquer controlo com o rato. Pediram para fazer um desenho, fomos ao Paint, desenhar – não conseguem desenhar, ou mesmo fazer riscos, sem ajuda.			

Session 6

Nome1	R – 3		
Nome2	M – 3		
Hora inicial	10h40	**Hora final**	11h02
Objectivos prévios			
Usar o aspirador para aspirar as imagens de Natal. Guardar as imagens de Natal no caderno.			
Desenrolar e resultados gerais:			
A M ainda não controla bem os movimentos do avião, não olhando para ver se aterra perto de uma casa. A M conseguiu perceber o barulho do aspirador e até disse: "A minha tia também tem um aspirador!". A R também precisa de ajuda para aspirar, pois ainda não entendeu que tem de apontar para o que quer aspirar. Expliquei-lhe como se faz.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk.			
Observações			
Ambas estão mais calmas a trabalhar, não clicando sem objectivo. Aperceberam-se de ter que desligar o aspirador para ele não ficar a fazer barulho.			

Session 7

NOME: R

IDADE: 3 anos

DATA:23-01-03

HORA DE INÍCIO: 11h00

HORA DE TERMINUS: 11h20

OBJECTIVOS PRÉVIOS: foi actividade livre para aferir conhecimentos do toontalk

DESENROLAR E RESULTADOS GERAIS: a R precisa de ajda para executar tarefas no Toontalk. Consegue tirar objectos da caixa de ferramentas. Perguntou: "Não vamos pôr nome?". Sendo assim, optamos por colocar a foto numa caixa azul, para podermos escrever o nome em baixo e também para ser mais fácil para a R, pois não teve de procurar a moldura do caderno.

Session 8

GRUPO: R e M

IDADE: 3 anos

DATA:04-02-03

HORA DE INÍCIO: 9h15m

HORA DE TERMINUS: 9h40m

OBJECTIVOS PRÉVIOS: fazer uma caixa com duas coisas diferentes para poder trocar

DESENROLAR E RESULTADOS GERAIS: a R conseguiu aterrar e entrar na casa sozinha.

A R fez uma caixa com um camião e um Robot. Conseguiu perceber como fazer e o que é a troca. Precisou de ajuda para perceber que os objectos tremem antes de podermos segurar neles.

A M também conseguiu perceber a troca mas pediu ajuda para trocar os objectos pois ainda tem dificuldade em manipular o rato.

OBSERVAÇÕES: a capacidade de concentração e destreza da M e da R tem evoluído positivamente.

Session 9

GRUPO: R e M

IDADE: 3 anos

DATA: 13- 02- 03

HORA DE INÍCIO: 9h10m

HORA DE TERMINUS: 9h44m

OBJECTIVOS PRÉVIOS: fazer duas caixas para fazer a "troca"

DESENROLAR E RESULTADOS GERAIS: a M precisou que lhe indicasse como se sentar e como caminhar sem se sentar. Já consegue ir buscar as caixas mas pergunta se deve clicar e onde, para fazer as coisas.

Quando levamos o desenho para o toontalk não sabia como reduzi – lo, já tinha esquecido.

"Como é a troca?"- expliquei que era mudar de lugar as coisas.

A R ainda tem muita dificuldade em segurar o que quer, diz: " Não consigo". Contudo, percebeu melhor a troca.

OBSERVAÇÕES: optei por criar uma caixa de comparação, explicando, assim, como trocar cria algo diferente do que temos de início. Assim, a M conseguiu perceber o que era "trocar".

Session 10

GRUPO: R e M

IDADE: 3 anos

DATA: 18-03-03

HORA DE INÍCIO: 10h20 m

HORA DE TERMINUS: 10h45m

OBJECTIVOS PRÉVIOS: já enunciados

DESENROLAR E RESULTADOS GERAIS: A R sobrevoa a cidade sem se cansar.

Depois, já perto do jogo, repete "eu não consigo, eu não consigo!". Expliquei-lhe que não pode ser preguiçosa nem dizer que não consegue antes de experimentar. É que a R diz o mesmo noutras actividades propostas pois tem medo de falhar. Acabou por conseguir perceber o jogo e realizá-lo com ajuda apenas nas indicações de movimentação. Percebeu a igualdade e sequência. A M nem precisou de explicações pois esteve atenta ao trabalho da R. Fez o jogo sem dificuldades. Ao colocar a árvore em cima da flor disse: "a flor roubou a árvore.

Eu: porque colocaste primeiro o passarinho?

M: Porque está do lado esquerdo.

Superou da dificuldade de largar coisas umas em cima das outras.

Session 11

NOMES: R e M

IDADE: 3 anos

DATA: 27-03-03

HORA DE INÍCIO: 11h15m

HORA DE TERMINUS: 11h36m

OBJECTIVOS PRÉVIOS: aprender a trocar dois objectos/coisas em caixas

DESENROLAR E RESULTADOS GERAIS: a destreza da M evoluiu muito.

M – Larga a caixa, mão. Ela nunca mais larga a caixa, PF! – não percebe que é ela a manipular a mão.

R – A árvore tem olhos.

M – É para ver bem.

R – Não gosto de árvores com olhos.

Eu - O que é trocar?

M – Trocar é pôr no outro lado.

A R ainda tem dificuldade em pegar e largar objectos do toontalk.

Eu – O que é que este Robô está a pensar?

M – Não sei.

Propus dar um nome ao Robô. Ficou "Sol". Elas ficaram animadas. Deram a caixa ao Sol.

Eu - Onde estamos?

M – Ensinamos – lhes a fazer o jogo.

Eu – O que aconteceu?

M – A flor tem duas cores.

OBSERVAÇÕES:

Session 12

NOMES: R e M

IDADE: 3 anos

DATA: 20- 05-03

HORA DE INÍCIO: 14h15m

HORA DE TERMINUS: 14h40m

OBJECTIVOS PRÉVIOS: colocar a imagem do peixe na parede, dando uma caixa dupla com: parede, desenho.

DESENROLAR E RESULTADOS GERAIS: vimos que já havia casas com nome. A M identificou o nome da casa "AX" e "G".

Elas construíram também uma caixa dupla para meter a parede e o desenho. Usaram a bomba de ar.

A M conseguiu fazer a caixa dupla sem ajuda. A R precisa de indicações.

Ambas já escrevem bem os próprios nomes.

Session 13

NOMES: R e M

IDADE: 3 anos

DATA: 05- 06-03

HORA DE INÍCIO: 11h10m

HORA DE TERMINUS: 11h46m

OBJECTIVOS PRÉVIOS: dar -lhes uma caixa com o desenho e parede para ver se conseguem trabalhar sozinhas. Fazer uma caixa sobre a troca.

DESENROLAR E RESULTADOS GERAIS: tornou se mais fácil perceberem que podem decorar a parede tendo -lhes dado a caixa já com as coisas necessárias.

R –E a porta não se fecha?

A R procura no caderno. Construiu cenários com sentido ao decorar a parede. Um deles era um menino, um avião e uma explosão. Disse: "o menino vai ver a explodir".

Trabalharam com a bomba de ar porque havia coisas que cobriam as outras, assim, exploraram tamanhos.

Já usam a tecla de atalho para chamar a bomba.

Ao pôr mais coisas a R testou sempre a ver como ficava. A M não vê como está a ficar. Conseguiram fazer a troca embora às vezes larguem imagens por cima de outras.

Group 3

Session 1a

Nome1	MN (3 anos)		
Nome2			
Hora inicial	9h48	**Hora final**	10h03
Objectivos prévios			
Desenvolver/promover a comunicação. Apresentação do ambiente do ToonTalk.			
Desenrolar e resultados gerais:			
A MN estava muito calada. Brinquei com ela dizendo-lhe que "o gato lhe tinha comido a língua". Ela mostrou a pontinha da língua para provar que sabe falar. Não consegui perceber se identificou o helicóptero pois, após alguma insistência continuou a não responder.			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
Mesmo noutros espaços da sala de jardim a MN está sempre muito quieta e calada.			

Session 1b

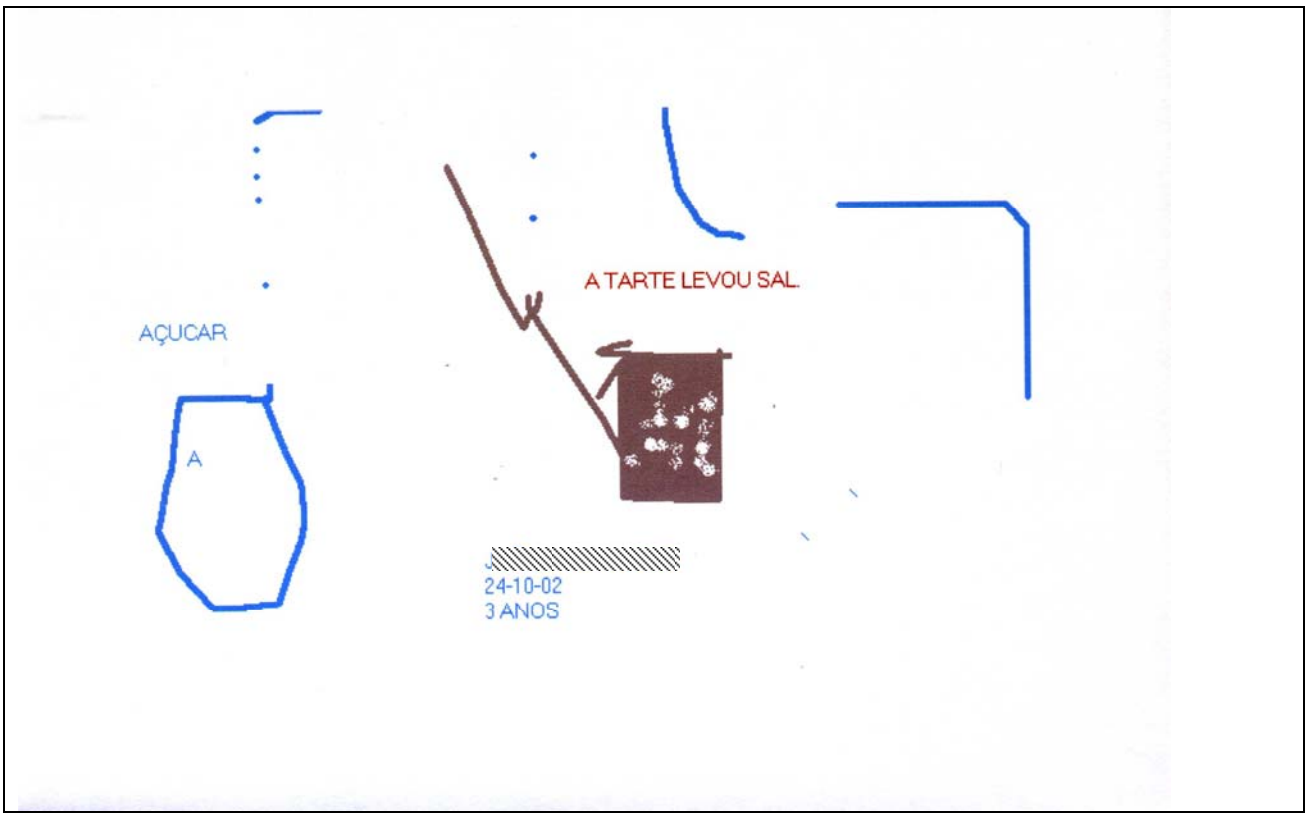
Nome1	J (3 anos)		
Nome2			
Hora inicial	15h06	**Hora final**	15h30
Objectivos prévios			
Identificar o ambiente do ToonTalk.			
Desenrolar e resultados gerais:			
Identificou o avião mas não identificou as casas. Como me apercebi que o João tem facilidade em manejar o rato, ensinei-lhe o robô construtor. J: "Agora quero fazer mais". Fez muitas casas e depois de levantar voo e ver o resultado quis fazer ainda mais. Ensinei-lhe a usar a varinha mágica. Usou-a sem dificuldades.			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
Foi o 1º contacto que o J teve com o ToonTalk.			

Session 2

Nome1	MN (3 anos)		
Nome2	J (3 anos)		
Hora inicial	14h15	**Hora final**	14h46
Objectivos prévios			
Desenvolver a capacidade de comunicação com a MN. Trabalhar/explorar ferramentas do Paint (como um auxiliar do ToonTalk).			
Desenrolar e resultados gerais:			
A MN identificou o avião. Já se lembrava do "sentar e levantar" pelo que o faz várias vezes, rindo-se. Como é muito pequena tem dificuldades em agarrar o rato. No Paint fez alguns riscos. Gosta de clicar de forma contínua como que gostando do "clic" que ouve. Começou a falar muito. Riu-se, disse coisas que foi descobrindo: "olha o passarinho" (e apontava). Ao imprimir os desenhos do Paint riu-se, revelando que o computador é uma experiência nova para ela. O J usou a varinha mágica para fazer muitos passarinhos. Esteve também a trabalhar "liga/desliga" com o aspirador. Interiorizou a ideia de desligar o aspirador após aspirar um determinado objecto.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint.			
Observações			

Session 3

Nome1	J (3 anos)		
Nome2	MN (3 anos)		
Hora inicial	14h35	**Hora final**	15h00
Objectivos prévios			
Desenvolver motricidade fina. Trabalhar a comunicação com a MN.			
Desenrolar e resultados gerais:			
A MN começou a repetir palavras das frases que eu dizia ao João. Ao ver a impressora a funcionar a MN começou a chamar-me e a dizer excitada: "olha, olha, olha mais..." Ao iniciar o ToonTalk começou: "olha, olha!" Quando disse ao João para largar o camião no chão ele largou o rato. A MN ao dar conta do movimento com o rato: "olha, fixe!!"			
Lista de elementos de registo (ficheiros) associados:			
Garagem de camiões no caderno do ToonTalk.			
Observações			
Expliquei que também nós temos que levar as coisas a onde as queremos arrumar. Exemplifiquei com uma caneta. A MN fez riscos no Paint. Achou graça aos riscos e disse que já não sabia o que levou (ingredientes) a tarte de maçã. A MN carrega no rato muitas vezes, sem perceber para quê.			



Session 4

Nome1	J – 3		
Nome2	MN – 3		
Hora inicial	10:27	**Hora final**	10:50
Objectivos prévios			
Trabalhar a igualdade e identificação dentro do ambiente do ToonTalk. Desenvolver a motricidade fina, em especial com a MN.			
Desenrolar e resultados gerais:			
A MN fica entusiasmada ao ver o desenho a sair na impressora: "A minha, a minha! Olha! ...Tira!! É minha." O J perguntou: "A árvore fala?". A árvore era uma das imagens guardadas em caixa. A MN consegue mexer o boneco a andar e ir para o helicóptero. O J gosta de explorar o espaço ToonTalk. Perguntou se o boneco entra sozinho no helicóptero. Expliquei que era ele que o fazia entrar. Ao brincar com o helicóptero perguntou: "Ele está doido?" Respondi: "não és tu que o mexes, não és?" J: Sim.			
Lista de elementos de registo (ficheiros) associados:			
Desenho de uma abóbora – Paint. Caixas de objectos – no caderno do ToonTalk.			
Observações			
O J começou por desenhar uma abóbora no Pain. A MN ainda não consegue segurar bem o rato. O J diz "Vou brincar!" quando movimenta o boneco em círculos pela casa (interior). J: "Quero fazer magia!" Deposi de me ver usar a varinha mágica.			

Session 5

Nome1	J – 3		
Nome2	MN – 3		
Hora inicial	9:50	**Hora final**	10:18
Objectivos prévios			
Trabalhar o controlo do aspirador e perceber a importância de manter o espaço limpo. MN – Trabalhar o controlo com o rato.			
Desenrolar e resultados gerais:			
No jardim, o J: "opss! Sentei-me!... Onde levanto?" O J conseguiu perceber o "liga/desliga" do aspirador, mas tem tendência a usar apenas a mão direita para trabalhar. Consegue, sem referência, aspirar objecto por objecto, desligando ao fim de cada objecto. A MN gosta de ver o "avião" rodar e clica com muita frequência. A MN quis brincar no jardim. Embora não controle o rato, a MN conseguiu perceber que tem que ligar e desligar o aspirador.			

Lista de elementos de registo (ficheiros) associados:
Observações

A MN continua a ter muitas dificuldades em segurar o rato e em clicar apenas para a execução de actividades específicas.
O J mal se sentou disse: "Eu quero o do helicóptero!"
Quando largou o aspirador sem querer a MN disse: olha, fugiu!

Session 6

Nome1	J – 3		
Nome2	MN – 3		
Hora inicial	11h05	**Hora final**	11h34
Objectivos prévios			
Trabalhar a varinha mágica e o aspirador para fazer mais imagens de Natal e para as aspirar. Guardar imagens no caderno.			
Desenrolar e resultados gerais:			
O J consegue usar bem a varinha mágica e o aspirador, sabe que tem que ligar e desligar. A MN clica sem saber bem para quê. Ainda não controla os movimentos com o rato de acordo com os objectivos. Acha muita graça ao passarinho que sai do ninho. A MN conseguiu perceber que tem que ligar e desligar o aspitrador, mas precisa de ajuda a coordenar os movimentos.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk e dsenho do Paint sobre o Natal			
Observações			

A MN e o J trabalham bem juntos. O João já tem alguma autonomia no ToonTalk, que a MN não tem.

Session 7

GRUPO: J e MN

IDADE: 3 anos

DATA:4-02-03

HORA DE INÍCIO: 11h00m

HORA DE TERMINUS: 11h26m

OBJECTIVOS PRÉVIOS: fazer caixa das trocas. Perceber a troca e conseguir fazer a troca.

DESENROLAR E RESULTADOS GERAIS: o J construiu a caixa das trocas sem ajuda e fez facilmente a troca. Avançamos para programação do robot a fazer troca. O J facilmente aprendeu a programar o robot, fazendo-o, depois, sem ajuda. Não se apercebeu que fica uma caixa igual à do pensamento e fez outra.

A MN percebeu a troca e conseguiu fazê- la sozinha.

OBSERVAÇÕES: a MN já consegue perceber e manipular os movimentos com o rato.

Session 8

GRUPO: J e MN

IDADE: 3 anos

DATA: 18- 02- 03

HORA DE INÍCIO: 10h25m

HORA DE TERMINUS: 10h43m

OBJECTIVOS PRÉVIOS: trocar imagens

DESENROLAR E RESULTADOS GERAIS: a MN já consegue fazer uma caixa dupla e colocar lá as imagens.

A MN não conseguiu fazer a troca logo de início. Por vezes coloca imagens em cima de outras pois não tem, ainda, a noção do tempo certo para largar as imagens onde quer.

A nível de destreza a MN já não precisa de ajuda. Já percebe como segurar as coisas.

Depois de algumas explicações, a MN acabou por conseguir fazer a troca.

OBSERVAÇÕES: o J faltou.

Session 9

GRUPO: J e MN

IDADE: 3 anos

DATA: 11-03-03

HORA DE INÍCIO: 15h30m

HORA DE TERMINUS: 15h54m

OBJECTIVOS PRÉVIOS: já enunciados

DESENROLAR E RESULTADOS GERAIS: para segurar os objectos o J teve tendência a arrastá-los, tal como se faz em outros jogos de CD-ROM.

Realizou a actividade sem lhe explicar os objectivos.

Eu- Porque é que as caixas estão iguais, João?

J - Porque assim ficavam giras.

Expliquei-lhes que para termos as caixas iguais tínhamos de colocar as imagens pela mesma ordem. O J explicou: "pois... passarinho/passarinho, rato/rato, flor amarela/flor amarela, flor rosa/ flor rosa, árvore/ árvore...".

A MN é autónoma a trabalhar mas custou- lhe perceber como criar duas caixas iguais.

O simples facto de haver imagens iguais parece satisfazê-los quanto à igualdade de caixas.

A MN por vezes coloca imagens sobre outras pois ainda não controla a movimentação em ambiente TOONTALK.

OBSERVAÇÕES: Deixei o J brincar um pouco livremente. Fez casas, usando a varinha mágica para copiar camiões já com caixas.

Session 10

NOMES: J e MN

IDADE: 3 anos

DATA: 27-03-03

HORA DE INÍCIO: 10h28m

HORA DE TERMINUS: 11h00m

OBJECTIVOS PRÉVIOS: trabalhar a troca e as competências no programa

DESENROLAR E RESULTADOS GERAIS: a MN está à - vontade a pegar e largar objectos no toontalk. Foi uma vitória!

O que é que o robô está a pensar? O que é o pensamento, J?

J - É aqui. - disse apontando para o pensamento do robô.

Eu - O que o ensinaste a fazer?

J - A trocar as coisas.

A MN não percebe a caixa no pensamento. Tenta pegar na caixa.

O J ao sair do pensamento do robô pergunta: "Dou - lhe esta caixa?" – disse apontando a caixa inicial.

Eu - Porque pára?

J – Porque já acabou de fazer isto.

O J diz que ensinou o "Hugo" a trocar as coisas de lugar. O J tentou dar uma caixa com vários buracos ao robô. Depois percebeu que só o ensinara com dois buracos e, como tal, assim não iria resultar.

OBSERVAÇÕES:

Session 11

NOMES: J e MN

IDADE: 3 anos

DATA: 20- 05-03

HORA DE INÍCIO: 14h55m

HORA DE TERMINUS: 15h20m

DESENROLAR E RESULTADOS GERAIS: o J desloca – se muito bem. A MN está muito faladora. Vai dizendo os nomes do que é preciso ao J.

Fizeram muitas casas, para dar uma para cada menino.

Eu – porque é preciso isso para as casas?

J – porque tem que se ir até à casa com o carro. A caixa serve para o robô se sentar e o robô serve para fazer a casa. O J queria pôr o desenho directamente na parede. A MN precisa de indicações para se movimentar.

OBSERVAÇÕES: para a MN construí com ela uma caixa tripla: parede, casa, desenho.

Foi uma grande conquista a MN já se movimentar sozinha no toontalk.

Session 12

NOMES: J e MN

IDADE: 3 anos

DATA: 03- 06-03

HORA DE INÍCIO: 11H00m

HORA DE TERMINUS: 11h30m

OBJECTIVOS PRÉVIOS: MN – colocar imagem na parede e fazer uma caixa dupla para trabalhar a troca; J – colocar nome no telhado e programar o robô para fazer o mesmo

DESENROLAR E RESULTADOS GERAIS: a MN não fala muito e por isso não percebo se ela entende o que faz. Quanto a juntar caixas ela já o faz percebendo o que está a fazer.

A MN já conseguiu perceber a troca, embora não explique, mas aponta e diz "olha, olha, olha!". Já tem autonomia na manipulação. O J depois de pôr o nome no telhado pediu para pôr o desenho na parede.

A ensinar o robô o J ficou um pouco confuso. Depois o J foi "dar um passeio"; ou seja, andou com o boneco pela zona verde.

A fazer casas o J usa a varinha mágica com destreza.

Group 4

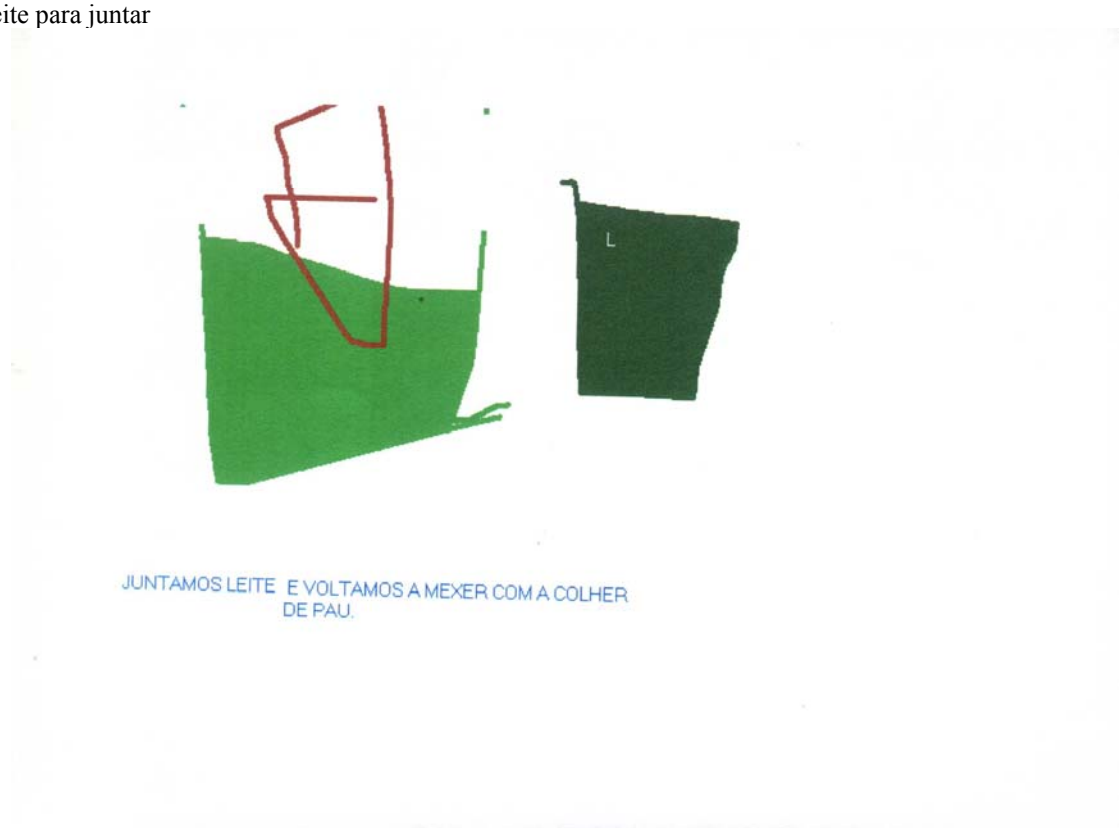
Session 1

Nome1	IN (4 anos)		
Nome2	C (4 anos)		
Hora inicial	10h05	**Hora final**	10h22
Objectivos prévios			
Identificar ambiente do ToonTalk. Apresentação da caixa de ferramentas.			
Desenrolar e resultados gerais:			
A IN: "É um avião". A C identificou as casas. Tiveram facilidade em sentar e levantar. Depois de lhes apresentar a caixa de ferramentas a IN quis tirar coisas da caixa e pousar no chão. A C também quis tirar coisas da caixa. A IN, por minha sugestão, usou o aspirador para limpar o chão.			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
A IN tornou-se uma criança muito comunicativa, isto é, comparativamente ao ano lectivo 2001/2002 em que a IN raramente se exprimia verbalmente.			

Session 2

Nome1	IN (4 anos)		
Nome2	C (4 anos)		
Hora inicial	10h22	**Hora final**	10h50
Objectivos prévios			
Explorar cores e ferramentas do Paint. Pegar e largar objectos do ToonTalk.			
Desenrolar e resultados gerais:			
A IN já controla um pouco melhor os movimentos do rato. No Paint consegue escolher bem as cores e tem a noção do "traço" no Paint. Quis desenhar uma casa. Quando começou a desenhar disse: "afinal é um barco". Depois quis desenhar o mar. Quando disse "levantar o boneco" ela levantou-se. A C não tem dificuldades em sentar e levantar. Quis desenhar uma casa, pediu-me ajuda.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint.			
Observações			

Session 3

Nome1	IN (4 anos)		
Nome2	C (4 anos)		
Hora inicial	10h00	**Hora final**	10h24
Objectivos prévios			
Desenvolver motricidade fina no Paint. Noções de "vazias" vs. "cheias". ToonTalk – noções de "dentro" e "fora". Manipular a mão no ambiente ToonTalk.			
Desenrolar e resultados gerais:			
A IN já não se lembrava de como levantar e aterrar. Andou um pouco perdida. Aterrou diversas vezes fora do espaço das casas. Precisou de ajuda para ir até às casas. Já consegue (IN) tirar e juntar caixas com destreza. A C trabalha bem com a IN. A C ainda tem dificuldade em tirar coisas da caixa de ferramentas.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint de registo da "tarte de maçã". IN – Mexer ingredientes C – Leite para juntar  			
Observações			
A IN chamou a atenção da C para não se distrair: "C olha pràqui." A C distraiu-se algumas vezes, prestando atenção ao que a educadora estava a dizer.			

Session 4

Nome1	IN – 4		
Nome2	C – 4		
Hora inicial	9:45	**Hora final**	10:15
Objectivos prévios			
Aprender a fazer o robô construtor de casas e testar a construção de casas. Trabalhar com a bomba de ar.			
Desenrolar e resultados gerais:			
A C começou por fazer uma caixa igual à que lhe preparei com objectos. A C já domina bem os movimentos no ambiente ToonTalk. Disse que o robô tem que levar a mala de viagem para ir para a nova casa. A C e a IN aprenderam a fazer casas novas e a ver se, de facto, existiam mais casas na cidade ToonTalk. C: "O camião é para pôr as malas." A IN ainda carrega/clica muitas vezes.			
Lista de elementos de registo (ficheiros) associados:			
Caixa de objectos no caderno ToonTalk. Casas na cidade.			
Observações			
A C e a IN cada vez que se chama a atenção ou surgem temas de maior ênfase que a educadora refere, perdem a concentração. Fizeram também o desenho de uma abóbora no Paint, a pedido da educadora, até porque estão a trabalhar abóboras no "dia das bruxas".			

Session 5

Nome1	IN – 4		
Nome2	C – 4		
Hora inicial	9:15	**Hora final**	9:46
Objectivos prévios			
Trabalhar o controlo do aspirador, com o objectivo de estabelecermos um paralelismo entre a arrumação da sala de jardim e da sala/quarto do ToonTalk.			
Desenrolar e resultados gerais:			
A IN ainda tem alguma dificuldade em conseguir entrar em casa. Senta-se algumas vezes sem a intenção de o fazer. A IN andou a aspirar o quarto mas custou um pouco a perceber que tinha de se sentar perto do "lixo" para o aspirar. Não desliga o aspirador sem a lembrar. A C também tem tendência a manter o aspirador ligado.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
Ao fim de 25 minutos a C comelou a ficar irrequieta na cadeira.			

Session 6

Nome1	I – 4 anos		
Nome2	C – 4 anos		
Hora inicial	10h12	**Hora final**	10h38
Objectivos prévios			
Trabalhar imagens de Natal para usar a bomba de ar e guardar imagens no caderno.			
Desenrolar e resultados gerais:			
A C é muito distraída. Não consegue usar o caderno de sensores sem ajuda. Ao trabalhar a bomba de ar não se lembra de desligar a bomba. A IN não consegue mudar as funções da bomba de ar. A IN já não se recorda de como levantar o boneco. A IN experimentou dar uma caixa a um passarinho e deu conta que ele "voou", mas não referiu que ele levou a caixa para o ninho.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk e desenhos (do Paint) de Natal.			
Observações			
A C pediu para se levantar antes da Iolanda terminar a actividade no ToonTalk.			

Session 7

GRUPO: IN e C

IDADE: 4 anos

DATA:4-02-03

HORA DE INÍCIO: 14h03m

HORA DE TERMINUS: 14h20m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado

DESENROLAR E RESULTADOS GERAIS: a C começou por mandar fazer uma casa nova, pois na cidade não havia mais casas sem nome no telhado. Tive de a ajudar. Depois voltou a fazer outra casa para colocar o nome.

Não teve dificuldades em fazer uma casa nova. Para escrever o nome no telhado queria fazê-lo directamente em cima do telhado.

Session 8

GRUPO: IN e C

IDADE: 4 anos

DATA: 13- 02- 03

HORA DE INÍCIO: 9h 45m

HORA DE TERMINUS: 10h06m

OBJECTIVOS PRÉVIOS: colocar desenho na parede e ensinar o robot a fazer o mesmo

DESENROLAR E RESULTADOS GERAIS: tive de ajudar a IN a encontrar o sensor de parede, apesar de ter simplificado o caderno de sensores.

O que fizemos/ precisamos para colocar o desenho na parede? Responderam: "cesta"… depois começaram a falar do desenho mas não fizeram referência à parede.

Para ensinar o robot não dizem que precisam de um robot, mas foram buscar um.

Não saem do pensamento do robot. Dizem: " acho que aprendeu!". Não verificam se o robot aprendeu o que lhe ensinaram.

OBSERVAÇÕES: no pensamento do robot: " estamos nas nuvens"- C

Perguntei porque é que o Robot parou depois de fazer o que lhe ensinaram. A resposta da C foi: " a caixa está vazia".

A IN não se lembra de como levantar o boneco.

Session 9

GRUPO: IN e C

IDADE: 4 anos

DATA:4-02-03

HORA DE INÍCIO: 14h03m

HORA DE TERMINUS: 14h20m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado

DESENROLAR E RESULTADOS GERAIS: a C começou por mandar fazer uma casa nova, pois na cidade não havia mais casas sem nome no telhado. Tive de a ajudar. Depois voltou a fazer outra casa para colocar o nome.

Não teve dificuldades em fazer uma casa nova. Para escrever o nome no telhado queria fazê-lo directamente em cima do telhado.

Session 10

NOMES: IN e C

IDADE: 4 anos

DATA:

HORA DE INÍCIO: 10h47m

HORA DE TERMINUS: 11h05m

OBJECTIVOS PRÉVIOS: 1º perceber e conseguir explicar o objectivo do jogo proposto.

DESENROLAR E RESULTADOS GERAIS: a IN aterrou no jardim. Já não se lembra como levantar o boneco.

Eu -Para que será este jogo?

IN -não sei.

Tive de lhes explicar. A C também começou por aterrar no jardim, mas disse que se tinha enganado.

OBSERVAÇÕES: Não avancei com programação do robô pois a nível de raciocínio a IN e a C ainda não me parece estarem preparadas, talvez daqui a duas sessões.

Session 11

NOMES: IN e C

IDADE: 4 anos

DATA: 27-03-03

HORA DE INÍCIO: 9h40m

HORA DE TERMINUS: 10h14m

OBJECTIVOS PRÉVIOS: Trocar a flor e a árvore de lugar e ensinar o robô.

DESENROLAR E RESULTADOS GERAIS: a C deu a caixa para ensinar o robô. Só não se lembrava de sair do pensamento do robô. Depois de sair diz: temos duas caixas PF!

Não ia testar se o robô sabia o que lhe ensinou. Ao dar ao robô uma caixa com a árvore e a flor e uma outra parte vazia, a C julgou ser igual.

A IN quando ensinou o robô disse: Já está. Fomos aspirar o pensamento para o robô trocar sempre. Aceitaram mas não percebem.

OBSERVAÇÕES:

Session 12

NOMES: I e C

IDADE: 4 anos

DATA: 22- 05-03

HORA DE INÍCIO: 9h14m

HORA DE TERMINUS: 9h44m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: a C precisou rever alguns comandos: levantar, sentar, bomba de ar.

Também já não se recordava como escrever o nome. Para colocar o nome na entrada de casa já conseguiu apontar que tinha de usar o caderno.

Não ia verificar se o nome estava mesmo na entrada.

A IN esteve a recordar como fazer casas o que já estava meio esquecido!

A IN não conseguiu explicar como pôr o nome na entrada de casa. Depois de pôr o nome não foi verificar. A C vai explicando.

Session 13

NOMES: IN e C

IDADE: 4 anos

DATA: 03- 06-03

HORA DE INÍCIO: 14H05m

HORA DE TERMINUS: 14h35m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado. Perceber o que é preciso para colocar o nome no telhado, colocando as coisas numa caixa.

DESENROLAR E RESULTADOS GERAIS: a C disse o que era preciso para pôr o nome no telhado. Depois resolvemos decorar a casa.

Reparou que o nome ficou enorme, ocupando 3 casas, o que eu calculo ter –se devido ao facto de ter colocado o nome sem tirar o telhado da caixa.

Expliquei e ela percebeu que tem que tirar o telhado da caixa para ver bem onde coloca o nome.

Experimentaram os três sensores da casa: telhado, parede e entrada.

A IN já sabe colocar o nome no telhado sem grandes dificuldades.

A IN confunde – se um bocado com os cadernos e parece –me que a ideia de colocar no "sensor pequeno" para aparecer no grande é confusa.

Group 5

Session 1

Nome1	L (4 anos)		
Nome2	MA (4 anos)		
Hora inicial	10h25	**Hora final**	10h45
Objectivos prévios			
Explorar o ambiente do ToonTalk. Noção de "tirar objectos para fora da caixa de ferramentas".			
Desenrolar e resultados gerais:			
A MA disse "É um avião". Identificaram as casas. A MA gostou especialmente dos ninhos, por causa dos passarinhos: "Oh PF, já tirei muitos!", disse com um sorriso após já ter tirado muitos ninhos da caixa de ferramentas. Depois usou o aspirador, por sugestão minha. Ao descobrirem a bomba/granada quiseram rebentar mais casas. Ao repararem que rebentando todas elas eram reconstruídas, quiseram rebentar mais casas.			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
O L após rebentar cada casa levantava voo para verificar que havia menos uma casa.			

Session 2

Nome1	L (4 anos)		
Nome2	MA (4 anos)		
Hora inicial	11h15	**Hora final**	11h40
Objectivos prévios			
Explorar ferramentas do ToonTalk e do Paint.			
Desenrolar e resultados gerais:			
A MA vai perguntando onde tem de carregar mas já controla bem os movimentos com o rato e reconhece os objectos da caixa de ferramentas do ToonTalk. No Paint já consegue traçar linhas, com precisão, com o pincel. Desenhou uma casa, uma flor e uma árvore. Pediu a minha ajuda para desenhar a árvore, o resto de desenho executou sozinha.			
Lista de elementos de registo (ficheiros) associados:			
Desenho do Paint.			
Observações			
O L faltou.			

Session 3

Nome1	L (4 anos)		
Nome2	MA (4 anos)		
Hora inicial	11h00	**Hora final**	11h35
Objectivos prévios			
Desenvolver a motricidade fina. Noções: "dentro vs. fora"; "vazia vs. cheia", utilizando as caixas do ToonTalk.			
Desenrolar e resultados gerais:			
A MA perguntou se era no rato que se carregava para baixar. Sentou-se fora de casa. O L carregou Esc para a MA entrar em casa e então se sentar. A MA riu e bateu palmas ao ver um passarinho sair do ninho. O L percebeu bem as noções "vazia" e "cheia" e já consegue tirar e pôr objectos na caixa.			
Lista de elementos de registo (ficheiros) associados:			
Caixas com coisas no caderno ToonTalk. MA – desenho da vestimenta de desfile de festa das tartes			
Observações			
11h15 – distraíram-se porque os colegas estavam a jogar o jogo das cadeiras e a fazer de gatos, em grande alvoroço.			

Session 4

Nome1	MA – 4		
Nome2	L – 4		
Hora inicial	15:30	**Hora final**	16.00
Objectivos prévios			
Explorar as ferramentas do Paint com o L. Trabalhar o tema "As abóboras do dia das Bruxas". Trabalhei a bomba de ar do ToonTalk.			
Desenrolar e resultados gerais:			
No Paint o L já consegue dominar melhor o traço. Desenhou uma abóbora perguntando: "Está a ficar bem?" Eu – sim, muito gira. O L já sabe utilizar bem o rato para levantar e aterrar no ToonTalk. Consegue aterrar e entrar nas casas. Com a bomba de ar ainda têm dificuldade em apontar a bomba e ligar e desligar, ao mesmo tempo. A MA trabalhou com uma flor. O L também. Ambos têm dificuldades em usar as duas mãos ao mesmo tempo.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
Antes de começar o L disse: vamos para o jogo do avião!			

Session 5

Nome1	MA – 4		
Nome2	L – 4		
Hora inicial	10:48	**Hora final**	11:15
Objectivos prévios			
Trabalhar o controlo com a barra de espaços. Aspirar para manter o espaço limpo.			
Desenrolar e resultados gerais:			
A MA gosta de experimentar o helicóptero mas não aterra perto das casas a não ser que lhe digam para o fazer. A ver o passarinho sair do ovo ambos se riem. Depois quiseram ver muitos pássaros a sair do ovo. Depois de tirarem coisas da caixa levantaram o boneco. Eu – "como está o chão?" MA – "Está desarrumado." L – "Está sujo." O L tem tendência a carregar muitas vezes no rato.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
O L perguntou: "Porque não vais buscar a coisinha cresce e abaixa, cresce e abaixa", referindo-se à bomba de ar. A MA chegou a cabeça ao ecrã e disse: "Ai, não se ouve nada." Para ir para outra casa, o L foi "a pé", acabou por se perder.			

Session 6

Nome1	MA – 4		
Nome2	L – 4		
Hora inicial	15h32	**Hora final**	15h56
Objectivos prévios			
Perceber que tal como "estamos" a decorar a sala de jardim com coisas sobre o Natal, podemos decorar uma casa do ToonTalk.			
Desenrolar e resultados gerais:			
A MA precisou de ajuda para encontrar o caderno de sensores. Depois, conseguiu encontrar a parede e decorá-la, o que lhe deu grande alegria, ao ver o Pai Natal que desenhou na parede. A MA levanta sempre o boneco para verificar o que fez. Foi preciso lembrar o L para verificar o que faz.			
Lista de elementos de registo (ficheiros) associados:			
Desenho do Paint e cidade ToonTalk.			
Observações			
Quiseram pôr uma árvore em frente da casa. Queriam uma árvore de Natal, mas como nas imagens só encontraram uma "árvore diferente", optaram por usá-la.			

Session 7

GRUPO: MA e L

IDADE: 4 anos

DATA: 4-02-03

HORA DE INÍCIO: 15h07m

HORA DE TERMINUS: 15h30m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado

DESENROLAR E RESULTADOS GERAIS: a MA precisa de indicações, não se lembrando como levanta o boneco. O L ajudou.

Na 2ª tentativa a MA voltou a precisar de ajuda. Para a MA as letras para o nome estão no teclado directamente, sem se lembrar de ir buscar as letras da caixa do toontalk.

OBSERVAÇÕES: a MA faz perguntas do género: " carrego aqui?". Ou diz: "não sei...", muitas vezes por preguiça e alguma insegurança.

Session 8

GRUPO: MA e L

IDADE: 4 anos

DATA: 18- 02- 03

HORA DE INÍCIO: 10h45m

HORA DE TERMINUS: 11h07m

OBJECTIVOS PRÉVIOS: colocar desenho na parede e ensinar o robot a fazer o mesmo.

DESENROLAR E RESULTADOS GERAIS: depois de repetir à MA como se coloca o desenho na parede, a MA não confirma se o desenho, de facto, está na parede grande.

Para levantar o boneco foi o L que a ajudou.

O L teve dificuldade em encontrar a parede para colocar o desenho, apesar de ter o caderno simplificado. Mas sabia colocar o desenho. Antes de entrar em casa disse: " vou espreitar à janela! Está aí alguém?". Fez de conta que espreitava.

OBSERVAÇÕES: a MA faz o sorriso de quem só faz quando lhe dão a certeza de estar a fazer bem.

Não ensinaram o robot pois ainda há muitas dúvidas relativamente a fazê-lo por eles próprios.

Session 9

NOMES: MA e L

IDADE: 4 anos

DATA:

HORA DE INÍCIO: 15h10m

HORA DE TERMINUS: 15h40m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: O L chamou avestruz ao pássaro.

A MA percebeu a intenção do jogo embora não se exprima bem ao tentar explicar ao colega.

Explicou a igualdade dizendo: "mete –se ali para ficar por baixo".

O L não percebeu a lógica do jogo, só depois de lhe explicar várias vezes é que conseguiu perceber. Mesmo assim, precisou que a MA o ajudasse.

A nível de destreza manual consegue trabalhar bem.

Para o L o jogo continuou a referir- se às noções de vazia e cheia e não a caixas que sejam iguais.

OBSERVAÇÕES:

Session 10

NOMES: MA e L

IDADE: 4 anos

DATA: 1-04-03

HORA DE INÍCIO: 14h32m

HORA DE TERMINUS: 14h46m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: perguntei à MA o que é a troca. Disse: é pôr aqui um e ali outro. - explicou apontando.

Eu – O que é que o robô está a pensar?

Resp.: A pensar que vai fazer outra vez o jogo.

Expliquei que ele só faz o jogo se lhe dermos uma caixa igual.

Eu - Porque parou?

Resp.: porque está a pensar noutra caixa.

Depois a MA experimentou trocar os objectos e voltou a testar o robô.

OBSERVAÇÕES: o L faltou

Session 11

NOMES: MA e L

IDADE: 4 anos

DATA: 22- 05-03

HORA DE INÍCIO: 9h46m

HORA DE TERMINUS: 10h06m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: o L sugeriu pegar no desenho e carregar no Space.

O L deu conta que a parede é igual à do caderno. A abstracção ainda não é feita. Só com indicação eles conseguem perceber que têm de usar uma parede "faz de conta". A MA sabe que para pôr o desenho numa parede precisa de uma parede mas mesmo com o caderno simplificado não a identificou. Depois esperei sem dizer nada, ela fez sozinha. Diz que vê que o desenho fica na parede porque está na pequena. Para pôr o nome a MA disse que precisava das letras. Não verifica se o nome fica na entrada da casa.

Session 12

NOMES: MA e L

IDADE: 4 anos

DATA: 05- 06-03

HORA DE INÍCIO: 10h30m

HORA DE TERMINUS: 11h00m

OBJECTIVOS PRÉVIOS: colocar nome no telhado. Perceber o que é preciso para colocar o nome no telhado, colocando as coisas numa caixa.

DESENROLAR E RESULTADOS GERAIS: Para o L é tudo complicado. Não vai verificar se o nome está no telhado pois diz que "está no pequeno, por isso está".

Depois desarrumou montes de coisas e começou a experimentar pegar em tudo. Tive de lhe dizer para aspirar as coisas e trabalhar com calma. Não sabe escrever o nome.

A MA já da indicações ao L de como fazer as coisas bem. Para escrever o nome só precisou que lhe apagasse o – A – para escrever.

Ao fazer a caixa com o que é preciso para pôr o nome no telhado largou o nome com o telhado ainda dentro da caixa.

Esquece como levantar o boneco e não verifica se o nome fica mesmo no telhado.

OBSERVAÇÕES: o L estava excitado porque tinham chegado à sala pintainhos que o S trouxe.

Group 6

Session 1

Nome1	U (5 anos)		
Nome2	CA (5 anos)		
Hora inicial	11h40	**Hora final**	12h04
Objectivos prévios			
Identificar ambiente do ToonTalk. Usar ferramentas. Noção de "dentro" vs. "fora", associada à ideia de "arrumar".			
Desenrolar e resultados gerais:			
Não tiveram dificuldades em aprender a sentar e levantar. Tiraram coisas da caixa de ferramentas e guardaram em caixas (que juntaram). Depois experimentaram tirar e tornar a meter. Brincaram com o robô construtor, usando também a varinha mágica para construir casas mais rapidamente. Depois, por sugestão minha, limparam o chão com o aspirador.			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
A CA já sabia usar o robô "construtor" pois o ToonTalk está instalado no computador da sala e as crianças costumam explorá-lo livremente, sem acompanhamento.			

Session 2

Nome1	CA (5 anos)		
Nome2	U (5 anos)		
Hora inicial	11h00	**Hora final**	11h36
Objectivos prévios			
Ver até que ponto a U e a CA estão autónomas no Paint. Explorar ambiente do ToonTalk.			
Desenrolar e resultados gerais:			
A U e a CA conseguiram descer com o helicóptero e manejar o boneco sem quaisquer dificuldades. A U quis desenhar uma casa, uma árvore e uma menina. Desenharam sem qualquer ajuda. A CA não sabia como levantar o boneco.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos no Pain.			
Observações			

Session 3

Nome1	CA (5 anos)		
Nome2	U (5 anos)		
Hora inicial	15h03	**Hora final**	15:25
Objectivos prévios			
Noções "dentro vs. fora", "vazia vs. cheia". Explorar ferramentas do ToonTalk.			
Desenrolar e resultados gerais:			
No Paint ambas revelam já dominar com alguma destreza os movimentos do rato. A CA já sabe levantar, sentar e movimentar o boneco sem dificuldades, embora às vezes peça ajuda. A U também já sabe usar o rato. Não se lembra muito bem dos comandos para levantar voo, aterrar e mesmo levantar. Questionam-me quando têm dúvidas.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint. Caixas no caderno do ToonTalk.			
Observações			
Gostam de trabalhar juntas e o que uma faz a outra quer fazer também.			

Session 4

Nome1	U -5		
Nome2	CA – 5		
Hora inicial	9:25	**Hora final**	10:08
Objectivos prévios			
Prepararem o jogo para os meninos mais novos utilizar a bomba de ar. Usar varinha mágica para fazer cópias.			
Desenrolar e resultados gerais:			
A U e a CA estão muito distraídas. Começaram a prestar mais atenção ao relembrarem os desenhos feitos no Paint. CA – "Eram 4 maçãs." No ToonTalk, a U para aterrar esteve cheia de "cuidados" para aterrar nas casas. A CA explicava: "para subir é no outro botão". A CA perguntou se usava Esca para ajoelhar.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint de 24-10-2002. Jogo no caderno ToonTalk.			
Observações			
Fases definidas da realização da tarte, pela U e CA: 1 – Ingredientes – 'O que precisamos... a farinha... e mais" 2 – Juntar farinha, açúcar, ovos e canela. 3 – "Mexemos tudo." 4 – "Untamos a forma com manteiga." 5 – "Deitamos a mistura da bacia na forma." 6 – "Pusémos a maçá, aos bocadinhos." 7 – Foi ao forno. 8 – "Ficou pronta para comer."			

Session 5

Nome1	U – 5		
Nome2	CA – 5		
Hora inicial	9:50	**Hora final**	10:15
Objectivos prévios			
Trabalhar o controlo "liga/desliga". A importância de mantermos os espaços arrumados e organizados. Aspirar e bomba de ar.			
Desenrolar e resultados gerais:			
Quiseram escrever o nome. A CA controla bem o "levantar/sentar". Como de costume, interroga-me sobre a boa execução de tarefas. Optei por lhe propor que descubra, ela mostrou já sabe, por exemplo "como se senta" (que foi uma das questões que levantou. A CA controla bem o "ligar/desligar" com o space. Com a bomba de ar conseguiu perceber que o G é igual a "grande" e o P = a "pequeno". Com a bomba de ar ainda confundem a localização da tecla para "ligar/desligar".			
Lista de elementos de registo (ficheiros) associados:			
Observações			

A CA descobriu que deve aspirar para que não fique "lixo" no chão.
A U ainda confunde um pouco o botão esquerdo do rato com o "space".
A CA explicou à U onde se carrega para o helicóptero levantar.

Session 6

Nome1	U – 5		
Nome2	CA – 5		
Hora inicial	9h49	**Hora final**	10h10
Objectivos prévios			
Trabalhar imagens de Natal para decoração.			
Desenrolar e resultados gerais:			
Ao ver o desenho, a CA quis guardá-lo no caderno. Não consegue procurar o telhado sem ajuda. A U também precisa de ajuda para encontrar a parede e trabalhar o caderno de sensores. Depois colocou a imagem do Pai Natal frente à casa, para parecer que estava a entrar em casa, mas precisou de ajuda mais uma vez, para encontrar o sensor da casa.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk e desenhos de Natal.			
Observações			
A U perguntou porque é que estamos sempre com o ToonTalk.			

Session 7

GRUPO: U e CA

IDADE: 5 anos

DATA:4-02-03

HORA DE INÍCIO: 11h30m

HORA DE TERMINUS: 11h45m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado e ensinar o robot a fazer o mesmo

DESENROLAR E RESULTADOS GERAIS: a U e a CA conseguiram colocar os seus nomes em telhados facilmente. A U ainda confunde alguns comandos do toontalk, como por exemplo, " como se sentar".

A CA e a U funcionam mesmo como uma equipa, ajudando- se mutuamente.

Ainda não entenderam bem como se ensina o robot. Não tiveram a iniciativa de testar o que lhe ensinaram.

OBSERVAÇÕES: a U e a CA ficaram agitadas quando se aperceberam que estavam a ser observadas por um grupo de pessoas que veio ver o nosso trabalho.

Session 8

GRUPO: U e CA

IDADE: 5 anos

DATA: 18- 02- 03

HORA DE INÍCIO: 9h20m

HORA DE TERMINUS: 9h48m

OBJECTIVOS PRÉVIOS: colocar desenho na moldura e ensinar o robot a fazer o mesmo.

DESENROLAR E RESULTADOS GERAIS: a U precisa de indicações da CA. Para colocar o desenho mais pequeno, a CA diz logo: "vai buscar a bomba".

A U já não sabe como colocar desenhos na parede.

A CA já se movimenta sem ser necessário dar-lhe indicações. Encontrou a moldura e colocou o desenho.

Para ensinar o robot, a CA sabia ir ao pensamento "temos de ir ao pensamento do robot." Aprendeu a ensiná- lo mas não verifica se ele aprendeu.

OBSERVAÇÕES: a CA disse que a mão que aparece no toontalk parece a mão de um macaco.

A U não ensinou o robot.

Session 9

NOMES: CA e U

IDADE: 5 anos

DATA:

HORA DE INÍCIO: 9h10

HORA DE TERMINUS: 9h35m

OBJECTIVOS PRÉVIOS: já enunciados

DESENROLAR E RESULTADOS GERAIS: a CA e a U exlicaram: "pôr igualzinho".

Para fazer o robô construtor:

CA –Tenho de ir ao pensamento.

Disse que era preciso um camião e uma caixa. Expliquei que também precisavam de um robô para ser o construtor.

Para verificar a CA disse que ia "lá fora".

Tive de lhes dizer, quando ensinaram o robô que para verse ele aprendeu tínhamos de lhe dar uma caixa igual à do pensamento dele.

Não usaram a caixa inicial.

Eu -Porque é ele parou?

U – Porque já fez a casa e a caixa está vazia.

A U precisou que a lembrasse de dar uma caixa com o necessário para ensinar o robô. A U não ia usar a caixa inicial.

Para guardar coisas no caderno sabiam que tinham de passar as páginas no mais.

OBSERVAÇÕES:

Session 10

NOMES: C e U

IDADE: 5 anos

DATA: 1- 04-03

HORA DE INÍCIO: 11h38m

HORA DE TERMINUS: 11h58m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: perguntei o que era a troca. Responderam: é chegar ali e trocar a árvore e a flor.

A U lembrava- se de usar a varinha mágica no pensamento. Perguntei porque estava a usar a varinha mágica. A CA respondeu que era para o robô fazer sempre igual.

A CA ao testar o robô não usa a caixa inicial.

Perguntei porque é que o robô que ensinaram não parava. A U respondeu que lhe tinha ensinado truques de magia.

Por já estarmos na Primavera estivemos ainda a decorar a entrada da casa com flores, o que as estimulou.

OBSERVAÇÕES:

Session 11

NOMES: AL e CA

IDADE: 5 anos

DATA: 20- 05-03

HORA DE INÍCIO: 15h25m

HORA DE TERMINUS: 15h49m

OBJECTIVOS PRÉVIOS:

DESENROLAR E RESULTADOS GERAIS: a AL já não se recordava como decorar a parede.

Estiveram a pôr o nome na casa.

A CA relembrou a AL onde levantava o boneco.

A CA ao guardar o que fez no caderno quis colocar de um lado o que fez e do outro o nome, como costumamos fazer.

A AL e a CA conseguiram perceber a actividade.

OBSERVAÇÕES: como a U está para a Ucrânia, a CA fez com a AL

Session 12

NOMES: U e CA

IDADE: 5 anos

DATA: 03- 06-03

HORA DE INÍCIO: 9h30m

HORA DE TERMINUS: 10h07m

DESENROLAR E RESULTADOS GERAIS: a CA disse logo que precisava de letras para fazer o nome. Ficou confusa quando pensou em pôr o nome no telhado e foi à rua. Depois expliquei que pomos sempre no pequeno para aparecer no grande. Fez sem ajuda e disse: agora vamos ver se resultou.

CA – "Para ensinar o robô preciso de uma caixa para pôr lá dentro o que preciso."

A U perguntou quando íamos trabalhar só outros jogos, que está cansada do toontalk.

A U gosta de mandar construir casas.

Tornou – se mais fácil para a U criar caixa com as coisas para ensinar o robô.

Group 7

Session 1

Nome1	CR (4 anos)		
Nome2	JG (4 anos)		
Hora inicial	10h58	**Hora final**	11h19
Objectivos prévios			
Aferir a destreza adquirida no ano 2001/2002 em termos de motricidade fina e de capacidade de explorar um jogo.			
Desenrolar e resultados gerais:			
Identificaram o helicóptero e as casas. O CR fez um robô para construção de casas. Depois fez mais 3. Já sabem levantar/sentar sem dificuldades. CR: "Vou fazer muitas casas." A JG usou a varinha mágica: "ah! Já sei! Mais camiões!" Usaram a bomba/granada. Eu disse-lhes: "vamos rebentar as casas todas?" Eles: "sim, porque depois algumas fazem-se!" Tiraram desenhos do caderno e arrumaram em caixas. Ao pegar no foguetão a J disse: "Temos de pôr isto mais pequeno!" E foi buscar a bomba de ar. A J fez uma "caixa de coisas" e guardou no caderno dela.			
Lista de elementos de registo (ficheiros) associados:			
J – "caixa de coisas" no caderno.			
Observações			
A J já sabe usar/brincar com o ToonTalk sem dificuldades, até porque tem o jogo em casa. Sem qualquer referência usou F1 para mandar o marciano embora. O CR também já sabia fazer o robô "construtor".			

Session 2

Nome1	CR (4 anos)		
Nome2	JG (4 anos)		
Hora inicial	9h41	**Hora final**	10h03
Objectivos prévios			
Aferir os conhecimentos da J no ToonTalk.			
Desenrolar e resultados gerais:			
A J não teve quaisquer dificuldades em aterrar, entrar na casa e pegar no caderno. No Paint trabalha com destreza as diferentes ferramentas. Inclusive sabe usar bem as formas geométricas. Fez o desenho de uma menina, identificando cabeça, tronco e membros. Apenas teve dificuldade em desenhar mãos, dizendo que estava a desenhar "esquisito, só com 3 dedos". Depois acrescentou 2 dedos.			
Lista de elementos de registo (ficheiros) associados:			
Desenho do Paint.			
Observações			
O CR faltou.			

Session 3

Nome1	CR (4 anos)		
Nome2	JG (4 anos)		
Hora inicial	15h54	**Hora final**	16h08
Objectivos prévios			
Explorar o ambiente do ToonTalk. Noções "dentro vs. fora", "vazia vs. cheia".			
Desenrolar e resultados gerais:			
O CR já conhece os comandos básicos do ToonTalk para voar, sentar, levantar. Eu – "Se juntarmos caixas..." CR – "Eu sei, eu sei!... O rato vem e junta!" Percebeu bem as noções "dentro e fora" da caixa, "vazia e cheia".			
Lista de elementos de registo (ficheiros) associados:			
Caixa no caderno do ToonTalk: "caixa truca".			
Observações			
A JG faltou. O CR não fez o desenho no Paint por falta de tempo.			

Session 4

Nome1	CR – 5		
Nome2	JG – 5		
Hora inicial	11:30	**Hora final**	12:00
Objectivos prévios			
Colocar nome nos telhados das casas. Ensinar o robô a fazer robôs construtores de casas.			
Desenrolar e resultados gerais:			
Ambos já sabem fazer o robô construtor de casas sem dificuldade. Eu: "O que é que podemos fazer para sabermos de quem são as casa?" JG – "As minhas são as cor-de-rosa." Eu: "E que tal pormos o nome no telhado?" Eles – "Boa!" A JG aprendeu à 1ª a pôr o nome no telhado. Ao ver o resultado começou a bater os pés de alegria e disse: "vou faer outra!".			
Lista de elementos de registo (ficheiros) associados:			
Cidade com nomes nos telhados.			
Observações			
Não tivémos tempo de ensinar o robô a fazer robôs construtores de casas. O CR também aprendeu com facilidade mas tem alguma ansiedade o que lhe dificulta a execução das tarefas.			

Session 5

Nome1	CR – 5		
Nome2	JG – 5		
Hora inicial	10:44	**Hora final**	11:10
Objectivos prévios			
Trabalhar o controlo do aspirador. Trabalhar a arrumação da casa.			
Desenrolar e resultados gerais:			
A JG mal entrou em casa começou a trabalhar o caderno, para fazer decoração do telhado. Quando a JG ao aspirar limpou o nome que pôs no telhado, queixou-se ao sair, por não ver o nome. O CR: "então, tu engoliste-o! Engoliste-o com o teu nariz!". O CR sentou-se uma vez no jardim, a JG comentou: "Oh CR, sentaste-te no jardim, que tu saibas!!".			
Lista de elementos de registo (ficheiros) associados:			
Observações			
O CR está a trabalhar muito bem com a JG. Quando o CR pegou no aspirador, alterou a sua função. Ao cuspir disse a JG: "O CR fez o aspirador a cuspir, que nojo!!". Querem decorar o avião!!			

Session 6

Nome1	JG – 5		
Nome2	CR – 5		
Hora inicial	11h26	**Hora final**	11h53
Objectivos prévios			
Trabalhar imagens do Natal para decoração de casas no ToonTalk.			
Desenrolar e resultados gerais:			
A JG achou melhor reduzir o desenho para o pôr na parede. Depois resolvemos fazer uma casa que levasse com um ninho e experimentou dar um desenho ao pássaro. Perguntou como sabia qual era a casa nova. Expliquei que seria a que estava a mais – que a situação inicial. Depois decorámos o telhado para podermos identificar a casa. O CR também conseguiu perceber que o passarinho leva tudo para o ninho e decorou o telhado com o desenho de Natal.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk, com os passarinhos com os nomes. Desenhos de Natal – Paint.			
Observações			
A intenção é que na próxima semana possamos trocar prendas. A JG quiz pôr um passarinho a viver na casa, para saber que era a casa dela. Depois disse: "Mas se eu pusesse o passarinho maior, sabia melhor que era a minha casa!"			

Session 7

GRUPO: CR e JG

IDADE: 5 anos

DATA: 4-02-03

HORA DE INÍCIO: 11h48m

HORA DE TERMINUS: 12h00m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado. Ensinar o robot a colocar o nome no telhado.

DESENROLAR E RESULTADOS GERAIS: a JG tem muito à- vontade no toontalk. A JG, após ter escrito o nome no telhado, assume isso como um dado e não vai verificar.

Ao ensinar o robot a colocar o nome no telhado também não verifica se ele aprendeu o que lhe ensinou. Expliquei-lhe que eu também só sei que ela sabe algo que lhe ensinei se a vir a fazê-lo.

OBSERVAÇÕES: a JG diz: " Eu já sei tudo". Só depois de lhe mostrar que pode ainda haver coisas que não sabe é que ela dá conta que ainda há muito a aprender e "inventar".

O CR faltou.

Session 8

GRUPO: CR e JG

IDADE: 5 anos

DATA: 18- 02- 03

HORA DE INÍCIO: 11h10m

HORA DE TERMINUS: 11h34m

OBJECTIVOS PRÉVIOS: colocar desenho na moldura e ensinar o robot a fazer o mesmo.

DESENROLAR E RESULTADOS GERAIS: Eu - " O que precisas para pôr o desenho mais pequeno?"

CR - " Do gato."

Eu -" Como vais pôr o desenho na parede?"

CR -"Pegar nele e levantar -me!"

Depois de colocar o desenho no sensor de parede não se levanta para verificar.

Ia sair da casa com tudo desarrumado. Quando entrou no pensamento do robot perguntei -lhe onde estávamos, ele respondeu: " no céu".

Perguntei- lhe o que era preciso para ensinar o robot a construir casas. A resposta foi: "duma caixa, de um camião e de um robot".

OBSERVAÇÕES: a JG faltou

Session 9

NOMES: JG e CR

IDADE: 5 anos

DATA:

HORA DE INÍCIO: 11h40m

HORA DE TERMINUS: 12h04m

OBJECTIVOS PRÉVIOS: objectivo geral já referido

DESENROLAR E RESULTADOS GERAIS: quando perguntei qual seria o objectivo do jogo a JG disse: "É para pôr em baixo as figuras que estão em cima."

Ambos fizeram o jogo sem dificuldade, como tal, avançamos para a programação do robô.

Eu -Como é que programo robô?

JG – Vou para o pensamento.

Eu – Para saber se o robô aprendeu, como faço?

R: Saio do pensamento do robô.

A JG usa a caixa inicial para testar o robô.

Como copiaram a caixa, perguntaram se tinham de esperar que o robô fizesse sempre igual. Expliquei que podem parar o robô.

O CR não se lembrava como ir para o pensamento do robô.

O CR ensinou o robot a construir uma casa pois não conseguiu ensinar o jogo ao robô.

Depois de sair do pensamento do robô perguntei ao CR: O que é que o robô está a pensar?

R: Tá a pensar fazer o que lhe ensinei.

OBSERVAÇÕES:

Session 10

NOMES: JG e CR

IDADE: 5 anos

DATA: 1- 04-03

HORA DE INÍCIO: 13h48m

HORA DE TERMINUS: 14h10m

OBJECTIVOS PRÉVIOS: Explicar a troca. Ensinar o robô a fazer a troca e ensiná – lo de forma a não parar.

DESENROLAR E RESULTADOS GERAIS: O CR ficou atrapalhado, de vez em quando, com o rato e desatou a abaná-lo.

O que esta a pensar o robô?

CR – Está a pensar fazer o que lhe ensinei.

O CR consegue ensinar bem o robô mas não se recorda como sair do pensamento.

Rebentou a casa ao ver o rapaz zonzo e disse que ele tinha um mosquito à volta da cabeça. Quando rebentou as casas, indo à última disse: Esta é a que se monta?

Eu – Como é isso?

CR – Os homens vêm e fazem –nas.

OBSERVAÇÕES: A JG faltou.

Session 11

NOMES: JG e CR

IDADE: 5 anos

DATA: 22- 05-03

HORA DE INÍCIO: 11h20m

HORA DE TERMINUS: 11h35m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: a JG tem uma capacidade de abstracção muito desenvolvida. Também é bom não esquecer que ela tem acesso ao programa em casa!

Quase se esquecia de verificar se o nome fica mesmo na entrada ou o desenho na parede. Acabou por dizer: tenho de ir ver".

Já sabe que para levar um objecto no bolso tem de o segurar.

Avançamos para a programação do robô.

Comecei por questionar o que seria necessário para ensinar o robô a colocar o desenho na parede.

JG – Preciso do desenho e da parede.

Pusemos as duas coisas na caixa para ensinar o robô. A JG ensinou – o sem a minha ajuda.

Identificou: "o pensamento do robô é aquela parte branca".

Depois disse que já estava cansada e pediu para parar.

OBSERVAÇÕES: o CR faltou.

Session 12

NOMES: JG e CR

IDADE: 5 anos

DATA: 05- 06-03

HORA DE INÍCIO: 9h52m

HORA DE TERMINUS: 10h 16m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado e programar o robô para fazer o mesmo

DESENROLAR E RESULTADOS GERAIS: o facto de lhes ter dado o caderno de sensores simplificado facilitou a tarefa e a JG não precisou de quaisquer indicações.

Eu –E agora?

JG – Agora vou ver se ficou.

Antes de ir ver se de facto tinha colocado o nome no telhado guardou o que fez no caderno.

A JG já sabe como ensinar o robô mas ia dar -lhe o telhado directamente. Depois lembrou –se como fazer, que tinha de lhe dar a caixa com as coisas. Identifica o robô do pensamento: agora vou fazer eu e depois é que faz ele.

Usou a caixa inicial para testar o que ensinou ao robô. Não ia verificar se o nome ficou no telhado pois, como ele faz o que ela lhe ensinou, acreditou estar tudo em ordem.

O CR tem um ritmo mais moroso mas também atingiu os objectivos propostos. Contudo, não vai verificar se o nome fica no telhado.

O CR atrapalha –se um pouco com a movimentação do rato. Não sai do pensamento do robô sem que eu o lembre. Verifica se o robô aprendeu o que lhe ensinou, mas também não vai verificar se o nome ficou no telhado.

Group 8

Session 1

Nome1	IM (5 anos)		
Nome2	S (5 anos)		
Hora inicial	14h25	**Hora final**	14h48
Objectivos prévios			
Trabalhar a capacidade de concentração com a IM. O S poderá aprender a ser mais paciente, uma vez que é bastante ansioso.			
Desenrolar e resultados gerais:			
Identificaram o "avião". O S já sabe fazer o "robô construtor". Os dois fizeram muitas casas e foram vê-las. A IM usou o rato pela 1ª vez pelo que, precisou da minha ajuda para coordenar movimentos. 14h42-Os colegas foram para os cantos e a IM quis ir para a casinha: "Não quero jogar mais, quero a casinha".			
Lista de elementos de registo (ficheiros) associados:			
-			
Observações			
A IM tem necessidades educativas. Esteve a trabalhar concentrada mas com o alvoroço repentino dos colegas (porque a educadora mandou) perdeu a concentração. O S quis continuar para trabalhar com o aspirador e rebentar uma casa.			

Session 2

Nome1	IM (5 anos)		
Nome2	S (5 anos)		
Hora inicial	15h20	**Hora final**	15h52
Objectivos prévios			
Trabalhar a concentração – com a IM. Trabalhar o relacionamento em trabalho com o S.			
Desenrolar e resultados gerais:			
O S entrou bem no jogo e conseguiu segurar bem o caderno. Também sabe utilizar as ferramentas do Paint. A IM gostou de desenhar no Paint. Ajudei-a, pois foi o 1º contacto que teve com o Paint e, como tal, de início estava "Assustada" por não conseguir. A IM gosta de ouvir os sons do ToonTalk. Ficou a virar páginas do caderno para ouvir o som. Treinou o pegar e largar coisas. Quanto ao desenho no Paint, a satisfação foi notória. Já de volta ao ToonTalk disse: "O meu desenho está giro!" 15h49 - A IM perdeu a concentração pois a educadora mandou arrumar, aos outros meninos.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
O S tem já algumas mudanças no relacionamento, uma vez que tem que ser paciente e procurar ajudar a IM e não recriminá-la, ou seja, aceitar a diferença.			

Session 3

Nome1	S (5 anos)		
Nome2	IM (5 anos)		
Hora inicial	9h10	**Hora final**	9h51
Objectivos prévios			
Colocar o boneco dentro de casa sentado. Compor caixas e colocar coisas dentro. Tirar e voltar a colocar. Noções "dentro" e "fora".			
Desenrolar e resultados gerais:			
A IM pousou o helicóptero sem estar preocupada se estava a pousar nas casas. Consegue mover bem o boneco. Chamou vela à bomba: "parecia que era uma vela!" 9h40 – A IM começou a ficar impaciente. 9h41 – "Não quero mais, estou cansada." Eu – "E um desenho?" Resposta: "Está bem, só um desenho."			
Lista de elementos de registo (ficheiros) associados:			
Registo da tarde de maçã. S – 1ª etapa (ingredientes) (Paint.) IM – 2ª etapa (juntar ingredientes)			
Observações			
O S se o deixarem fica indefinidamente no "contador"!! O S já experimentou dar coisas aos passarinhos e viu que eles as levam para os ninhos: "levam os números".			

Session 4

Nome1	IM – 5		
Nome2	S – 5		
Hora inicial	14:06	**Hora final**	14:40
Objectivos prévios			
Deixar explorar o ToonTalk para encontrar interesses. Trabalhar a concentração e motricidade fina com a IM. Aprender a colocar nomes no telhado.			
Desenrolar e resultados gerais:			
"Vamos aprender a saber quais são as casas do S e as da IM?" IM (animada): "Vamos, tá bem!". O S perguntou o que era as luzes que dão do lado da "casa"/telhado. Respondi que eram as luzes que saiem da casa. O S aprendeu a pôr o nome no telhado e sai da casa e levanta vôo para verificar o nome no telhado. A IM mostrou muito interesse em trabalhar. (A IM manipula os objectos da caixa de ferramentas.)			
Lista de elementos de registo (ficheiros) associados:			
Cidade com nomes, S e IM.			
Observações			
A IM disse que tinha de estar no computador: "Tenho de trabalhar muito para aprender." A IM gosta de trabalhar, experimentar e tentar fazer as coisas sozinha. Tive de lhe "pedir" para a ajudar porque ela dizia "não consigo" mas evitava ajuda.			

Session 5

Nome1	IM – 5		
Nome2	S – 5		
Hora inicial	14:15	**Hora final**	14:46
Objectivos prévios			
Trabalhar o controlo do aspirador englobando a noção de "limpeza". Trabalhar a concentração com a IM.			
Desenrolar e resultados gerais:			
O S "desarrumou" a casa, depois levantou-se para ver o quanto desarrumou. Depois esteve a aspirar. Não tem quaisquer dificuldades e gosta de estar no computador o tempo inteiro (se pudesse!). A IM evita que eu a ajude. De vez em quando fala com o boneco: "vai, entra!". A IM não associa o "limpa-o-pó" com um aspirador.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk.			
Observações			
O S gosta de explorar. Aprendeu a decorar a casa por for, pois queria fazer um jardim. No computador, a IM é muito observadora. Esteve muito calma, sossegada e interessada em trabalhar.			

Session 6

Nome1	IM – 5		
Nome2	S – 5		
Hora inicial	14h10	**Hora final**	14h35
Objectivos prévios			
Trabalhar imagens de Natal para decoração na cidade ToonTalk.			
Desenrolar e resultados gerais:			
O S consegue utilizar bem os sensores mas tem dificuldade em encontrar o caderno de sensores. Também não consegue alterar as funções do aspirador e da bomba de ar. Depois de fazer a actividade de decoração, o S quis fazer casas e para isso, livremente, utilizou sem dificuldade a varinha mágica. A IM ao folhear o caderno distrai-se com o som e não olha para procurar a parede. Conseguiu usar a varinha mágica.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk.			
Observações			
As decorações feitas nas casas do ToonTalk vêm de encontro à decoração que está a ser feita na sala de jardim.			

Session 7

NOME: Stefano

IDADE: 5 anos

DATA: 23-01-03

HORA DE INÍCIO: 10h09

HORA DE TERMINUS: 10h28

OBJECTIVOS PRÉVIOS: realizar uma cidade em que as casas tivessem as fotos das crianças

DESENROLAR E RESULTADOS GERAIS: o Stefano visualiza facilmente as fotos que deseja. No Toontalk tive de lhe explicar o que é uma moldura. Depois conseguiu colocar a foto na moldura sem ajuda e conseguiu encontrar a parede/ sensor. Teve ajuda para encontrar o caderno de sensores.

Session 8

GRUPO: S e IM

IDADE: 5 anos

DATA: 13- 02- 03

HORA DE INÍCIO: 15h40m

HORA DE TERMINUS: 16h00m

OBJECTIVOS PRÉVIOS: colocar desenhos da visita à azenha na parede

DESENROLAR E RESULTADOS GERAIS: o S fez a actividade sem dificuldade. Lembrou - se de como ensinar o robot, dando uma caixa dupla comas coisas necessárias ao robot. A IM conseguiu colocar o desenho na parede com dicas que o colega lhe foi dando. Não trabalhou a programação do robot pois ainda lhe é muito confusa a abstracção.

OBSERVAÇÕES: a IM já tinha a mãe à espera mas quis ficar no computador a acabar a actividade.

Session 9

GRUPO: IM e S

IDADE: 5 anos

DATA: 4-02-03

HORA DE INÍCIO: 14h 22m

HORA DE TERMINUS: 14h40m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado.

DESENROLAR E RESULTADOS GERAIS: o S, depois de ensinar o robot, lembrou-se de usar a caixa que criara para ser mais rápido. A IM conseguiu pôr o nome no telhado. Para programar o robot precisou de ajuda. Não se referiu a verificar o que ensinara ao robot.

OBSERVAÇÕES: a IM ficou sorridente por ir "ter uma casa".

Session 10

NOMES: S e IM

IDADE: 5 anos

DATA:

HORA DE INÍCIO: 11h05m

HORA DE TERMINUS: 11h34m

OBJECTIVOS PRÉVIOS:

DESENROLAR E RESULTADOS GERAIS: foi a IM que explicou a finalidade do jogo. Disse: é para pôr igual.

Dando a caixa ao robô o S disse: vai para o pensamento. Foi ao pensamento e lá ensinou a fazer a casa e começou a brincar com a caixa no pensamento.

A IM disse que o robô, quando faz o que lhe ensinamos, pára porque a caixa fica vazia.

A IM também não saia do pensamento do robô. O S ajuda a IM. Para voltar a fazer outra caixa igual para o robô trabalhar a IM ia usar o próprio robô que ensinara.

OBSERVAÇÕES: A IM sobre o jogo de sequência disse: "este jogo está muito giro".

Session 11

NOMES: IM e S

IDADE: 5 anos

DATA: 1- 04-03

HORA DE INÍCIO: 14h16m

HORA DE TERMINUS: 14h45m

OBJECTIVOS PRÉVIOS: trabalhar a troca

DESENROLAR E RESULTADOS GERAIS: o S explicou a troca dizendo: é pôr a árvore onde estava a flor e a flor onde estava a árvore.

Ao ensinarem o robô a IM não o testa, diz "tá ensinado".

O S não sai do pensamento do robô.

O S, para além de lhe ensinar a troca, ensinou o robô a folhear o caderno.

O S e a IM estiveram a aprender a copiar a caixa para que o robô não pare.

IM - Faz igual porque o ensinei a fazer. – e continua a explicar a meu pedido – Parou porque queria parar. Porque só troca como lhe ensinei.

O S ao ver a árvore ficar vermelha diz: está a arder!

A IM esteve a treinar o uso da varinha mágica para fazer mais flores.

O S esteve a decorar a casa com flores. Tive de o ajudar a encontrar o caderno de imagens.

OBSERVAÇÕES: só acabou a sessão a 1 de Abril.

Session 12

NOMES: IM e S

IDADE: 5 anos

DATA: 19- 05-03

HORA DE INÍCIO: 14h40m

HORA DE TERMINUS: 15h10m

DESENROLAR E RESULTADOS GERAIS: a IM não aterrou junto à casa. O S lembrou para ir para casa.

A IM procurou pôr o desenho directamente na parede.

O S disse – lhe para ir ao livro buscar a "parede faz de conta".

A IM não percebe a parede faz de conta, apesar de eu ter simplificado o caderno. Deixei a IM explorar livremente o toontalk porque a IM estava a dizer que já queria sair do computador. Notei que isso se devia ao facto de não perceber o que lhe foi pedido.

A IM fez uma caixa com três coisas: um camião, um ninho e o número 2. disse que era para dar ao robô. Depois não chegou a da - la ao robô e quis pôr o nome dela na casa.

A IM esteve a aspirar a casa. Ás 15h a IM pediu se podia ir embora. O S continuou. Relembraram como usar a bomba de ar e como pôr o nome. O S até já sabe usar a tecla para começar a escrever o nome.

Session 13

NOMES: IM e S

IDADE: 5 anos

DATA: 03- 06-03

HORA DE INÍCIO: 10h35m

HORA DE TERMINUS: 10h55m

OBJECTIVOS PRÉVIOS: para a IM - colocar o nome no telhado; para o S – colocar o nome no telhado e ensinar o robô a fazer o mesmo.

DESENROLAR E RESULTADOS GERAIS: a IM faz as coisas muito atrapalhadamente. Às vezes faz muito bem mas não explica o porque de o ter feito. A IM não tira o telhado da caixa e mete o nome directamente no telhado. Já sabe levantar o boneco sem precisar que lhe indique onde carregar. Ao mandar fazer mais casas, o S não tem quaisquer problemas em usar a varinha mágica para fazer muitos camiões e caixas. Para ensinar o robô o S disse que tinha de fazer igual. Ensinou bem o robô. Deu a caixa construída inicialmente para testar o robô, mas não verificou se o nome ficou no telhado.

Group 9

Session 1

Nome1	AM (5 anos)		
Nome2			
Hora inicial	14h50	**Hora final**	15h05
Objectivos prévios			
Apresentação e contacto com o "rato". Explorar e conhecer o ambiente do ToonTalk.			
Desenrolar e resultados gerais:			
Identificar o avião. Eu: "Por baixo do avião está o quê, AM?" AM: "Riscos..." Não teve dificuldade em aprender a sentar e levantar. Fez 2 pássaros gémeos a que deu o nome "os manos pássaros", depois ajudei-a a guardar a caixa com os dois pássaros no caderno.			
Lista de elementos de registo (ficheiros) associados:			
Caderno – caixa com os "manos pássaros"			
Observações			
A AM brincou pela 1ª vez no computador. É irmã gémea da IM mas quis separá-las pois a AM, noutros espaços, vive as "frustrações" que a irmã não sente! Fica envergonhada com as "não realizações da irmã". Como tal, achei que seria benéfico trabalharem separadas.			

Session 2

Nome1	AM (5 anos)		
Nome2	V (5 anos)		
Hora inicial	14h45	**Hora final**	15h15
Objectivos prévios			
Explorar caixa do ToonTalk. Colocar o V como forte colaborador da AM.			
Desenrolar e resultados gerais:			
Não tem dificuldades a sentar o boneco. O V desenhou uma cara de homem. Depois apagou a cara e fez um barco com uns salva-vidas e peixinhos no mar. Quando quisemos sair do ToonTalk disse-lhe "Vamos levantar?" e o V levantou-se da cadeira. A AM precisou de ajuda no Paint, pois ainda não domina os movimentos com o rato. Foi a 1ª vez que a AM trabalhou no Paint.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint.			
Observações			
Como foi a 1ª vez que a AM desenhou no Paint, ajudei-a para evitar a "frustração". No ToonTalk ainda tem dificuldade em manipular o rato e usar ferramentas, não conseguindo fazê-lo sozinha.			

Session 3

Nome1	AM		
Nome2	V		
Hora inicial	15h30	**Hora final**	15h52
Objectivos prévios			
Trabalhar as ferramentas do ToonTalk com a AM, principalmente as noções básicas; isto é, sentar, levantar, pegar e largar objectos. Noções "dentro e fora", "cheia e vazia".			
Desenrolar e resultados gerais:			
O V é exigente consigo mesmo. Quando não consegue fazer as coisas como esperava diz sempre: "Oh, enganei-me" e quer repetir. A AM gosta de brincar com o avião. Não tem noções de "levantar/aterrar". Não identifica as casas.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos do Paint. V – caixa no caderno do ToonTalk			
Observações			
O V está a melhorar a sua capacidade de expressar.			

Session 4

Nome1	AM		
Nome2	V		
Hora inicial	14:42	**Hora final**	15:12
Objectivos prévios			
Desenvolver a capacidade de desenho no Paint, com a AM. Aprender a colocar o nome nos telhados da cidade ToonTalk.			
Desenrolar e resultados gerais:			
O V mal entrou na casa do ToonTalk: "Vou rebentar a casa". Contudo, esqueceu-se que para rebentar a casa tinha de segurar a bomba na mão. Eu "O que podemos fazer para saber de quem são as casas?" V – "Podemos dar dinheiro ao senhor e ficamos lá!" Expliquei que também nos desenhos é o nome que nos permite saber quem os fez. A AM só consegue pôr o nome no telhado com ajuda.			
Lista de elementos de registo (ficheiros) associados:			
Desenhos sobre as abóboras – no Paint. Cidade ToonTalk.			
Observações			
O V hoje está um bocado impaciente, concentrou-se quando descobriu como pôr o nome no telhado. Disse que queria pôr o nome numa "casa de férias":			

Session 5

Nome1	AM – 5		
Nome2	V - 5		
Hora inicial	11:34	**Hora final**	11:59
Objectivos prévios			
Trabalhar o controlo do aspirador, abordando a importância de mantermos o nosso espaço limpo.			
Desenrolar e resultados gerais:			
O V consegue trabalhar bem com as duas mãos e consegue usar bem o aspirador, contudo por vezes altera as funções do aspirador sem querer. Começou a usa a varinha: "Paula, queres ver uma coisa?" – e copiou aspiradores. A AM trabalhou o tirar coisas da caixa de ferramentas mas teve dificuldades. Percebeu a utilidade do aspirador, embora não o saiba, ainda, usar sozinha.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk com nomes nos telhados.			
Observações			
A AM e o V estão com níveis de trabalho muito diferentes. A AM ainda não controla os movimentos da mão do ToonTalk.			

Session 6

Nome1	AM – 5		
Nome2	V – 5		
Hora inicial	14h58	**Hora final**	15h10
Objectivos prévios			
Trabalhar imagens de Natal para decoração da casa ToonTalk.			
Desenrolar e resultados gerais:			
A AM não aterrou perto das casas. Tive de identificar as casas para que a AM as visse. Não consegue encontrar o caderno de sensores nem a parede, ou o telhado, sem ajuda. Gostou de criar prendas com as caixinhas, o que fez sem dificuldade.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk.			
Observações			
O V faltou. Hoje a AM esteve muito distraída, só se concentrou quando lhe disse para fazermos prendas (com as caixas) para ela oferecer.			

Session 7

GRUPO: AM e V

IDADE: 5 anos

DATA: 4-02-03

HORA DE INÍCIO: 9h50m

HORA DE TERMINUS: 10h20m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado. Ensinar o robot a colocar o nome no telhado.

DESENROLAR E RESULTADOS GERAIS: o V concentra- se mais quando é a AM a mexer no rato pois gosta de explicar à colega.

O V controla o boneco e o rato mas é um pouco trapalhão, o que faz com que, por vezes, clique sem ser necessário.

Eu- "Porque é que o robot pára?"

V- " Porque está vazia, a caixa".

A AM não aterra numa casa. A AM só trabalhou o nome no telhado e precisou de ajuda pois estava muito distraída.

OBSERVAÇÕES: a AM esteve muito desconcentrada. A educadora estava a falar das ganchas e a mostrá-las e a AM não parava de olhar para ela.

Session 8

GRUPO: AM e V

IDADE: 5 anos

DATA: 13- 02- 03

HORA DE INÍCIO: 10h10m

HORA DE TERMINUS: 10h42m

OBJECTIVOS PRÉVIOS: colocar o desenho na parede e ensinar o robot a fazer o mesmo

DESENROLAR E RESULTADOS GERAIS: Eu- "o que precisamos para colocar o desenho na parede?"

V- " vou buscar a fita cola e colo".

O V não saiu do pensamento do robot sem que lhe lembrasse. Depois, a fazer sozinho, o V ia ensinar o robot fora do pensamento, mas já saiu do pensamento do robot.

A AM não programou o robot pois, como de costume, estava muito distraída e, como tal, esteve a trabalhar colocando ela o desenho na parede.

OBSERVAÇÕES: a AM não sabe como levantar o boneco.

O V, trocando com a AM para trabalhar ela com o rato, também se distraiu.

Session 9

NOMES: AM e V

IDADE: 5 anos

DATA:

HORA DE INÍCIO: 11h40

HORA DE TERMINUS: 12h03m

OBJECTIVOS PRÉVIOS:

DESENROLAR E RESULTADOS GERAIS: O V ia colocar os objectos iguais uns sobre os outros. A AM tem evoluído a nível de controlar movimentos e perceber como se seguram as coisas no toontalk. A AM já não se lembrava como fazer casas. Disse que usava a bomba (de rebentar).

O V disse que íamos para o pensamento do robô.

A AM não ia sair do pensamento do robô.

Perguntei o que tinha o robô no pensamento.

V -"Fazer uma casa".

Eu –Porque parou, AM?

R: A caixa está vazia.

O V usou a caixa inicial. Da 1ª vez ensinou o robô a aspirar.

Antes de verificar se ensinou o robot o V rebentou a casa. Voltou a ensina –lo.

Eu –Porque parou?

Em vez de responder, o V voltou a encher a caixa com as coisas que o robô tinha na cabeça.

OBSERVAÇÕES: o V tem andado muito distraído, só se concentra quando está ele a trabalhar.

Session 10

NOMES: AM e V

IDADE: 5 anos

DATA: 1- 04-03

HORA DE INÍCIO: 14h12m

HORA DE TERMINUS: 14h30m

OBJECTIVOS PRÉVIOS: trabalhar a troca

DESENROLAR E RESULTADOS GERAIS: tive de ajudar a AM a sair do pensamento do Robô.

A AM quis fazer outra caixa dupla para arrumar coisas.

A AM demorou a ensinar o Robô. Mesmo na 2ª tentativa a AM construiu outra caixa para testar se o Robô sabia o que lhe ensinou.

Eu – Porque parou, AM?

AM – Não sei.

A AM esteve a desenvolver o uso da varinha mágica. Confunde - se e pra copiar clica no rato.

OBSERVAÇÕES:

Session 11

NOMES: AM e V

IDADE: 5 anos

DATA: 19- 05-03

HORA DE INÍCIO: 15h12m

HORA DE TERMINUS: 15h35m

DESENROLAR E RESULTADOS GERAIS: a AM teve dificuldade em perceber como colocar o desenho na parede faz de conta. Pôr o nome à frente da casa já foi mais fácil mas não me pareceu perceber. O V relembra o que é preciso para fazer a casa: camião, caixa e robô.

O V sabe usar o caderno dos sensores embora inicialmente o confundisse com o caderno de imagens.

Session 12

NOMES: AM e V

IDADE: 5 anos

DATA: 05- 06-03

HORA DE INÍCIO: 15h02m

HORA DE TERMINUS: 15h22m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado e programar o robô para fazer o mesmo

DESENROLAR E RESULTADOS GERAIS: o V confunde–se muito. Para pôr o nome no telhado ia ao telhado directamente.

A AM também precisa de instruções para atingir os objectivos propostos.

Para a AM tornou –se mais fácil construindo uma caixa dupla com o necessário: telhado e nome. Mesmo assim a AM distrai –se facilmente e pergunta porque tem de repetir o que ainda não conseguiu fazer. Expliquei que era para aprender a fazer sem ajuda.

OBSERVAÇÕES: O V tem andado irrequieto, mesmo no decorrer do resto das actividades na sala de jardim. Ás vezes fica apático.

Não avancei na programação do robô pois ambos revelaram falta de interesse e distracção. O calor também complica qualquer fim de ano!

Group 10

Session 1a

Nome1	AX (3 anos)		
Nome2			
Hora inicial	11h40	**Hora final**	12h02
Objectivos prévios			
Aferir coisas que o AX já faz no ToonTalk quase que "naturalmente".			
Desenrolar e resultados gerais:			
A cidade que o AX tem já desde o ano 2001/2002 está com muitas casas. Sozinho, foi para a memória do robô. Eu – "Que estás a fazer, AX?" AX – "Estou a ensinar o robô a fazer casas. Olha pra isto..." Contudo, não verificava se o robô de facto fazia o que ele lhe ensinou.			
Lista de elementos de registo (ficheiros) associados:			
Desenho do Paint.			
Observações			
No Paint quis apagar o 1º desenho que fez e fazer de novo.			

Session 1b

Nome1	G		
Nome2			
Hora inicial	14h04	**Hora final**	14h25
Objectivos prévios			
Explorar e conhecer o ambiente do ToonTalk.			
Desenrolar e resultados gerais:			
Foi a 1ª sessão acompanhada do G. Treinou "sentar/levantar", andou a voar com o helicóptero. Aliás, quando clicámos para entrar no jogo disse: "É o jogo do helicóptero..." No Paint quis desenhar uma casa. De volta ao ToonTalk fez o robô construtor (já sabia fazer). Explorou o aspirador e a varinha mágica.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
Com o aspirador e a varinha ainda tem alguma dificuldade em perceber que tem que tê-los na mão para os poder usar.			

Session 2

Nome1	AX (3 anos)		
Nome2	G (4 anos)		
Hora inicial	10h30	**Hora final**	10h45
Objectivos prévios			
Noções: "vazio" vs. "cheio" e "dentro" vs. "fora".			
Desenrolar e resultados gerais:			
Na construção de caixas Eu - Para que é que podem servir as caixas? AX – "Pra meter coisas." Eu – "Então se isto for a nossa mala de viagem, o que é que lhe vais pôr dentro?" AX – "Vou levar um robô, um camião, uma bomba, um número, um camião e uma balança. Já está." Entrou para o pensamento do robô: "vou ensinar o robô a explodir a caixa..." Eu – Afinal o que é que lhe estás a ensinar? AX: A fazer casas.			
Lista de elementos de registo (ficheiros) associados:			
Desenho do Paint e caixa de "coisas para viagem" no caderno do ToonTalk.			
Observações			
AX: os passarinhos levam as coisas para o ninho para os filhotes comerem. Estive a explicar ao AX que se der uma caixa vazia ao robô o que lhe ensinar ele vai fazer sempre seguido. AX: fica sempre a fazer, é? Depois expliquei-lhe como se deve ensinar o robô.			

Session 3

Nome1	AX - 4		
Nome2			
Hora inicial	10:51	**Hora final**	11:26
Objectivos prévios			
Programar o robô para colocar o nome no telhado. Começou por aprender a colocar o nome no telhado.			
Desenrolar e resultados gerais:			
Não se lembrava de usar a caixa que fica feita. Ia fazer outro robô, fazendo outra caixa. Ficou sorridente ao ver o nome dele no telhado, quis fazer mais. "Tem os números verdes!", não era o sensor certo que ele tinha fixado. Mas na 2ª tentativa fez direito. Já se levantou para verificar se o nome estava no telhado.			
Lista de elementos de registo (ficheiros) associados:			
Robô que põe nome no telhado – no caderno do ToonTalk.			
Observações			
"Vou fazer o nome nas casas todas". Virou-se para os colegas: "Estas casas todas vão ser minhas." Já queria decorar a casa. O AX, se o deixarem, fica a trabalhar no computador indefinidamente.			

Session 4

Nome1	AX – 4		
Nome2	G – 4		
Hora inicial	10:17	**Hora final**	10:45
Objectivos prévios			
Trabalhar o controlo com o "space". A importância de mantermos os espaços do jardim de infância limpos e arrumados. Aspirador, bomba de ar.			
Desenrolar e resultados gerais:			
O espaço de uma casa já tinha muitas coisas no chão. Perguntei ao AX se deixávamos as coisas assim, ele entendeu que devia arrumar. "Oh G, estou a comer isto!" O AX já controla muito bem o "aspirar". Por vezes carrega sem querer no botão esquerdo do rato, alterando as funções do aspirador e depois queixa-se "Oh PF, isto não vai!" O AX quis ir visitar a casa dele, explicando ao G que o nome está no telhado. O G também controla bem os movimentos com o rato.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
Quando usou a bomba de ar, o AX soltou uma grande exclamação: "O robô está a encher!" Ao chegar à casa "AX", o AX já disse: "ah, tenho muitas coisas para aspirar!" O AX propôs ao G dar coisas ao passarinho. Perguntei-lhe porquê. AX: "Ele põe no ninho."			

Session 5

Nome1	AX – 4		
Nome2	G – 4		
Hora inicial	9h20	**Hora final**	9h47
Objectivos prévios			
Trabalhar imagens de Natal para decorar a casa ToonTalk.			
Desenrolar e resultados gerais:			
O AX entra no ToonTalk e começa a brincar a tirar coisas da caixa e a aspirar. Tem o cuidado de ligar e desligar o aspirador. Depois fomos buscar o desenho de Natal do AX. Ele quis guardar o desenho no caderno. Ao tentar pôr o desenho mais pequeno, teve dificuldade porque não soube alterar as funções da bomba de ar. Fez camiões pequenos para tentar procurar ver se conseguia construir casas pequeninas. O G teve de rever alguns comandos "básicos", pois tem faltado a sessões do ToonTalk.			
Lista de elementos de registo (ficheiros) associados:			
Cidade ToonTalk.			
Observações			
O AX gosta de ajudar o G, explicando-lhe o que tem de fazer. Ao dizer para levantar o boneco, o G levantou o boneco que tinha na mão.			

Session 6

NOME: AX

IDADE: 4 anos

DATA:23-01-03

HORA DE INÍCIO: 10h30

HORA DE TERMINUS: 10h40

OBJECTIVOS PRÉVIOS: realizar uma cidade em que as casas tivessem as fotos das crianças

DESENROLAR E RESULTADOS GERAIS: o AX sabe como reduzir a foto e procurar em cadernos sem ajuda. Só precisa que lhe indiquem quais os cadernos certos a usar.

Session 7

GRUPO: AX e G

IDADE: 4 anos

DATA: 18- 02- 03

HORA DE INÍCIO: 10h00m

HORA DE TERMINUS: 10h20m

OBJECTIVOS PRÉVIOS: colocar desenho na parede e ensinar o robot a fazer o mesmo.

DESENROLAR E RESULTADOS GERAIS: o AX já encontra o caderno simplificado de sensores. Perguntou porque é que não dá para entrar pela janela. Expliquei que nós também não entramos em casa pela janela, nem devemos tentar fazê- lo.

O AX fixou como trouxe o desenho para o toontalk através de "Ctrl+V". Construiu sozinho a caixa dupla para ensinar o robot a colocar o desenho na parede. Não testou o robot. Achou que por o ter ensinado ele já sabia.

OBSERVAÇÕES: o G faltou.

Session 8

GRUPO: AX e G

IDADE: 4 anos

DATA: 4-02-03

HORA DE INÍCIO: 14h42m

HORA DE TERMINUS: 15h05m

OBJECTIVOS PRÉVIOS: fazer o nome e colocá-lo no telhado.

DESENROLAR E RESULTADOS GERAIS: o AX conseguiu perceber o que precisava para ensinar o robot, mas tem tendência a dar uma caixa vazia ao robot pois já tinha descoberto que assim iria para o pensamento do robot.

O G precisou de indicações para pôr o nome no telhado. Depois, para programar o robot, já não precisou de indicações.

O G não ia usar a caixa já criada par verificar o que o robot aprendeu, mas o AX disse: "olha uma ali igual!".

OBSERVAÇÕES: de uma maneira geral, resultou ter simplificado o caderno de sensores. Tornou- se visível o telhado na 1ª página o que evitou que demorassem e tivessem que pedir ajuda.

Session 9

NOMES: AX e G

IDADE: 4 anos

DATA:

HORA DE INÍCIO: 14h06m

HORA DE TERMINUS: 14h41m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: O AX conseguiu perceber a finalidade do jogo. Perguntei -lhe porque o tinha feito daquela forma, respondeu: "Porque era assim."

No pensamento do robô o AX não percebe que se trata do robô, continuando a agir como sendo ele próprio.

Tivemos de ensinar algo mais simples ao robot.

O AX construiu uma caixa igual à do jogo.

Resolvemos ensinar o robô a fazer uma casa, o que foi um ânimo para o AX, uma vez que já domina totalmente essa "tarefa", daí que seja fácil ensina -la.

O AX esquece- se de sair do pensamento do robô.

Expliquei que se ensinarem o robô a copiar a caixa antes de a usar ele terá sempre uma para copiar e, como tal, não pára de fazer caixas.

Eu –se a caixa ficar vazia o que é que acontece?

Resp.: "ele não tem com que trabalhar".

O G não controla tão bem quanto o AX os movimentos a fazer com a mão, mas já consegue fazer o jogo autonomamente.

O G percebeu o que precisa para ensinar o robô a fazer a casa. Não se lembrava de dar a caixa com as coisas ao robô. Não se refere ao pensamento do robô, é uma noção demasiado abstracta para ele. Não saia do pensamento do robô.

Não ia usar a caixa inicial, igual ao pensamento do robô. O AX a rir diz: "Agora vai fazer muitas!"

OBSERVAÇÕES: O AX mostra sempre muito entusiasmo no computador.

Session 10

NOMES: AX e G

IDADE: 4 anos

DATA: 25-03-02

HORA DE INÍCIO: 15h23m

HORA DE TERMINUS: 15h45m

OBJECTIVOS PRÉVIOS: ensinar o robô a trocar a árvore com a flor

DESENROLAR E RESULTADOS GERAIS: o AX rapidamente trocou ele a flor e a árvore mostrando perceber a troca. Depois ensinou o robô e saiu do pensamento do robô e deu –lhe a caixa inicial para ele fazer o que lhe tinha ensinado. O AX ensinou o robô a tirar e voltar a encher as caixas.

Eu –Porque parou, AX?

AX – Porque só tinha uma caixinha.

O AX ensinou o G a fazer a troca e a ensinar o robô. O AX disse ao G para dar a caixa ao robô nas mãos, para ver se ele aprendeu.

O G esquece-se de dar a caixa ao robô para o ensinar. Já no pensamento do robô o G lembrou-se de usar a varinha mágica, mas não sai do pensamento. Também não testa o robô.

Eu – O que é que o robô está a pensar?

G – Nada.

Usou a caixa inicial para dar uma caixa ao robô.

OBSERVAÇÕES:

Session 11

NOMES: AX e G

IDADE: 4 anos

DATA: 19- 05-03

HORA DE INÍCIO: 15h37m

HORA DE TERMINUS: 16h02m

DESENROLAR E RESULTADOS GERAIS: o AX explicou ao G como pôr o desenho na parede verdadeira, através do caderno. Disse para ir buscar a parede ao caderno, para abrir e depois colocar o desenho na parede.

O AX começou por colocar o desenho dele mais pequeno, antes mesmo de usá –lo para pôr na parede. Encontrou o caderno simplificado dos sensores. Tirou logo a parede e a casa pois disse "vou fazer tudo ao mesmo tempo", para explicar que ia fazer rápido o que tinha de fazer.

O AX colocou o nome sem dificuldades na entrada de casa. Quis guardar as coisas que fez sem eu lho dizer.

O G também consegue entender o que tem de fazer para colocar nome ou desenho mas a um ritmo muito diferente do AX, que consegue abstrair –se e entender que colocando o desenho ou nome na parede ou casa "faz de conta" ele vai aparecer "na grande".

O AX quis depois ensinar o robô a colocar o nome na parede. Tentou fazê –lo dando uma caixa vazia ao robô. Expliquei - lhe depois que era melhor dar uma caixa já com o nome ao robô.

O AX, ao ver o robô aspirar depois de ter feito o que lhe ensinou, disse:"Este robô é malandro!". Expliquei –lhe que ele faz o que nós fazemos antes de sair do toontalk ou seja, deixa tudo arrumado.

Session 12

NOMES: AX e G

IDADE: 4 anos

DATA: 05- 06-03

HORA DE INÍCIO: 14h35m

HORA DE TERMINUS: 15h00m

OBJECTIVOS PRÉVIOS: ensinar o robô a colocar o nome no telhado

DESENROLAR E RESULTADOS GERAIS: o AX confunde o sensor do telhado com o sensor da casa. Depois de rapidamente ter posto o nome no telhado, o AX disse: agora vou ver.

Eu – O que precisas para ensinar o robô?

AX – Fazer uma caixa e depois pôr lá as coisas. O AX construiu a caixa mas depois ia dar uma caixa vazia ao robô.

Sai do pensamento do robô. Ia logo ver o resultado no telhado antes de o testar. Expliquei que ensinou o robô mas que ainda não tinha visto se ele sabia mesmo fazer. O AX usa o mais e menos para mexer no caderno.

O AX descobriu que se for o robô a pôr o nome no telhado o nome fica maior e então foi compor o primeiro que tinha feito. O AX já escreve o nome no robô. O G tem dificuldade em encontrar o caderno de sensores. Já sabe pôr o nome no telhado mas não vai verificar. O AX ajuda –o dando - lhe instruções. O G precisa de orientação para ensinar o robô.

Group 11

Session 1

Nome1	AL – 5		
Nome2			
Hora inicial	14:07	**Hora final**	14:34
Objectivos prévios			
Exploração do ambiente ToonTalk. Robô construtor de casas. Pôr o nome no telhado. Usar bomba de ar.			
Desenrolar e resultados gerais:			
A AL já conhece o jogo. Não se recordava dos diferentes botões para levantar, voar, aterrar. Ao pôr coisas dentro de caixas e ver o rato a sobrepor letras riu estridentemente. A AL quiz que o robô levasse um ninho para a casa. Deu coisas ao passarinho para o ver levar coisas. Depois de escrever o nome em duas casas fez uma casa para uma amiga (JG).			
Lista de elementos de registo (ficheiros) associados:			
Cidade com nomes.			
Observações			
É a 1ª sessão (acompanhada) deste ano lectivo, da AL. Num dos telhados colocou uma flor, depois quis colocar também o nome, para se saber que a casa era dela.			

Session 2

Nome1	AL		
Nome2			
Hora inicial	11:40	**Hora final**	12:02
Objectivos prévios			
Trabalhar o controlo da bomba de ar e do aspirador. A limpeza do espaço de trabalho e de brincadeiras.			
Desenrolar e resultados gerais:			
A AL não tem quaisquer dificuldades em utilizar os objectos da caixa de ferramentas. Também conseguiu perceber que P = pequeno e G = grande. Gosta de mandar construir casas e sai para, de helicóptero, ver as casas que construiu: "Olhas as casas que mandei fazer" (disse apontando). Sem lhe dar instruções quis pôr o nome no telhado. Só perguntou onde desfolhar o caderno.			
Lista de elementos de registo (ficheiros) associados:			
Observações			
Ja sabe usa F1, sem eu fazer qualquer referência, para mandar o marciano embora. Para pôr o nome no telhado tentou fazê-lo sem estar dentro de casa.			

Session 3

Nome1	AL		
Nome2			
Hora inicial	15h12	**Hora final**	15h28
Objectivos prévios			
Usar os desenhos do Paint para decorar uma casa na cidade do ToonTalk. Aprender que o passarinho leva as coisas para o ninho.			
Desenrolar e resultados gerais:			
A AL precisou de ajuda para encontrar o caderno de sensores. Quis decorar também o telhado com o desenho de Natal. A AL ao usar a bomba de ar perguntou como é que fazia para pôr as coisas pequenas. Disse-lhe que é na letra P e ela disse que ainda não sabe essa letra. Percebeu que o passarinho leva as coisas para o ninho e experimentou dar-lhe o nome para ele levar para outra casa onde estava o ninho.			
Lista de elementos de registo (ficheiros) associados:			
Desenho do Paint, cidade ToonTalk.			
Observações			
A AL sugeriu que era bom usarmos um rato sem fio!!			

Session 4

NOME: AL

IDADE: 5 anos

DATA: 4-02-03

HORA DE INÍCIO: 15h31m

HORA DE TERMINUS: 15h46m

OBJECTIVOS PRÉVIOS: colocar o nome no telhado e ensinar o robot a fazer o mesmo.

DESENROLAR E RESULTADOS GERAIS: a AL precisou que lhe lembrasse de procurar o telhado/ sensor no caderno, para escrever o nome. A AL perguntou como se levantava.

Não reparou na caixa igual à que tinha dado ao robot.

Saiu da casa para verificar que o nome estava no telhado e depois disse para voltarmos para a casa para arrumar tudo.

Session 5

NOME: AL

IDADE: 5 anos

DATA:

HORA DE INÍCIO: 15h42m

HORA DE TERMINUS: 15h57m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: a AL percebeu e explicou que estava a fazer igual. Ás vezes acontece -lhe largar uns objectos sobre outros sem querer.

Para ensinar o robô a fazer casas a AL pensou precisar de uma caixa e de um camião, pois robô já tinha (o que queria ensinar!).

Disse que tinha de sair do pensamento do robô.

Não usa a caixa inicial.

Aprendeu a copiar a caixa para que o robot não pare.

AL –"Temos que ensinar coisas e para não parar temos de ensina -lo a fazer truques com a varinha mágica."

OBSERVAÇÕES:

Session 6

NOME: AL

IDADE: 5 anos

DATA: 25-03-02

HORA DE INÍCIO: 14h55m

HORA DE TERMINUS: 15h17m

OBJECTIVOS PRÉVIOS: Ensinar o Robô a fazer uma troca de objectos numa caixa dupla

DESENROLAR E RESULTADOS GERAIS: a AL perguntou porque é que a árvore tem olhos. Respondi que era uma árvore de desenhos animados, como tal podia ter olhos. Fez a troca rapidamente. Quando ia ensinar o robot ia copiar a caixa antes de entrar no pensamento do robô e depois perguntou se tinha de lhe dar duas caixas vazias. No pensamento do robô lembrou -se de usar a varinha mágica mas para copiar objectos. Não se lembrou que é para o robot não parar. Quando saiu disse: "tenho de lhe dar a caixa".

AL: Porque é que a árvore ficou vermelha?

Eu – Porque ele já tinha usado a árvore.

Copiar a caixa baralhou a AL na troca. Expliquei que no 1º robô a árvore fica vermelha porque só ensinamos o robô com a caixa igual ao pensamento e depois da troca a caixa tem outra ordem. A AL voltou a ensinar o robô e lembrou –se de lhe dar a caixa para ver se ele Aprendeu. Diz que não o ensinou a aspirar mas que ele aspira. Expliquei que nós também costumamos deixar tudo arrumado.

OBSERVAÇÕES:

Session 7

NOMES: AL e CA

IDADE: 5 anos

DATA: 20- 05-03

HORA DE INÍCIO: 15h25m

HORA DE TERMINUS: 15h49m

OBJECTIVOS PRÉVIOS:

DESENROLAR E RESULTADOS GERAIS: a AL já não se recordava como decorar a parede.

Estiveram a pôr o nome na casa. A CA relembrou a AL onde levantava o boneco. A CA ao guardar o que fez no caderno quis colocar de um lado o que fez e do outro o nome, como costumamos fazer. A AL e a CA conseguiram perceber a actividade.

OBSERVAÇÕES: como a U está para a Ucrânia, a CA fez com a AL

Session 8

NOMES: AL

IDADE: 5 anos

DATA: 03- 06-03

HORA DE INÍCIO: 10h10m

HORA DE TERMINUS: 10h30m

OBJECTIVOS PRÉVIOS: já referidos

DESENROLAR E RESULTADOS GERAIS: para pôr o nome no telhado disse que precisava de letras e do telhado.

Foi verificar se o nome estava no telhado.

Teve dúvidas em sair do pensamento do robô.

A AL coloca a caixa para o pensamento na zona branca.

A AL gosta sempre de arrumar o que desarrumou no toontalk, deixando as casas arrumadas.

São Vicente de Paula preschool

Group 1

Session 1

Nome1	IT		
Nome2	ACS		
Hora inicial	14:30	**Hora final**	14:50

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

A Catarina ainda tem um pouco de receio do computador. Esteve muito bem mas chegou a uma dada altura em que não queria fazer mais, não demorou o tempo todo.

A Inês, por sua vez, como a Catarina era muito sossegada ajudou a Catarina, tendo ela própria tido um bom desenrolar da actividade.

Lista de elementos de registo (ficheiros) associados:

Observações:

Uma falha que notei foi que ao colocar o robô a funcionar, tendo de lhe dar uma caixa azul, a que se encontrava no chão da casa, fica sempre por trás do pensamento do robô, sendo difícil para as miúdas ir descobri-lo para dar ao robô.

Session 2

This matches session 2b in group 10.

Nome1	J		
Nome2	IT		
Hora inicial	9:40	**Hora final**	10:00
Objectivos prévios			
Relação de 1 para 1, relação de 1 pássaro –> n (ninhos) Pergunta->O que é que o pássaro fazia ao objecto que nós lhe dávamos na relação de 1 pássaro para 1 ninho. Depois o que é que o pássaro fazia se copiasse os ninhos? Onde é que ele iria pôr o objecto? Depois entrei no pensamento do robô para os miúdos o programarem a fazer a relação de 1 pássaro para n ninhos. Neste caso, copiei apenas 3 ninhos. Inicialmente, expus as duas situações fazendo as perguntas para ver como reagiram.			
Desenrolar e resultados gerais:			
J: para pegar no ninho levantaram o rato do tapete. Em relação à segunda pergunta, disse que o pássaro morria. Sabem como é que se entra no pensamento do robô. Conseguiu descobrir a caixa azul por detrás do robô para o pôr a funcionar. Achou piada no pensamento do robô estar a utilizar o aspirador e por ser tão pequenino com pernas e olhos. IT: Resposta à pergunta: se calhar come-os e depois, morre.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Group 2

Session 1

Nome1	F		
Nome2	M		
Hora inicial	14:50	**Hora final**	15:10

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

São dois miúdos que já dominam razoavelmente o computador. Estiveram a ouvir a minha explicação e depois fizeram bastante bem, tendo sido a minha intervenção no que diz respeito a ajuda muito pouco.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Nome1	F		
Nome2	M		
Hora inicial	10:00	**Hora final**	10:20
Objectivos prévios			
Os mesmos			
Desenrolar e resultados gerais:			
Ambos não souberam responder à primeira pergunta. São muito calados. Em relação à segunda disseram que ia colocá-los em todos os ninhos. F: Depois de entrar no pensamento do robô ia pô-lo a fazer casas, mas depois conseguiu fazer sozinho o que estava a fazer fora do pensamento do robô. O M, depois de ver o F, entrou directamente no pensamento do robô e utilizou bem a varinha para fazer cópias dos ninhos. Saiu bem do pensamento do robô e conseguiu pô-lo a funcionar.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Group 3

Session 1

Nome1	IF		
Nome2	MG		
Hora inicial	15:15	**Hora final**	15:40

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

São duas miúdas que ainda estão um pouco verdes no que diz respeito à motricidade fina. Sendo necessário ainda a minha ajuda. Nestes dois casos eu ainda ajudei bastante, portanto as duas ainda não conseguiram fazer sozinhas.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2a

This matches session 2a in group 7.

Nome1	FL		
Nome2	IF		
Hora inicial	14:41	**Hora final**	15:00
Objectivos prévios			
Os mesmos			
Desenrolar e resultados gerais:			
IF: Ao colocar a pergunta na relação 1 para 1, disse que iria colocá-lo, o objecto, no ninho. Quando fiz a cópia dos ninhos, também me disse que iria pôr o objecto num ninho. Então perguntei porque é que não vai pôr nos outros ninhos também. Depois, ficou à espera do que poderia acontecer. Ainda se lembrou do que é que a varinha fazia e que era para copiar. FL: Colocava o objecto num só ninho porque há só um. Não conseguiu fazer sozinho, eu é que tive de o ajudar. Sabem bem onde é sair e copiar. Dentro do pensamento do robô, quando disse que era com a outra mão do robô, ele trocou a mão dele no rato.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Session 2b

This matches session 2a in group 10.

Nome1	MG		
Nome2	MA		
Hora inicial	15:00	**Hora final**	15:20
Objectivos prévios			
Os mesmos.			
Desenrolar e resultados gerais:			
MG: disse depois de fazer a cópia que punha somente num, porque somente tenho um passarinho. Também tive de a ajudar muito. MA: lembra-se como é que se entra; conseguiu tirar o ninho. Respondeu à pergunta dizendo que só punha a balança em dois ninhos! Pouco tive de ajudar.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Group 4

Session 1

Nome1	AT		
Nome2	CP		
Hora inicial	15:40	**Hora final**	16:00

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

Aplica-se a mesma situação à da IF e MG.

Lista de elementos de registo (ficheiros) associados:

Observações:

Group 5

Session 1

Nome1	FM		
Nome2	D		
Hora inicial	9:30	**Hora final**	9:50

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

O FM tem um bom desempenho. Depois, como o D ainda não consegue desenvolver, eu é que tenho de fazer tudo, o FM acabou por ajudar o D, dizendo-lhe como é que ele devia fazer.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2a

This matches session 2a in group 11.

Nome1	AC		
Nome2	D		
Hora inicial	14:15	**Hora final**	14:40
Objectivos prévios			
Os mesmos			
Desenrolar e resultados gerais:			
AC: Tive de pôr a minha mão por cima da dela e fazer praticamente tudo. Depois não quis fazer sozinha. D é um daqueles miúdos que não fala, não reage, tenho de fazer tudo, não tem muito poder de concentração. Depois de fazer a pergunta, já com as cópias dos ninhos, perguntei se ao dar um objecto ao pássaro ele iria colocar nos 3 ninhos e ambos disseram que sim. Mas eu é que tive de sugerir o que é que o pássaro iria fazer.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Session 2b

This matches session 2b in group 9.

Nome1	BB		
Nome2	FM		
Hora inicial	10:40	**Hora final**	11:00
Objectivos prévios			
Os mesmos.			
Desenrolar e resultados gerais:			
BB: resposta à primeira pergunta: vai é voar e vai-se embora. Reposta à segunda: vai fazer mais pássaros. Queria tirar outro ninho e copiar mais pássaros. Pôs o robô a funcionar com a caixa azul por trás do pensamento. Disse: este robô é complicado porque ela não estava a colocá-lo onde queria. Disse a caixa vai sempre comigo. FM: A CA explicava e punha a mão no FM para por a mão no monitor para ter muitos pássaros. Tive de o ajudar em todos os passos.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Group 6

Session 1

Nome1	FS		
Nome2	C		
Hora inicial	9:50	**Hora final**	10:10

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

O FS somente ao fim é que descobriu que o robô fazia igual, ficando um pouco admirado com a saída dos camiões das casas.

A C conseguiu mais ou menos, tendo gostado de ver os camiões a construir as casas na ilha.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

This matches session 2b in group 7.

Nome1	DI
Nome2	FS

Hora inicial	15:20	**Hora final**	15:40
Objectivos prévios			
Os mesmos.			
Desenrolar e resultados gerais:			
DI: é um miúdo que fala muito e diz tudo o que tem a dizer. Resposta à pergunta: vou ver, teve ele de dar um objecto ao pássaro para ver o que é que ele iria fazer. Passou logo para o pensamento do robô para o pôr a fazer o mesmo. Pegou na caixa azul que estava por detrás do pensamento do robô, para o pôr a funcionar. FS: Achava que o pássaro iria pôr o objecto sempre num só ninho. Quando chegou à parte de dar ordens ao robô perguntou e agora. Depois o DI é que lhe disse que tinha de ir buscar a caixa azul. O DI ajudou-o muito nos passos a seguir.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Group 7

Session 1

Nome1	FL		
Nome2	DI		
Hora inicial	10:10	**Hora final**	10:30

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

O DI é um miúdo que está sempre a fazer perguntas, lembrou-se bem da sessão anterior e pensou que iríamos fazer a mesma coisa. Conseguiu após a minha explicação fazer bem a actividade.

O FL ainda tenho de o ajudar bastante.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2a

This matches session 2a in group 3.

Session 2b

This matches session 2 in group 6.

Group 8

Session 1

Nome1	B		
Nome2	CA		
Hora inicial	10:30	**Hora final**	10:50

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

A B ajudou a CA, depois da minha explicação. Ambas precisaram de pouco da minha ajuda.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2

Nome1	CA		
Nome2	B		
Hora inicial	9:20	**Hora final**	9:40
Objectivos prévios			

Os mesmos.
Desenrolar e resultados gerais:
Primeira pergunta: vai pô-lo no ninho. Segunda pergunta: a CA disse que ia pô-lo nos três ninhos. B: fez bem a cópia dos ninhos. Entrou facilmente no pensamento do robô e depois de o programar conseguiu pô-lo a funcionar com a caixa por detrás do pensamento. A B ajudou a CA nos passos em que se enganava.
Lista de elementos de registo (ficheiros) associados:
Observações

Group 9

Session 1

Nome1	BB		
Nome2	L		
Hora inicial	10:55	**Hora final**	11:20

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

A BB já não precisou muito da minha ajuda e conseguiu, após a explicação, fazer o exercício, sem problemas.

O L é o oposto, eu é que tive de colocar a minha mão no rato e fazer quase tudo.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2a

This matches session 1 in group 12.

Session 2b

This matches session 2b in group 5.

Group 10

Session 1

Nome1	J		
Nome2	MA		
Hora inicial	11:20	**Hora final**	11:40

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

A J conseguiu descobrir a caixa atrás do pensamento do robô.

Antes, tinha pedido para ir ver a onde fora parar o camião.

A MA, depois de ver a J, conseguiu sozinha fazer a actividade.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2a

This matches session 2b in group 3.

Session 2b

This matches session 2 in group 1.

Group 11

Session 1

Nome1	A		
Nome2	AC		
Hora inicial	1140	**Hora final**	12:00

Objectivos prévios:

Construção de casas

Desenrolar e resultados gerais:

Mais uma vez com a AC não conseguiu descobrir a caixa que estava por detrás do pensamento do robô para o pôr a funcionar.

A A conseguiu fazer tudo sozinha até ao fim. Enquanto que a AC precisou muito da minha ajuda.

Lista de elementos de registo (ficheiros) associados:

Observações:

Session 2a

This matches session 2a in group 5.

Session 2b

Nome1	AC		
Nome2			
Hora inicial	11:20	**Hora final**	11:45
Objectivos prévios			
Os mesmos.			
Desenrolar e resultados gerais:			
A AC não fala quase nada e tive de estar a fazer-lhe tudo. Não quer ficar muito tempo no computador.			
Lista de elementos de registo (ficheiros) associados:			
Observações			

Group 12

Session 1

This matches session 2a in group 9.

Senhora da Pena preschool

Group 1

Session 1

This matches Session 1 in Group 3.

Data: 10/10/2002 **Grupo:** F/JG

Hora Inicial: 10h **Hora Final:** 10.30h

Objectivos prévios:

- Conhecer
- Explorar/Brincar
- Mexer objectos

Desenrolar e resultados finais:

Não houve nesta sessão uma proposta objectiva de trabalho. Sendo o primeiro contacto, era necessário observar, explorar, mexer e interrogar. O despertar para algo novo e o estímulo a novas sessões foi de certo modo conseguido.

No decorrer da sessão é de salientar alguns comentários feitos pelas crianças.

Frederico: "Olha um avião..."

" Vamos pôr nas caixas"

"Olha a minha mão abanar, olha..."

"Olha o que ela apanhou!..." "Vou largar"

O F adorou tirar números infinitamente, e camiões e fazia-os transportar os números, descobriu que colocando um robô ele andaria.

O facto de descobrirem que abanavam os objectos (alertados para esse facto), deu-lhes muita satisfação e entusiasmo.

O JG queria que existisse uma cadeira para o boneco se sentar.

Lista de elementos de registo (ficheiros) associados:

Observações:

Na sua totalidade estas crianças não tinham tido contacto com o computador, algumas estão no jardim pela primeira vez.

O Toon Talk provocou-lhes admiração, entusiasmo e interesse.

Session 2

Data: 17/10/2002 **Grupo:** F/JO

Hora Inicial: 14.15h **Hora Final:** 15.45h

Objectivos prévios:

- Conhecer
- Explorar/Brincar
- Mexer objectos
- Manipular

Desenrolar e resultados finais:

As crianças ainda não exploraram totalmente o jogo, daí que as estratégias para os objectivos que definam uma actividade ainda se torne difícil.

As crianças carregaram camiões com caixas e pretendiam despejá-los. O facto de lhe ter dito que com um camião transportando um objecto e um robô poderia construir uma casa, quiseram fazê-lo repetidas vezes. As caixas e os números dentro das mesmas foi o que mais fizeram.

" Vou pôr números para dar os anos da B..."

" Arruma isso..."

Nesta sessão o F já trabalhou o pensamento do robô, um pouco inconsciente, mas deu-lhe uma caixa com uma balança.

Lista de elementos de registo (ficheiros) associados:

Observações:

As crianças já interiorizaram um pouco o jogo em si, ou seja as suas componentes e certas finalidades.

Session 3

Data: 28/11/2002 **Grupo:** JO/F

Hora Inicial: 14.45h **Hora Final:** 15.30h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

O JO e o F tiraram caixas e letras para fazerem o nome e colocaram nas caixas, depois tornavam a tirar e com a bomba aumentaram e diminuíram para verem o grande e o pequeno. O F gosta de escrever o nome e tirar do caderno os camiões com os bonecos para brincar. O JO constrói casas e depois vai vê-las navegando no helicóptero.

Lista de elementos de registo (ficheiros) associados:

Observações:

O JO disse que a bomba de encher era a bomba da gasolina. O F gostava que os bonecos, que são de imagem saíssem do camião para brincarem com ele.

Session 4

Data: 13/01/2003 **Grupo:** JO/F

Hora Inicial: 10.45h **Hora Final:** 11.15h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

Hoje decidiu-se em grande grupo que se a partir do toon talk os símbolos para os cabides da sala de jardim. O JO e o F antes de trazerem a fotografia, com é habitual quiseram andar com o helicóptero, depois sendo a sua preferência brincar com os camiões resolveram colocar caixas e robôs e construir muitas casas. Já querem "fazer", o nome para em seguida colocar no telhado, onde podemos registar o seu sentimento de posse e de algo que conseguiram construir. Depois quiseram trazer a fotografia para o toon talk para elaborarem a moldura.

Lista de elementos de registo (ficheiros) associados:

Observações:

Nota-se que percebem o que estão a fazer e que no quarto da casa é o sítio dos brinquedos.

Session 5

Data: 10/02/2003 **Grupo:** JO/F

Hora Inicial: 14.15h **Hora Final:** 14.45h

Objectivos prévios:

- Trabalhar os pássaros e os ninhos
- Grau de quantidade
- Relacionar
- Desenvolver a observação

Desenrolar e resultados finais:

O JO e o F adoram andar de "avião", descem, param saem do helicóptero e descobriram que se podem afastar porque carregando na tecla do helicopetrr4o ele vem ter com eles. Com os pássaros quiseram duplicar os pássaros e os ninhos e entregar números aos pássaros para eles aprenderem a contar a confusão instalou-se, mas eles divertiram-se. Cada pássaro era um menino por isso teriam que ser muitos pássaros.

Lista de elementos de registo (ficheiros) associados:

Observações:

O JO, explicou ao Frederico, "o homem foge mas carrego no botão do avião e ele vêm porque o outro foi para o lixo."

Group 2

Session 1

Data: 18/10/2002 **Grupo:** A/L

Hora Inicial: 10h. **Hora Final:** 10.45h

Objectivos prévios:

- Explorar/Brincar
- Mexer objectos
- Manipular
- Encaixar

Desenrolar e resultados finais:

A A aderiu muito bem ao jogo, e está sempre aberta a novas descobertas.

Mexe bem o rato por isso consegue muito bem agarrar e largar objectos. A A utilizou a bomba de encher e resolveu aumentar os números, " vou encher números..." o L contrapôs a colega dizendo, " não, não, a bomba é para encher os pneus dos camiões". O L quis mexer com o aspirador e chamou-lhe peixe gordo, "vou passeá-lo".

A A já compreendeu que pode aspirar o que não consegue arrumar na caixa.

Lista de elementos de registo (ficheiros) associados:

Observações:

Nesta sessão as crianças já coordenam o rato em simultâneo com os movimentos dos objectos que pretendem mexer. O ambiente do jogo já lhes é familiar, podendo-se avançar para actividades direccionadas e interligá-la com as da sala.

Session 2

Data: 21/10/2002 **Grupo:** A/L

Hora Inicial: 10.20h **Hora Final:** 11.10h

Objectivos prévios:

- Mexer objectos
- Manipular
- Colocar objectos nas caixas
- Escrever nomes nas caixas
- Registar no caderno

Desenrolar e resultados finais:

Como referi no grupo anterior, tentei que se utiliza-se a actividade da receita para trabalhar no toon talk, mas a vontade de criar foi mais forte, ligada á de curiosidade e de exploração.

A A começou por andar um pouco de helicóptero não perdendo de vista as casas, depois quis entrar na casa azul. Escolheu caixas e quis colocar numa números e noutra letras. Depois quis juntar números dentro da caixa. Depois quis aspirar tudo. Ficou um objecto no chão, teve a percepção disso e voltou a ajoelhar-se e disse, " ...Oh, não tires tudo da caixa, palerma...", falou para o boneco.

Lista de elementos de registo (ficheiros) associados:

Observações:

Já têm a noção de ajoelhar baixar o helicóptero, entrar nas casas e das ferramentas da mala.

Session 3

Data: 28/11/2002 **Grupo:** A/L

Hora Inicial: 9.45 **Hora Final:** 10h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

A A tirou caixas, ninhos e robôs. A A gosta de ver os pássaros a saírem dos ninhos, depois faz jogos de faz de conta, como se estivesse a brincar com bonecos no chão. Fala sozinho e com os "bonecos", com o pássaro e o robô dando-lhes ordens. De vez em quando vai andar de helicóptero para ir fazer compras. Depois regressa e percorre o caderno para tirar objectos, a árvore, os bonecos e as flores para fazer quadros. O L é mais caladinho mas adere ás brincadeiras da A. Este tira da mala os camiões para transportar coisas e fazer casas.

Lista de elementos de registo (ficheiros) associados:

Observações:

Estamos a preparar algo alusivo ao Natal para trabalhar no toon talk.

Session 4

Data: 28/11/2002 **Grupo:** A/L

Hora Inicial: 15.30h **Hora Final:** 15.45h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

A A e o L quiseram voltar ao toon talk, para acabarem as "construções".

Colocaram 2 balanças no chão da casa, depois de brincarem os dois o L disse que aquilo (a balança) eram passadores 1 para a farinha e outro para o azeite, a avó tem um assim.

Lista de elementos de registo (ficheiros) associados:

Observações:

São crianças muito pequenas, no entanto já interiorizaram o jogo e trazem muitas das suas vivências para o pequeno ecrã, como se dele fizessem parte num mundo do faz de conta.

Session 5

Data: 13/01/2003 **Grupo:** A/L

Hora Inicial: 10.00h **Hora Final:** 10.30h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

Hoje decidiu-se em grande grupo que se a partir do toon talk os símbolos para os cabides da sala de jardim. A A e o L escolheram a moldura no caderno, escreveram o nome a partir da letra e trouxeram a fotografia para o toon talk e depois de colocada na moldura com o respectivo nome imprimimos.

Lista de elementos de registo (ficheiros) associados:

Observações:

O facto de trabalharem no toon talk criando algo e terem ao mesmo tempo colocado a sua imagem dentro deste é algo muito delirante e positivo para eles. Ficaram entusiasmadissímos com tal facto.

" olha, o boneco já não está sozinho eu já estou lá." (L)

Group 3

Session 1

Data: 21/10/2002 **Grupo:** JG/C

Hora Inicial: 9.45h **Hora Final:** 10.15h

Objectivos prévios:

- Mexer objectos
- Manipular
- Colocar objectos nas caixas
- Escrever nomes nas caixas

Desenrolar e resultados finais:

Anteriormente a esta sessão, a actividade da sala tinha sido culinária, mais propriamente "Biscoitos". Trabalhamos no paint os desenhos da receita como registo acompanhado do registo escrito feito no Word. Deste modo, propus transportarmos esta actividade para o toon talk. Não foi possível porque as crianças ainda necessitaram explorar um pouco mais. Resolvi direccionar para o trabalho de caixas e colocar o que iam fazendo no caderno.

O Helicóptero continua a ser a preferência das crianças. No caso do JG e da C, quiseram andar de helicóptero acompanhavam o barulho deste com vocalizações. O efeito da bomba a estoirar também foi feito repetidamente.

Lista de elementos de registo (ficheiros) associados:

Observações:

O JG e a C têm 3 anos, gostam do toon talk, a exploração dos objectos ainda foi feita com muita vivacidade. A curiosidade pelo funcionamento nota-se na constante pergunta, "...e isto para que serve?", "vou ver, sim?..."

Session 2

Data: 10/02/2003 **Grupo:** JG/C

Hora Inicial: 15.00h **Hora Final:** 15.30h

Objectivos prévios:

- Trabalhar os pássaros e os ninhos
- Grau de quantidade
- Relacionar
- Desenvolver a observação

Desenrolar e resultados finais:

O JG brincou com o helicóptero, e queria trazer os pássaros para o jardim. Duplicou os pássaros e depois dava-lhe objectos para ele comer. Depois duplicou os ninhos para guardarem a comida.

Lista de elementos de registo (ficheiros) associados:

Observações:

A C divertiu-se muito com os passarinhos mas tinha pena deles porque não tinham mãe.

Group 4

Session 1

Data: 21/10/2002 **Grupo:** P/FI

Hora Inicial: 11.15h **Hora Final:** 11.45h

Objectivos prévios:

- Mexer objectos
- Manipular
- Colocar objectos nas caixas
- Escrever nomes nas caixas

Desenrolar e resultados finais:

O P manuseia bem o rato, tem destreza manual, o que ajuda bastante no trabalho no toon talk. Ao entrar no ecrã foi decidido. Colocou caixas em vários camiões para construir muitas casas. O P, lembrou-se da nossa conversa e disse que uma das casas seria o armário dos biscoitos.

Adora explodir com as casas, (o pai é técnico de explosivos em obras e ele referiu fazer como o pai). Tirou um robô e trabalhou o pensamento com caixas.

Em todos os camiões que tirava colocava caixas com objectos diferentes e com os robôs, e resolveu a certa altura utilizar a varinha para os duplicar.

A FI só quis andar de helicóptero à deriva.

Lista de elementos de registo (ficheiros) associados:

Observações:

As crianças mantem-se entusiasmadas, ainda não querem fazer actividades direccionadas, continuam a explorar a função dos objectos exploram.

Session 2

Data: 13/01/2003 **Grupo:** FI/P

Hora Inicial: 11.30h **Hora Final:** 12.00h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

Hoje decidiu-se em grande grupo que se a partir do toon talk os símbolos para os cabides da sala de jardim. A FI e o P também estiveram empenhados na concretização da tarefa de hoje. Acederam ao toon talk normalmente, escrevem o seu nome e clicam para entrar. Já no ecrã da casa foram ao caderno escolher a moldura para colocar a fotografia. Depois da actividade o P quis construir casa para de seguida explodir para passarem estradas até França.

Lista de elementos de registo (ficheiros) associados:

Observações:

O P e a FI já se sentem á vontade no toon talk, e interagem bem os dois. A actividade e a possibilidade de trazerem as fotografias deles para o toon talk aproximou-os mais do jogo tornando-o mais íntimo.

O P chegou a comentar:

" Eu já consegui entrar para alí...olha ..."

Group 5

Session 1

Data: 21/10/2002 **Grupo:** JA/CA

Hora Inicial: 11.45h **Hora Final:** 12h

Objectivos prévios:

- Mexer objectos
- Manipular
- Colocar objectos nas caixas
- Escrever nomes nas caixas

Desenrolar e resultados finais:

Colocaram objectos nas caixas, trabalharam objectos nas caixas e colocaram imagens no caderno. Quiseram fazer o nome, (são dos 5 anos). Escreveram-no nas caixas. No fim aspiraram tudo não quiseram arrumar.

Lista de elementos de registo (ficheiros) associados:

Observações:

(O trabalho não ficou completo, estava na hora do almoço.)

Estas duas crianças, tem 5 anos e já aceitam actividades sugeridas.

Session 2

Data: 28/11/2002 **Grupo:** JA/CA

Hora Inicial: 11h **Hora Final:** 11.45h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

A JA e a CA já se movimentam bem no jogo. Trabalhamos com caixas algumas noções de tamanhos, de dentro e fora e de sequências. Elas gostam imenso de trabalhar com o caderno, construindo caixas com nomes e desenhos e colocando no caderno. Fazem construções de palavras e sequências de números.

Lista de elementos de registo (ficheiros) associados:

Observações:

Estamos a preparar algo alusivo ao Natal para trabalhar no toon talk. A CA disse que gostava de fechar o caderno para ver a capa.

Session 3

Data: 13/01/2003 **Grupo:** JA/CA

Hora Inicial: 14.30h **Hora Final:** 15.00h

Objectivos prévios:

- Manipular o caderno
- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

Hoje decidiu-se em grande grupo que se a partir do toon talk os símbolos para os cabides da sala de jardim. A JA e a CA estavam entusiasmadíssimas em ver as suas fotografias no toon talk e quiseram enfeitar a moldura com flores. Depois quiseram enfeitar a casa e o jardim.

Lista de elementos de registo (ficheiros) associados:

Observações:

A JA e a CA estão muito bem adaptadas com o toon talk, para além de mexerem muito bem no caderno gostam de trabalhar os números e letras escrevendo palavras conhecidas delas e copiando outras, encaixando-as de seguida em caixas.

Session 4

Data: 10/02/2003 **Grupo:** JA/CA

Hora Inicial: 10.30h **Hora Final:** 11.15h

Objectivos prévios:

- Trabalhar os pássaros e os ninhos
- Grau de quantidade
- Relacionar
- Desenvolver a observação

Desenrolar e resultados finais:

A JA e a CA tiraram um ninho da caixa, o pássaro saiu e com a varinha mágica copiaram vários ninhos. Depois foram ao caderno e trouxeram uma flor para dar ao pássaro que se desdobrou em vários para colocar nos ninhos copiados. Fizeram isso repetidas vezes e com objectos diferentes.

Lista de elementos de registo (ficheiros) associados:

Observações:

A JA, " Oh, o pássaro fez muitas flores..."

A CA, " Não apareceram foi muitos pássaros e depois fugiram e ficou só um..."

Group 6

Session 1

Data: 28/11/2002 **Grupo:** E/S

Hora Inicial: 10.5h **Hora Final:** 11h

Objectivos prévios:

- Manipular os objectos sabendo as suas funções,
- Articular acções com o tema do Jardim.

Desenrolar e resultados finais:

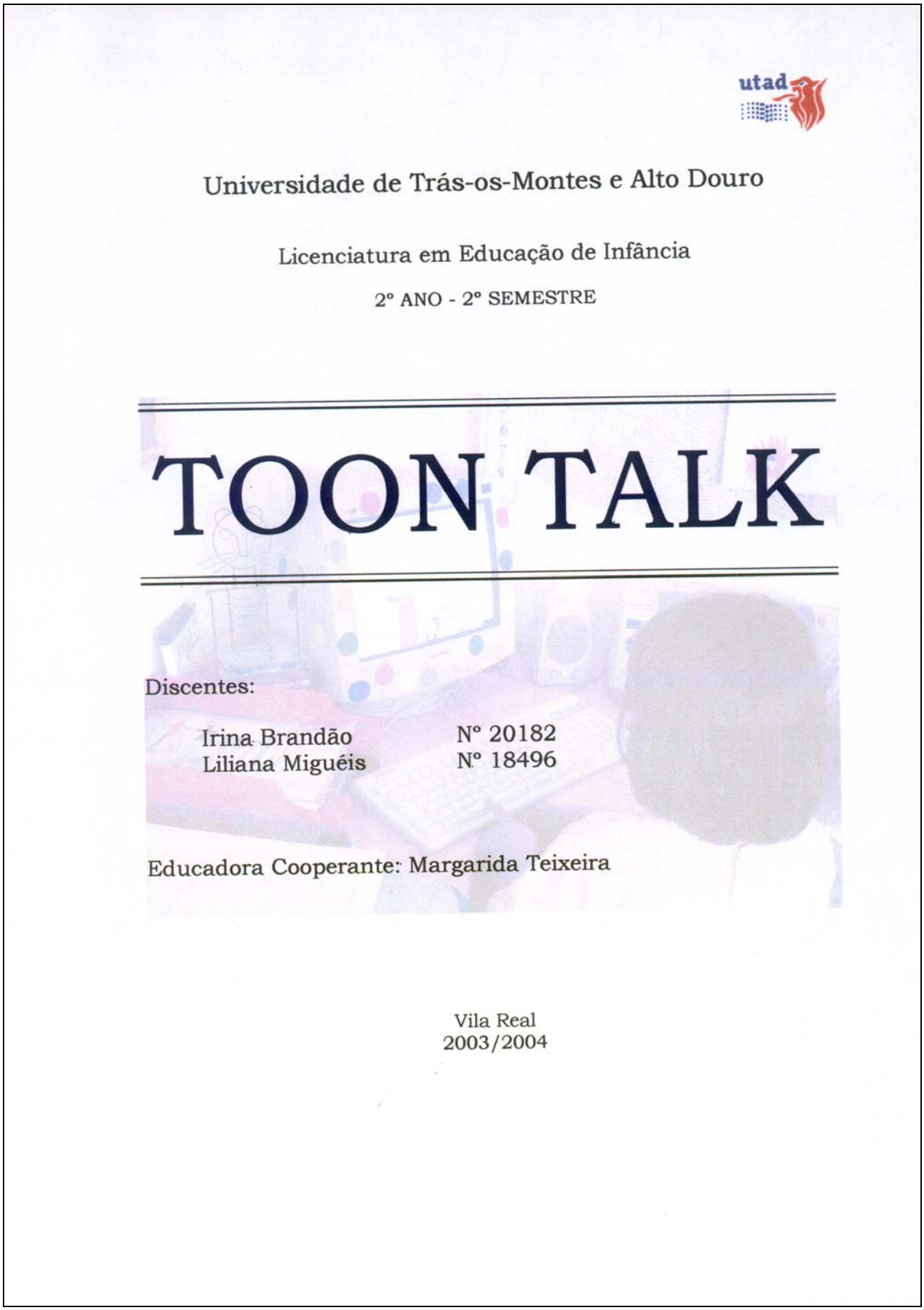
O E e a S gostam de experimentar todos os objectos da mala. Os camiões são os seus preferidos. O E já coloca caixas robôs e objectos nos camiões para construir muitas casas "quando vier da Suíça".

Lista de elementos de registo (ficheiros) associados:

Observações:

O E, " A Bomba é para cantar os parabéns."

Annex IX
—
Report on activities conducted by two second-year trainees of the Early Childhood Education baccalaureate



Universidade de Trás-os-Montes e Alto Douro

Licenciatura em Educação de Infância

2º ANO - 2º SEMESTRE

TOON TALK

Discentes:

Irina Brandão Nº 20182
Liliana Miguéis Nº 18496

Educadora Cooperante: Margarida Teixeira

Vila Real
2003/2004

Introdução

Este relatório consiste na delineação do trabalho realizado no Jardim de Infância de Parada de Cunhos com um grupo de 18 crianças de 3, 4 e 5 anos com o jogo Toon Talk. Integrado na nossa prática de Disciplina de Observação das Actividades Educativas II, o trabalho foi feito durante o período das actividades livres (momento em que as crianças circulam nos vários espaços da sala). Animámos o cantinho do Computador com o grande grupo, explorando o Toon Talk e com um grupo pequeno, de 4 crianças, às quais fizemos uma proposta específica integrada no projecto "Vamos Descobrir Portugal" desenvolvido ao longo do semestre.

O nosso trabalho divide-se principalmente em duas partes, o registo diário com o pequeno e o grande grupo. Ambas as partes estão subdivididas em objectivos, comentários das crianças, avaliação e registos (em tabelas no grande grupo e imagens do Toon no pequeno grupo). Os registos do pequeno grupo estão completados com ficheiros (em anexo no CD) das várias evoluções que a Cidade sofreu (pasta CIDADE), a localização das pastas está abaixo do título «Registos/observações do dia…». Note-se que o CD contém ainda este relatório (ficheiro Toon), pastas (identificadas) com imagens do Toon (PrintScreen) realizadas e ainda uma pasta com os ficheiros dos crash´s dos dias correspondentes.

Quando tivemos problemas técnicos as crianças tinham conhecimento, era inevitável, mas tentamos sempre que isso não constituísse um problema para o avanço do trabalho. Achámos relevante colocar alguns comentários das observações das crianças mediante a ocorrência desses acontecimentos.

Instalação do jogo Toon Talk

A instalação do jogo Toon Talk foi no dia 26 de Janeiro, neste dia o objectivo principal foi experimentar o jogo de forma a conhecer o que cada menino conhecia do jogo, pois alguns já tinham experimentado o jogo no ano anterior.

Avaliação/Comentário das crianças:

- Brincámos com os brinquedos da caixa de ferramentas.
- Explodimos casa.
- Vimos desenhos no caderno.
- Aspiramos a caixa.
- Vimos o pássaro a sair do ovo.

Registos:

Nome	Conhece o jogo	Manipula helicóptero	Sabe sentar e por em pé	Conhece os brinquedos	Gosta do jogo	Observações 26/02/04
Francisco	S	S	S	S	S	Tirou todos os brinquedos da Caixa de Ferramentas (CF)
Diogo	S	N	N	N	S	Tirou o caderno dos desenhos e perguntou como se tirava a CF. Retirou sempre o mesmo desenho (CF) colocando-os no chão. Afastou o caderno quando estorvava
Ricardo	S	S	N	N	S	Brincou com os brinquedos da CF, não reconheceu o robô como tal. Pôs a caixa dentro de um camião mas não ficou curioso porque se moveu. "E mais?" Foi às três casas e depois perguntou sobre a bomba e acabou por rebentar duas casa (gostou).
Ana Rita	N	Dificuldade			S	Dificuldade a localizar-se na cidade a voar e no chão.
Fábio	N (S)	S	N	S	S	Tirou os brinquedos da CF
David	N (S)	S (manipula bem o rato)	N	S	S	Pôs a caixa dentro de um camião mas não ficou curioso. Tirou os brinquedos da CF. Gostou dos pássaros e utilizou o aspirador. Explodiu casas. (? Carregou na tecla Q para sentar?)
Carina	S	S (manipula bem o rato)	N	S	S	Tirou os brinquedos da CF. Explodiu casas. Pôs a caixa dentro de um camião mas não ficou curiosa. Viu imagens. Muito independente.
Gonçalo	S	S	N	S	S	Tirou os brinquedos da CF. (tinha o caderno alterado)
Miguel	S	S	N	S	S	Aterrou imediatamente e brincou na rua, "quero ir para a casa". Quis mexer no caderno mas não sabia esfolhar. Retirou uma letra e depois foi para a pg do aspirador (caderno de imagens) e colocou o nº em cima da letra e o caderno foi para a pg da Bola (sem intenção). Achou piada. Tirou os brinquedos da CF.
Diogo Martis	S	S	N	S	S	Tirou os brinquedos da CF. Pegou na CF para ter mais espaço. Adorou as letras e nº. Por acaso colocou um nº por cima de outro e riu-se com o resultado. Repetiu várias vezes e fez leras+nº adorando o resultado, "D de Diogo".
Diana	N	N	N	Alguns	S	Tirou todos os brinquedos da CF. Disse que o robô é um monstro e não quis tocar, o ninho é comida e que o helicóptero é uma aranha
Ana	N	N	N	N	S	Tirou todos os brinquedos da CF, mas não mostrou interesse, nem curiosidade. Disse que o helicóptero é um avião.
Alexandra	N	N	N	N	S	Tirou todos os brinquedos da CF. Mostrou alguma curiosidade, disse que o robô é um pássaro.
Cátia	N	N	N	N	S	Tirou todos os brinquedos da CF. Achou que tem poucos brinquedos. "Só isto?"
Bruno	S	N	N	Alguns	S	Tirou todos os brinquedos da CF, mas não mostrou interesse. Reconheceu o helicóptero como tal.

Trabalho realizado com o grupo grande

Registos/Observações do dia 04/03/04 e 11/03/04

(Crianças aos pares)

Objectivos:

Fazer pares para jogar no Toon

Explorar o Toon

Fazer casas

Explodir casas

Explorar a bomba de ar, aspirador e a varinha

Colocar nome no telhado, parede exterior e interior da casa

Trabalhar com os pássaros e ver o que fazem

Nossas sugestões para as crianças:

"Podias construir uma casa só tua" "Queres por o teu nome no telhado/quarto para toda a gente saber que é tua?" "O nome está muito grande, como por mais pequeno?"

"O que é aquilo, um ovo? Dá qualquer coisa ao pássaro. Ele só aceita coisas quadradas. Vamos engana-lo (esconder o ninho ou levar o pássaro para outra casa)."

Não precisámos de sugerir a explosão de casa pois foi o que mais gostaram de fazer no dia de 26 de Janeiro.

Avaliação/Comentário das crianças:

Gostamos de explodir e construir casas, do helicóptero, de aspirar, tentar enganar o pássaro.

Explodimos casas. Andamos de helicóptero, aspirámos, construímos casas e tentámos enganar o pássaro "não conseguimos". Continuámos na Quinta-feira porque não houve tempo.

Escrevemos o nome no telhado.

Escrevemos Benfica na parede da casa. "Quero escrever Benfica na parede da rua"

Copiar pássaros: "quero mais passarinhos".

Nossa Avaliação:

A maioria das crianças gostou do jogo, existe um desinteresse por aquelas que não frequentam assiduamente o Jardim. As crianças mais novas de início não se mostraram muito entusiasmadas, mas no segundo dia perante as nossas sugestões ficaram vibrantes.

NOTA:

Muitos objectivos para trabalhar com todas as crianças. Continuamos na semana a seguinte com os mesmos objectivos e também com as crianças que faltaram.

Registos:

Nome	Observações individuais (dos objectivos dos dias 04 e 11/03/04)
Francisco	Gostou muito de aspirar.
Diogo	Gostou de por o nome no telhado.
Ricardo	Adora explodir casas, levou o pássaro para outra casa e gostou de ver o nome no telhado.
Ana Rita	Ainda se perdia na cidade.
Fábio	Faltou.
David	Adorou explodir casas; Brincou com as caixas e balança para ver qual o maior número, gostou de por o nome no quarto e telhado.
Carina	Tentou dar o pássaro ao pássaro. Anda muito bem na cidade.
Gonçalo	Faltou.
Miguel	Gostou de fazer casa e de explodir casas, não quis brincar com os pássaros.
Diogo Martins	Muito desembaraçado. Gostou de enganar o pássaro.
Diana	Gostou de ver o nome no telhado.
Ana	Adorou ver o nome no telhado.
Alexandra	Faltou.
Cátia	Não mostra interesse.
Bruno	Gostou de explodir casas, e somar os números.
Mara	(1ª vez que jogou) Gostou de construir, explodir casas e brincar com os pássaros.
Rafael	Não mostra muito interesse (foi embora quando o seu par estava a jogar). Perde-se na cidade.

Registos/Observações do dia 18/03/04

Objectivos:

Continuar a explorar as várias ferramentas do Toon

Nossas sugestões para as crianças:

"Para saberem que o pássaro é teu podes por o teu nome nele e no ninho", "É impossível enganar o pássaro"

"Podes enviar o que quiseres, se não for quadrado metes numa caixa e dás ao pássaro"

Avaliação/Comentário das crianças:

Tentamos enganar o pássaro, levamos o pássaro para outra casa. Escrevemos no Toon e mandamos mensagens.

Nossa Avaliação:

As crianças começam a gostar mais do jogo, como já se sentem inseridas no jogo começam a querer fazer. Algumas esquecem como se fazem algumas coisas (de Quinta para Quinta).

NOTA:

Algumas das crianças, ao colocar o nome no telhado, colavam letra a letra. Pensavam que só podiam escrever uma letra de cada vez (como só tem a letra A…)

Registos:

Nome	Observações individuais (dos objectivos de 18/03/04)
Francisco	Faltou
Diogo	Faltou
Ricardo	Fugia com o ninho para ver se o pássaro conseguia colocar a mensagem. Colocou 3 pássaros só para um ninho (sobrepôs os ninhos). Escreveu o nome no pássaro e no ninho. Aumentou e diminui com a bomba de ar um camião e a caixa de ferramentas.
Ana Rita	Tentou enganar o pássaro.
Fábio	Faltou.
David	Lembrava-se de algumas coisas que já tinha feito, escondeu um ninho do pássaro.
Carina	Explodiu casas e levou o pássaro para a casa do Miguel. Recebeu a mensagem pelo pássaro ao Miguel (foi verificar se tinha chegado a casa).
Gonçalo	Pôs o nome no telhado.
Miguel	Explodiu casas, fez casa e pôs o nome no telhado. Enviou uma mensagem à Carina.
Diogo Martins	Explodiu casas, levou o pássaro para outra casa.
Diana	Explodiu e fez casas.
Ana	Pôs o nome no telhado.
Alexandra	Pôs o nome no telhado e queria que o pássaro ficasse no telhado (perguntar à Irina o que fez)
Cátia	Explodiu casas e tentou enganar o pássaro levando-o para outra casa.
Bruno	Explodiu casas.
Mara	Faltou
Rafael	Fez casas, gostou do camião.

Registos/Observações dos dias 01/04/04, 29/04/04, 13/05/04 e 20/05/05

Objectivos:

Vamos continuar a explorar as várias ferramentas do Toon

Fazer casas

Explodir casas

Explorar a bomba de ar, aspirador e a varinha

Colocar nome no telhado, parede exterior e interior da casa

Trabalhar com os pássaros

Nossas sugestões para as crianças:

"Para saberem que o pássaro é teu podes por o teu nome nele e no ninho", "É impossível enganar o pássaro"

"Podes enviar o que quiseres, se não for quadrado metes numa caixa e dás ao pássaro"

Avaliação:

Não houve tempo, tivemos muito que fazer e não foi possível ir para o Toon.

Registos/Observações do dia 27/05/04

Objectivos:

Vamos continuar a explorar as várias ferramentas do Toon

Nossas sugestões para as crianças:

"Para saberem que o pássaro é teu podes por o teu nome nele e no ninho", "É impossível enganar o pássaro"

"Podes enviar o que quiseres, se não for quadrado metes numa caixa e dás ao pássaro"

Avaliação/Comentário das crianças:

"Pega-se na letra A põe-se no nº 1 por cima: já está, mudou a letra"

"Vou mas é explodir a casa"

"Para a casa é a caixa, o camião e o senhor" (Diogo)

Registos/Observações do dia 09/06/04

Objectivos:

Vamos continuar a explorar as várias ferramentas do Toon

Avaliação/Comentário das crianças:

"Gostei de destruir a casa" (Bruno e Gonçalo)

"Gostei de brincar com os pássaros" (Ana Isabel)

"Gostei de brincar com as letras e os números" (Alexandra)

"Gostei de arrebentar com a bomba" (Diogo Matos)

"Gostei de explodir a casa" (Cátia)

Trabalho realizado com o grupo pequeno

Proposta

Nas actividades livre reunimos com a Educadora e com quatro crianças, propusemos a construção da Cidade das profissões no Toon Talk. Cada criança escolheu a profissão (das que nós demos a conhecer: Carteiro João, Jardineiro Jasmim, Padeira Brites, e Teresa Limpeza, inseridas no projecto "Vamos Descobrir Portugal"). As quatro crianças que escolhemos foram aquelas que se mostraram mais interessadas no jogo.

- Miguel: Carteiro João
- Carina: Padeira Brites
- Carina: Teresa Limpeza (Cantoneira de Triagem)
- Diogo: Jardineiro Jasmim

Ilustração 1 Reunião como pequeno grupo.

A Cidade na fase inicial (no CD pasta "Cidade26-02") é só constituída por duas casas, uma é a casa dos desenhos onde as crianças vão buscar as imagens que necessitam, a outra casa é a chamada casa das construções, é a partir desta casa que as crianças constroem as outras casas. Optámos por construir esta casa pois verificámos que se as crianças construíssem as casas através da casa dos desenhos haveria um problema relativamente à decoração. A decoração da casa dos desenhos iria aparecer no caderno da nova casa e poderia confundir a criança desnecessariamente numa fase inicial.

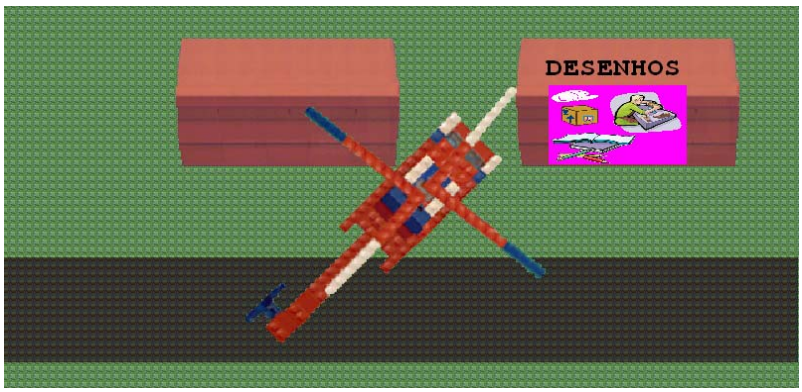


Ilustração 2 Imagem da Cidade (na fase inicial).

Registos/Observações do dia 29/04/04

Objectivos:

Explorar a Cidade

Construir a casa para a sua profissão

Nossas sugestões para as crianças:

"Para poderes trabalhar terás de construir uma casa."

"Como é que os outros meninos vão saber que é a tua casa?"

Avaliação/Comentário das crianças:

"Construí a casa e dei-lhe nome." (Carina)

"Tentei enganar o pássaro." (Miguel)

Registos/Observações do dia 13/05/04

Objectivos:

Explorar a Cidade

Construir a casa para a sua profissão

Por o nome no telhado e na parede exterior da casa

Nossas sugestões para as crianças:

"Para poderes trabalhar terás de construir uma casa."

"Como é que os outros meninos vão saber que é a tua casa?"

Avaliação/Comentário das crianças:

"Construí a casa para a minha profissão e dei-lhe nome." (Miguel)

"Tentei enganar o pássaro." (Rita)

Registos/Observações do dia 20/05/04

Objectivos:

Explorar a Cidade

Construir a casa para a sua profissão

Por o nome no telhado e na parede exterior da casa

Nossas sugestões para as crianças:

"Para poderes trabalhar terás de construir uma casa."

"Como é que os outros meninos vão saber que é a tua casa?"

"Quando aterras não se vê o telhado, como é que os outros meninos vão saber que é a tua casa?"

Avaliação/Comentário das crianças:

"Gostei de por o nome na casa." (Rita)

"Gostei de escrever o nome o telhado" (David)

Registos do Toon:

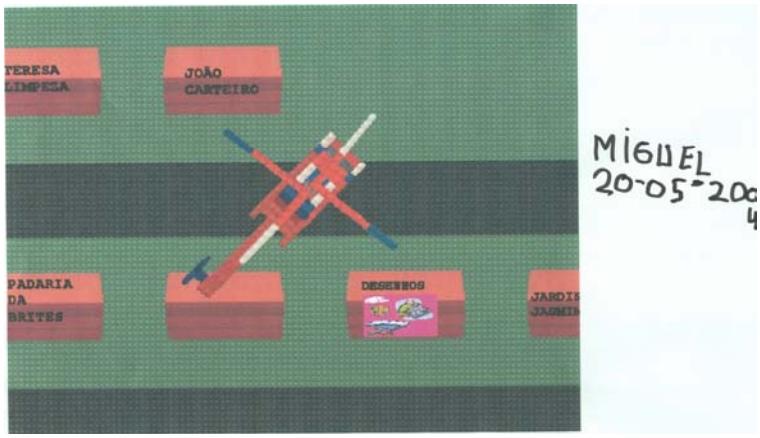


Ilustração 3 Registo da casa (vista do telhado) do Miguel, datado e assinado.

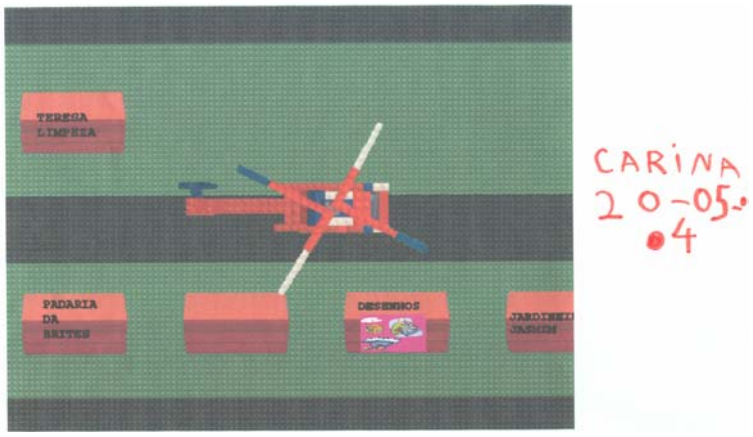


Ilustração 4 Registo da casa (vista do telhado) da Carina, datado e assinado.

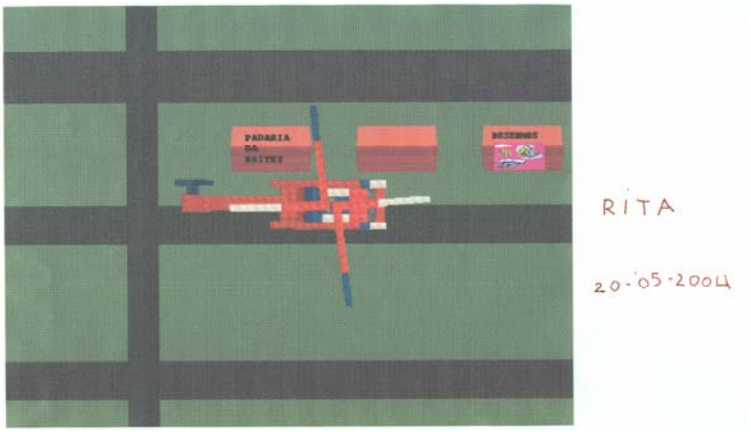


Ilustração 5 Registo da casa (vista do telhado) da Rita, datada e assinado.

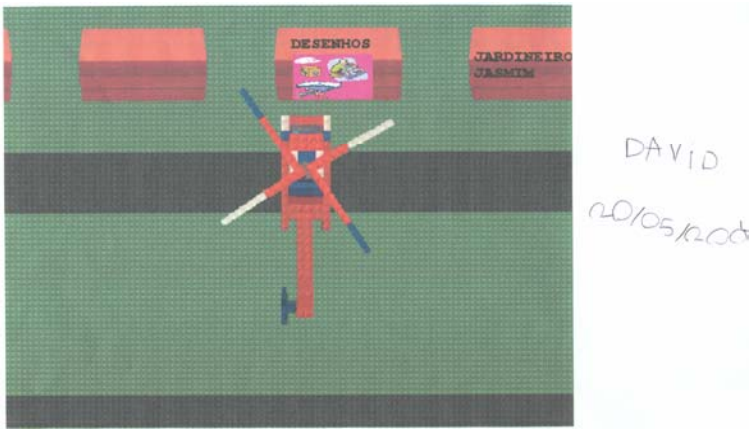


Ilustração 6 Registo da casa (vista do telhado) do David, datado e assinado.

Registos/Observações do dia 27/05/04

Objectivos:

Explorar a Cidade

Colocar no telhado e na parede exterior da casa o desenho que as crianças fizeram no Paint

Decorar a parede interior da casa numa moldura a fotografia do dono da casa

Nossas sugestões para as crianças:

"Sabes que podes por na casa os teus desenhos e a tua fotografia?"

"E se colocasses a tua fotografia dentro de casa? Assim se alguém se enganasse e entrasse sabe que está na casa errada."

"Olha, tens a casa toda desarrumada, é melhor limpar a casa."

Avaliação/Comentário das crianças:

"Vamos por o meu desenho no telhado para saberem que é a minha casa" (Rita)

"Gostei de por o desenho da minha profissão na casa." (Carina)

"Gostei de por o nome João Carteiro no telhado e o desenho do João Carteiro na parede." (Miguel)

"Gostei de por no telhado o desenho da Padeira Brites." (Rita)

"Gostei de por o desenho do Jardineiro Jasmim na casa." (David)

Registos do Paint:

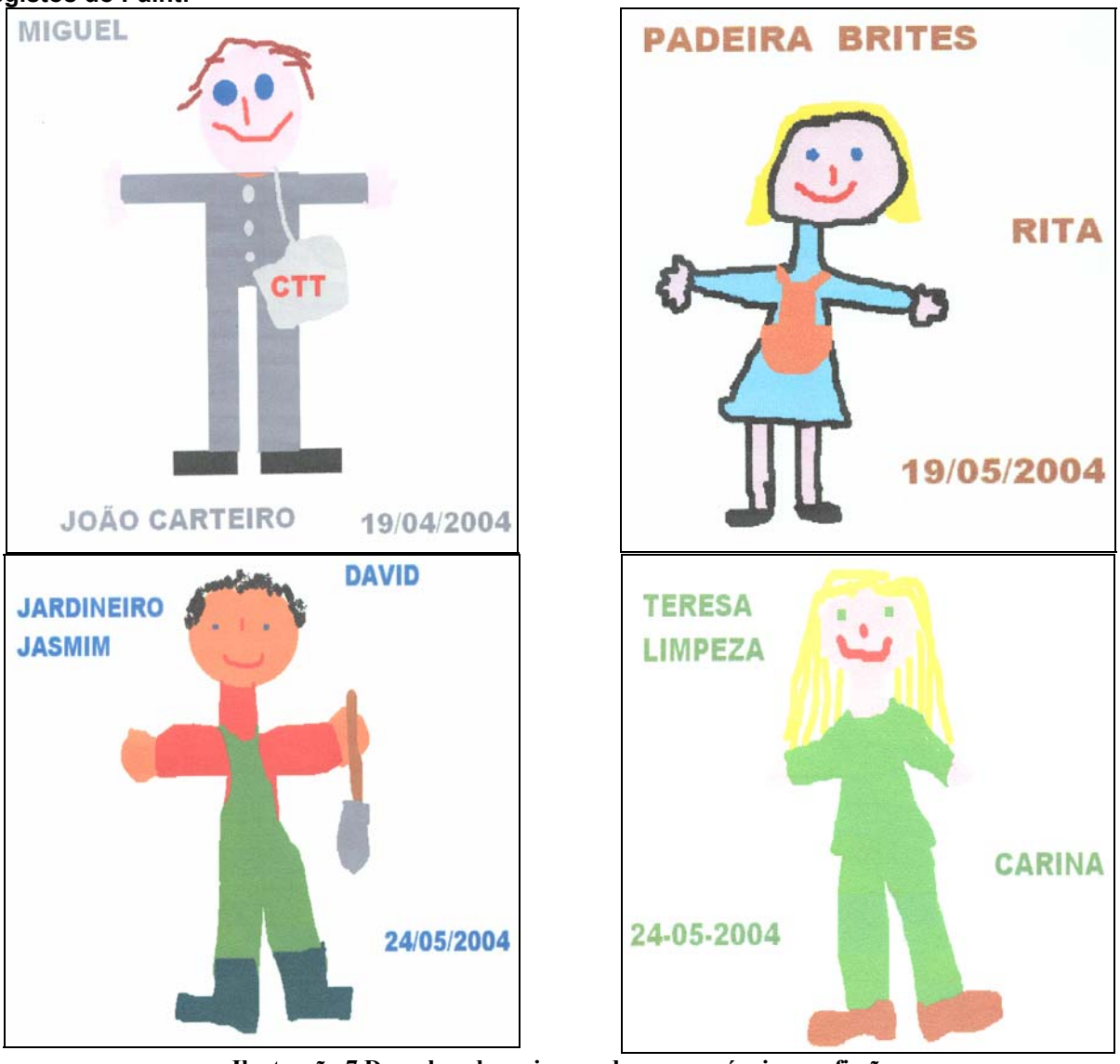


Ilustração 7 Desenhos das crianças das suas próprias profissões.

Registos do Toon:

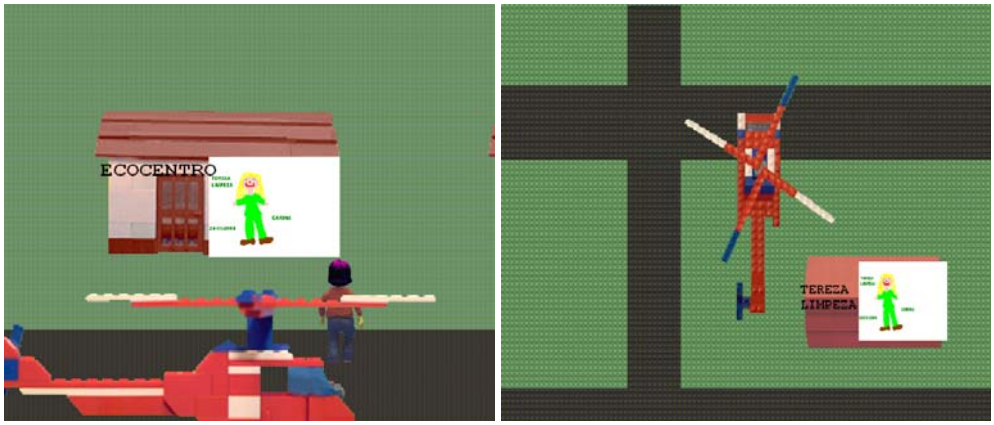


Ilustração 8 Casa da Carina (Teresa Limpeza) vista do telhado e da parede exterior.

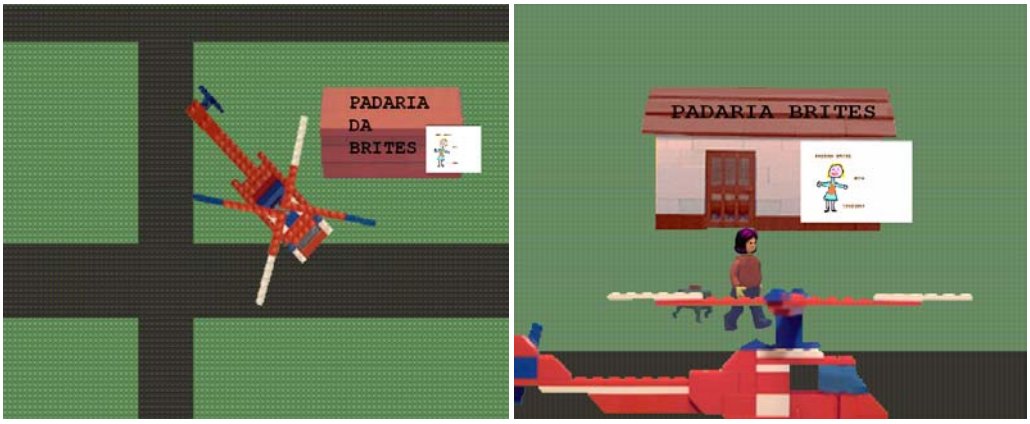


Ilustração 9 Casa da Rita (Padeira Brites) vista do telhado e da parede exterior.

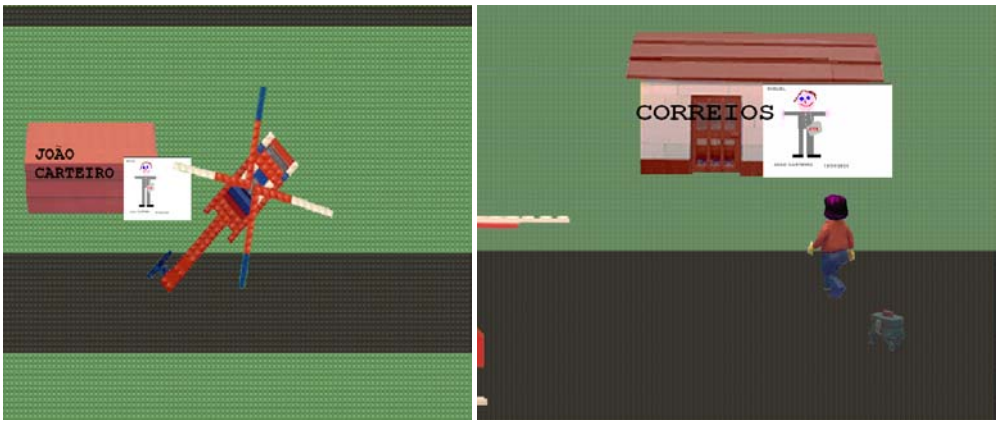


Ilustração 10 Casa do Miguel (Carteiro João) vista do telhado e da parede exterior.

Ilustração 11 Casa do David (Jardineiro Jasmim) vista do telhado e da parede exterior.

Registos/Observações do dia 09/06/04

Objectivos:

Arrumar a casa, aspirar tudo o que está no chão que não seja necessário

Pôr a fotografia numa moldura e na parede do quarto

Colocar o guarda-roupa na casa e a roupa de cada profissão

Colocar o armário para as ferramentas e produtos de cada profissão

Nossas sugestões para as crianças:

"Para poderes trabalhar precisas da tua farda, a roupa de trabalho."

"Para guardar a roupa precisas de um guarda-roupa"

"Vamos ter de por um armário para guardar os produtos e ferramentas da tua profissão"

Avaliação/Comentário das crianças:

"Pus o armário na casa para por as cartas, faltam os envelopes." (Miguel)

"Falta o rolo da massa." (Rita)

"Oh, o computador está lento" (Rita e Miguel)

"Gostei de por a roupa no guarda-roupa." (David)

"Gostei de por a fotografia na parede." (Miguel)

"Consegui por a fotografia na moldura." (Carina)

"Gostei de por a roupa no armário" (Rita)

Nossa Avaliação:

Tivemos alguns problemas com o jogo pois algumas imagens não eram reconhecidas e o computador começou a ficar muito lento devido à opção da viagem no tempo. Tentámos retirar esta opção, mas o resultado era o desaparecimento de todo o trabalho realizado com as crianças até ali, apreciam somente duas casas (a dos desenhos e a casa das construções).

Para resolver o problema das imagens não reconhecidas colocaremos os ficheiros dentro do ficheiro Cidade.

Registos/Observações do dia 18/06/04

Objectivos:

Colocar as ferramentas e os produtos no armário para poder abrir o negócio

Continuar a arrumar a roupa

Nossas sugestões para as crianças:

"O armário é muito pequeno para lá colocares tudo."

"Já temos a roupa, as ferramentas e os produtos, estás pronto para trabalhar."

Avaliação/Comentário das crianças:

"Gostei de por as botas no guarda-fatos." (David)

"Gostei de por as coisas no sítio, envelopes e caixas." (Miguel)

"Gostei de ir buscar o ecoponto e as ferramentas para separar o lixo" (Carina)

"Encolhi os pães e o bolo" (Rita)

Nossa Avaliação:

Algumas das imagens foram feitas pelas crianças (roupa e ferramentas) de forma a ultrapassar o problema das imagens não serem reconhecidas na Cidade, foi uma boa opção (apesar de demorada) pois as crianças gostaram mais de utilizar os seus próprios desenhos.

Registos do Toon:

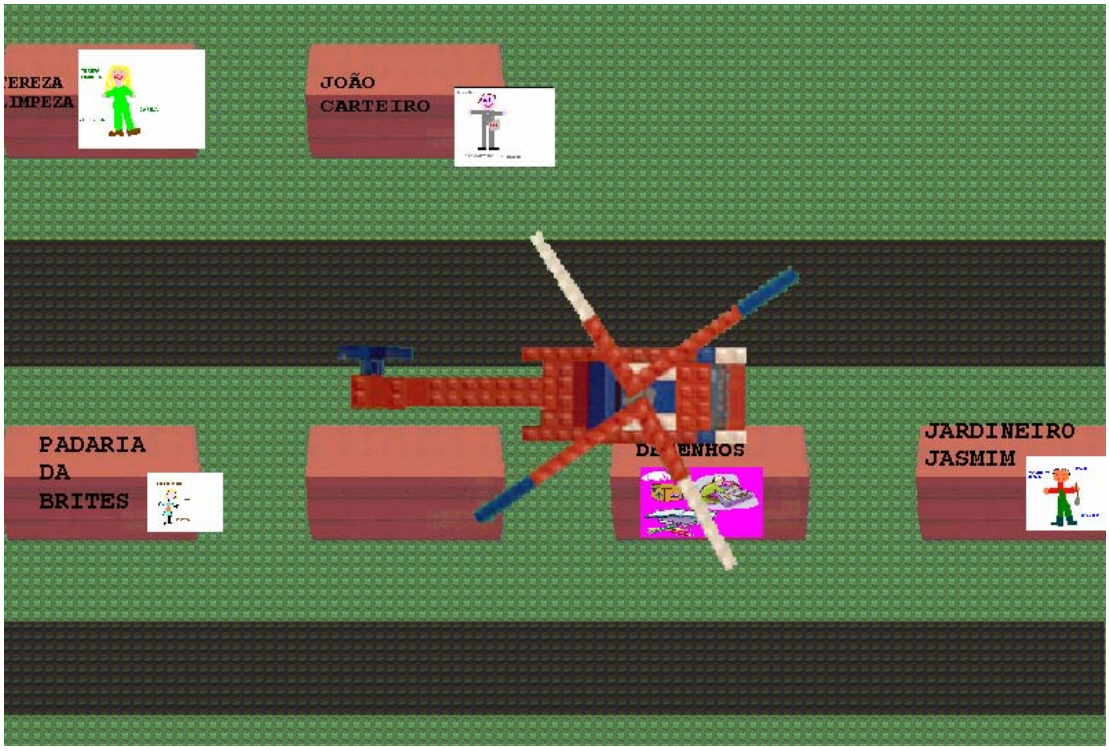


Ilustração 12 Vista da Cidade (telhados).

Ilustração 13 Casa da Rita (Padeira Brites) vista da parede exterior e interior da casa.

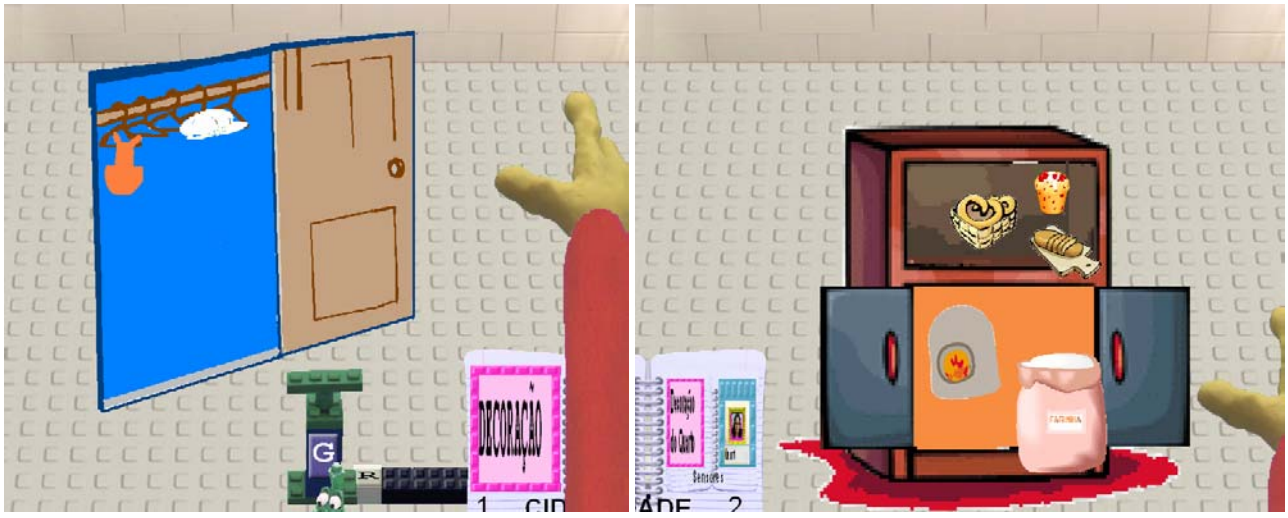


Ilustração 14 Casa da Rita: armário de ferramentas (baixo) e produtos (em cima) e guarda-roupa.

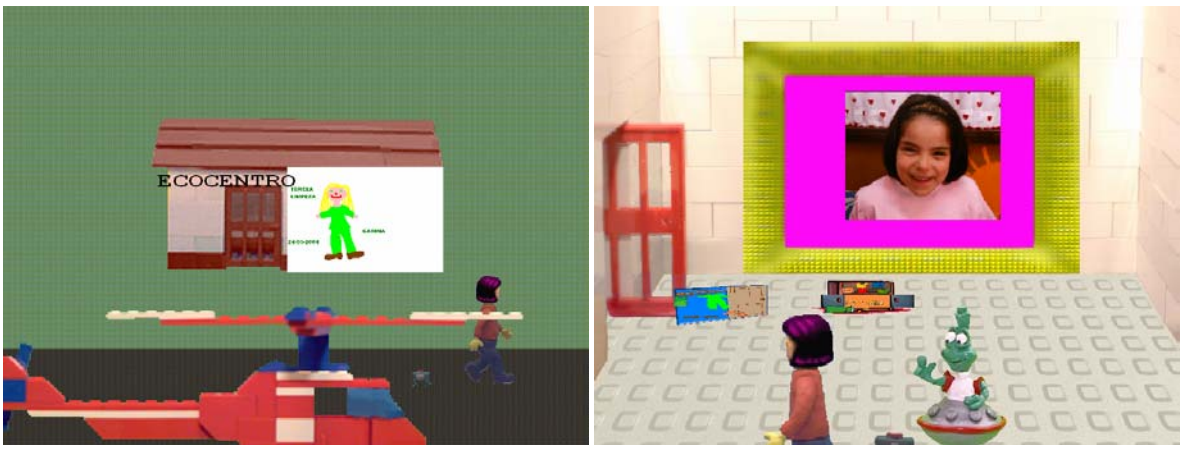


Ilustração 15 Casa da Carina (Teresa Limpeza) vista da parede exterior e interior da casa.

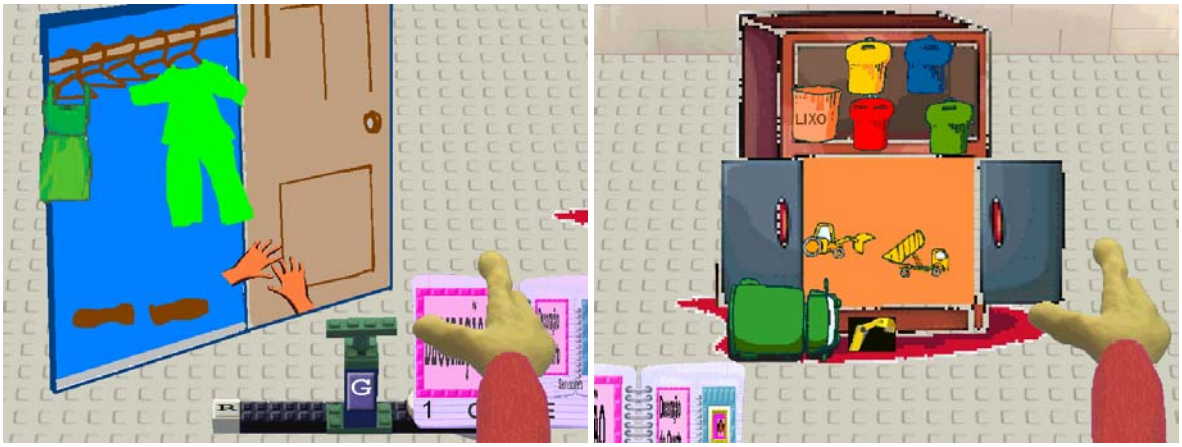


Ilustração 16 Casa da Carina: guarda-roupa e armário de ferramentas (baixo) e produtos (em cima).

Ilustração 17 Casa do David (Jardineiro Jasmim) vista da parede exterior e interior da casa.

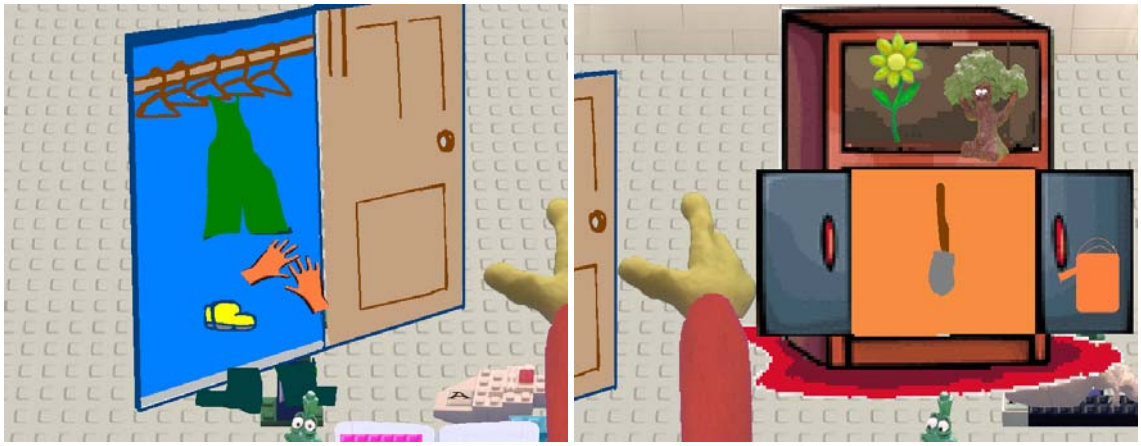


Ilustração 18 Casa do David: guarda-roupa e armário de ferramentas (baixo) e produtos (em cima).

Ilustração 19 Casa do Miguel (Carteiro João) vista da parede exterior e interior da casa.

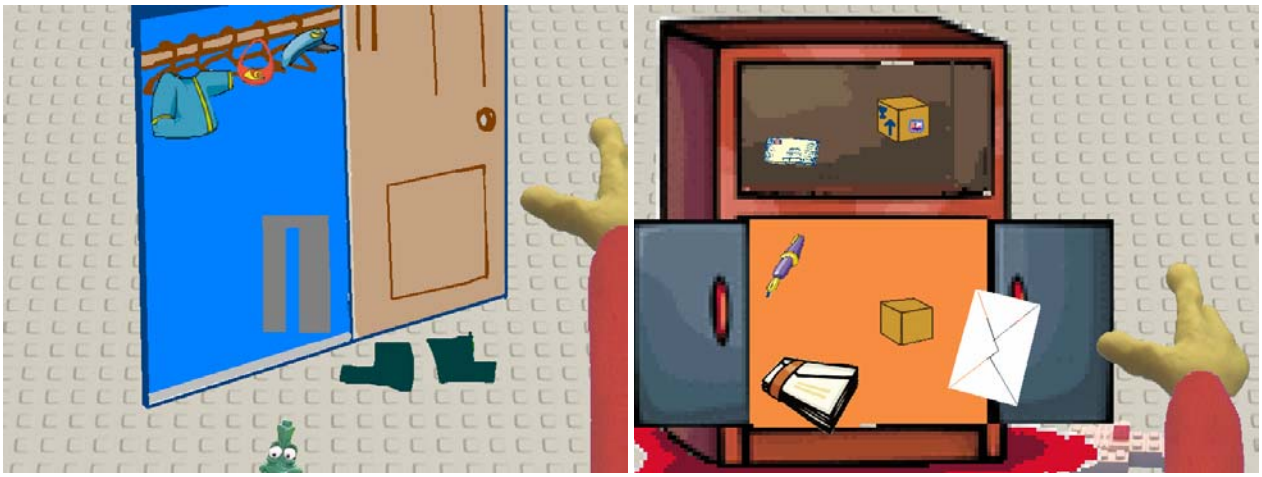


Ilustração 20 Casa do Miguel: guarda-roupa e armário de ferramentas (baixo) e produtos (em cima).

Registos/Observações do dia 21/06/04

Objectivos:

Limpar a casa

Começar a distribuir os produtos

Nossas sugestões para as crianças:

"Como vamos distribuir os produtos? Se calhar com o que não conseguíamos enganar. O que era?" _ o pássaro

"Vamos deixar o pássaro na casa dos outros meninos para eles poderem fazer as encomendas."

"Como é que os outros meninos vão saber que é o teu pássaro?"

"Como é que os outros meninos vão responder aos teus pedidos? É melhor levares um pássaro teu."

"Os pássaros e o ninho vão ficar no chão? Se calhar é melhor arruma-los, vamos procurar na casa dos desenhos."

Avaliação/Comentário das crianças:

"Descobri que ao colar dois desenhos não precisava de caixas para levar os desenhos." (Rita)

"Tem de se ter cuidado com os pássaros porque o que lhes aparece à frente levam tudo." (Rita)

"Gostei de pedir as encomendas, pedi 3 bolos à Padeira." (Miguel)

"Gostei de por o pássaro na casa dos meninos." (Carina)

Nossa Avaliação:

A Rita fez as suas descobertas por acaso pois é a que tem mais dificuldades a manusear o rato (apesar de se notar uma melhoria), em compensação, ao largar os objectos sem querer, descobre coisas novas.

A disposição dos armários e o tipo de armários para arrumações foram opções das crianças, logo cada interior da casa tem um cunho pessoal. As crianças gostaram muito de identificar os seus pássaros, ficaram muito admiradas.

Colocámos imagens em cada casa para as crianças poderem escolher e colocar nos respectivos armários (Cidade20-06Melhorada).

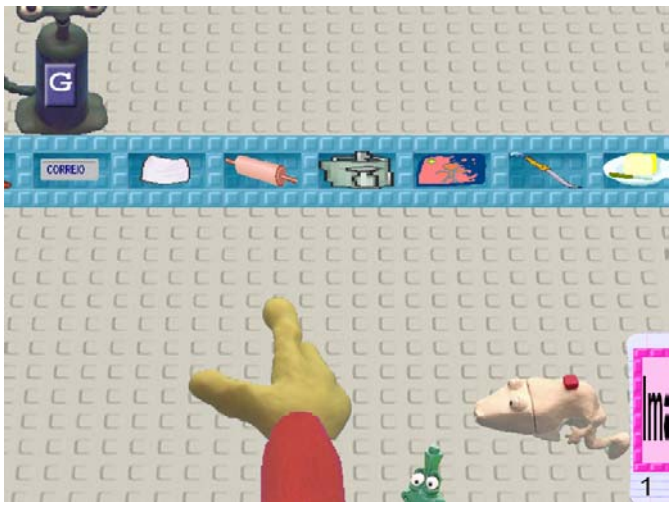


Ilustração 21 Algumas imagens para as crianças escolherem colocadas nas casas.

Registos do Toon:

Ilustração 22 Interior da casa do David (Jardineiro Jasmim), vista geral, mesa dos pássaros, guarda-roupa e armário de produtos e utensílios.

Ilustração 23 Interior da casa da Rita (Padeira Brites), vista geral, mesa dos pássaros, guarda-roupa e armário de produtos e utensílios.

Ilustração 24 Interior da casa da Carina (Teresa Limpeza), vista geral, armário dos pássaros, guarda-roupa e armário de produtos e utensílios.

Ilustração 25 Interior da casa do Miguel (Carteiro João), vista geral, armário dos pássaros, guarda-roupa e armário de produtos e utensílios.

Registos/Observações do dia 22/06/04

Objectivos:

(Manhã)

Verificar se temos encomendas na nossa casa

Responder aos pedidos

Fazer novas encomendas

(Tarde)

Pedir os contentores

Nossas sugestões para as crianças:

Para o Miguel: "Como podes enviar cartas se os outros meninos não têm caixas de correio? Podias fazer um surpresa e enviar as caixas de correio para eles colocarem na porta."

Avaliação/Comentário das crianças:

"Já tenho os três pássaros!" (Rita)

"Recebi uma encomenda!" (Carina) – 1 embalão da Padeira

"Mandei dois selos e duas caixas que a Padeira pediu." (Miguel)

"Vou buscar um armário para guardar os pedidos. Vou a pé." (Carina)

"Gostei de arrumar as encomendas que me fizeram." (Rita)

"Gostei de pedir as encomendas à Carina." (Miguel)

"Gostei de por a caixa de correio na porta" (Carina)

Nossa Avaliação:

Deveria ter sido primeiro a Rita a jogar porque era a única que tinha encomendas e seria mais fácil de recomeçar a partira daí, mas para não destabilizar o andamento no Jardim teve de ser o Miguel. Daí surgiu a ideia de enviar as caixas de correio para os outros meninos.

O David faltou surgindo um espaço no tempo de cada menino, dai surgiu a ideia de na tarde pedir os contentores do ecoponto para eventualmente colocar no exterior.

As crianças estão cada vez mais independente no jogo e daí surgem novas ideias.

Como todos têm problemas com rato, ao andar de armário em armário dentro de casa pois não havia área da mesa suficiente: "levanta o rato e vai fazendo festas à mesa ". Resultou e houve melhorias consideráveis na utilização da área da mesa ao deslocar o rato.

Tivemos problemas com a decoração das casas, daí a repetição ou desaparecimento dos nomes na parede exterior, problema que não conseguimos resolver. Quando se ia colocar algo de novo para decorar a parede, as decorações anteriores não apareciam, pensamos esse problema apareceu devido à instalação da nova versão do Toon Talk.

Registos do Toon:

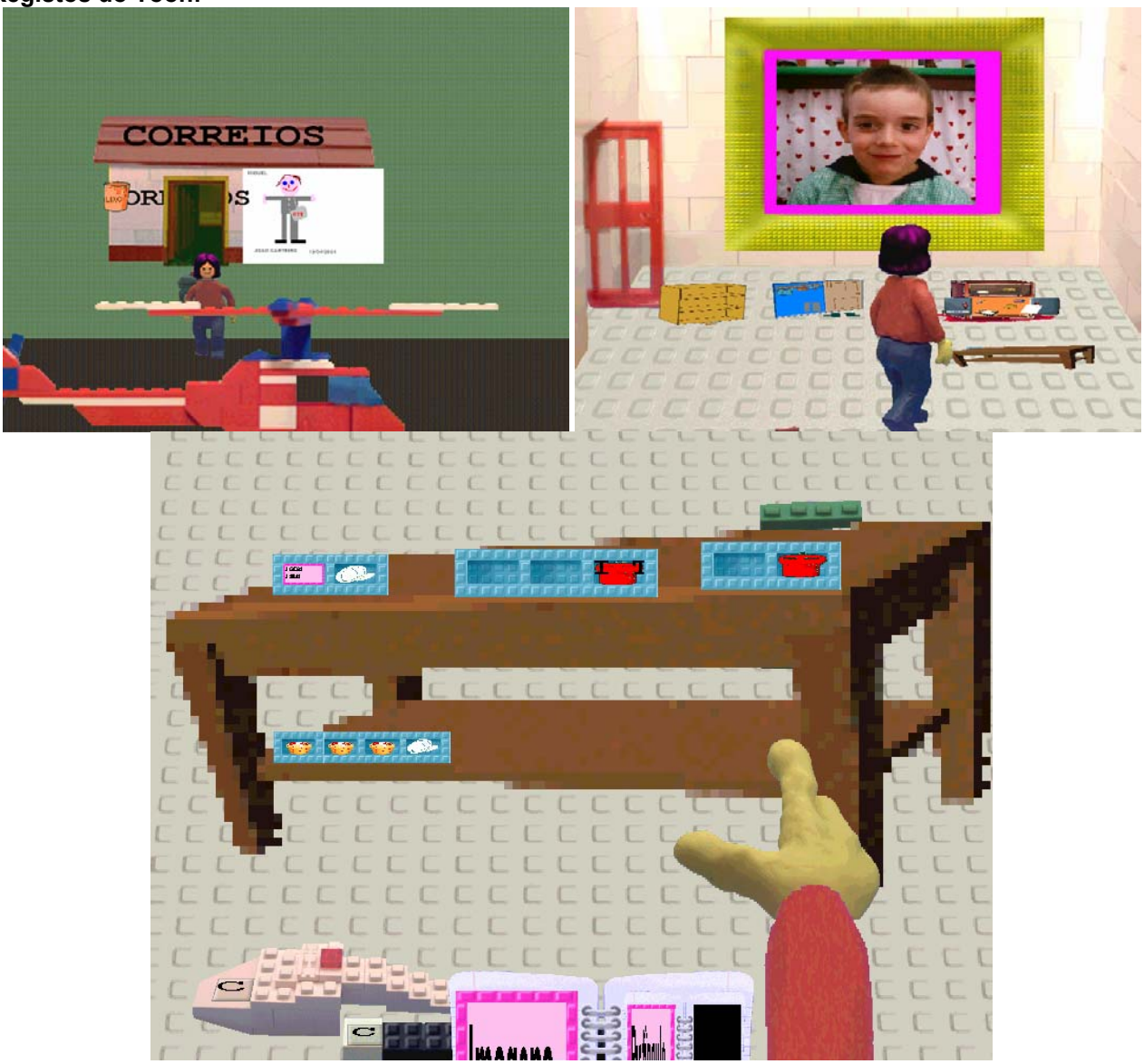


Ilustração 26 Casa do Miguel (Carteiro João), vista do exterior e interior e mesa dos pedidos.

Ilustração 27 Casa da Rita (Padeira Brites), vista do exterior e interior e armário dos pedidos.

Ilustração 28 Casa da Carina (Teresa Limpeza), vista do exterior e interior e mesa dos pedidos.

Registos/Observações do dia 23/06/04

Objectivos:

Continuar a responder aos pedidos

Colocar o ecoponto na parede exterior da casa.

Nossas sugestões para as crianças:

Avaliação/Comentário das crianças:

"Quero por o pinheiro na rua." (Carina)

"Quero tirar as letras que não preciso [da parede exterior]." (Miguel)

"Gostei da Carina mandar os ecoponto". (Miguel)

"Gostei de por a caixa de correio na porta." (David)

Nossa Avaliação:

As crianças estão cada vez mais envolvidas e entusiasmadas, adoram responder e encomendar, quiseram enviar com as encomendas pedidas textos "Aqui está …".

Registos no Toon:

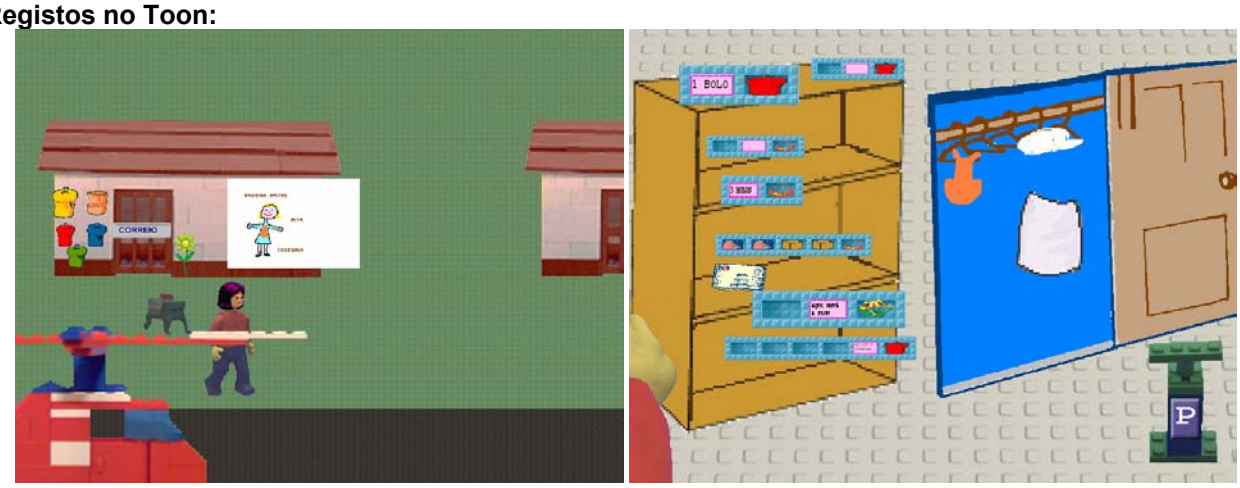


Ilustração 29 Casa da Rita (Padeira Brites), vista do exterior, armário dos pedidos e guarda-roupa.

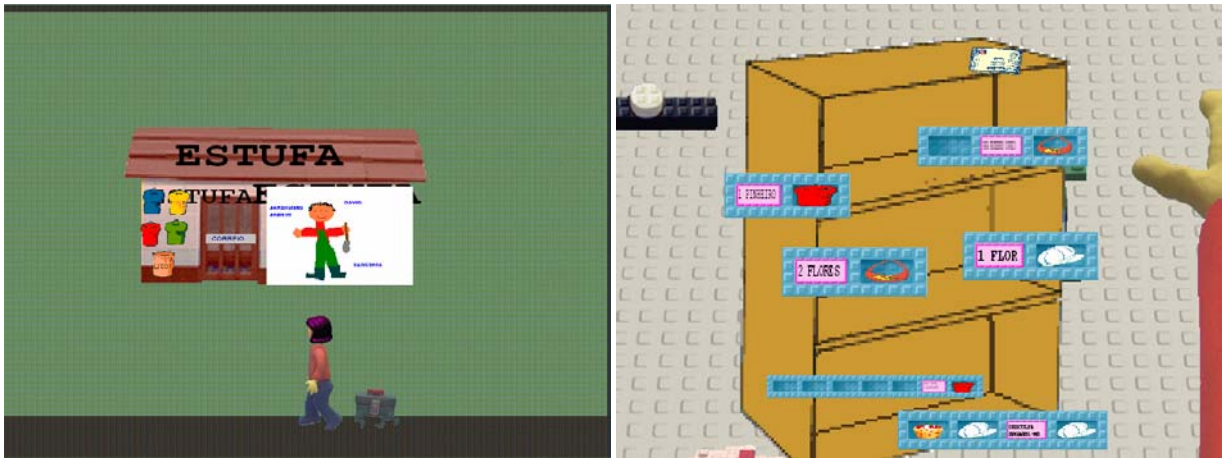


Ilustração 30 Casa do David (Jardineiro Jasmim), vista do exterior e armário dos pedidos.

Ilustração 31 Casa da Carina (Teresa Limpeza), vista do exterior e mesa dos pedidos.

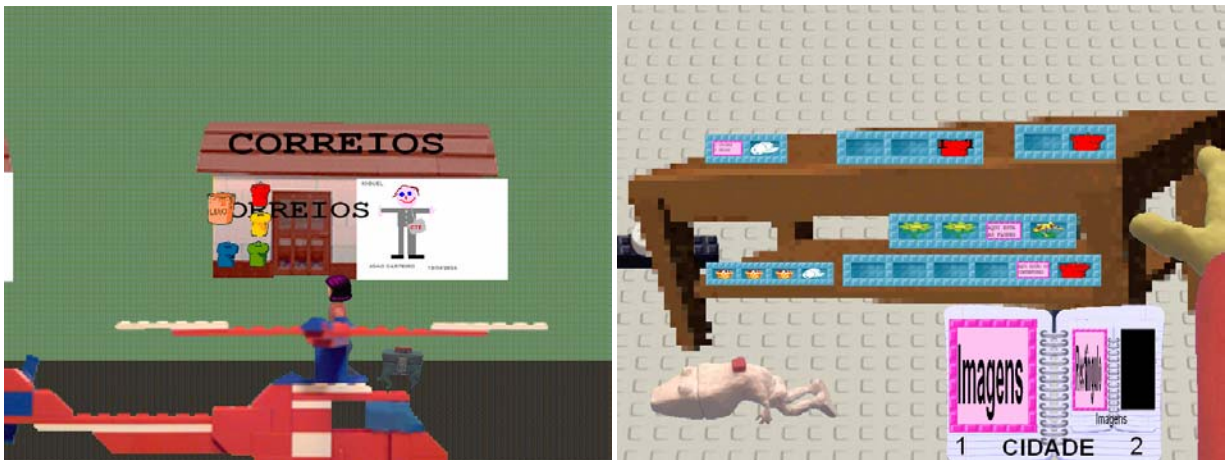


Ilustração 32 Casa do Miguel (Carteiro João), vista do exterior e mesa dos pedidos.

Registos/Observações do dia 24/06/04: Jogo final

Objectivos:

Livre

Nossas sugestões para as crianças:

"Vamos às compras à casa dos desenhos" (para enviar o lixo para a Teresa Limpeza).

Avaliação/Comentário das crianças:

"Quero por o ecoponto na rua." (Carina)

"Gostei de por o ecoponto no telhado." (Carina)

"Gostei de por Padeira Brites na casa." (Rita)

"Gostei de mandar o pacote de leite à Carina." (Miguel)

"Gostei de dar ao pássaro uma carta para o Carteiro enviar à Padeira."

Nossa Avaliação:

O trabalho era para ter sido terminado no dia anterior, mas como as crianças estavam a aderir muito bem e a gostar decidimos continuar por mais este dia, daí a falta de preparação de objectivos.

Começamos por trabalhar com o David (Jardineiro), este foi à casa dos desenhos e pegou numa das imagens colocadas recentemente, uma caixa de cartão amassada, surgiu a ideia de enviar o lixo para a Teresa (Carina) para depois se poder reciclar. A Rita e o Miguel fizeram o mesmo, cada um escolheu o lixo do livro dos desenhos e enviou para a Carina (Teresa).

O David também enviou uma carta para a Padeira (imagem enviada pelo Carteiro) através dos CTT.

O Miguel enviou a Carta do David para a Padeira (Rita) e enviou um pacote de leite para a Teresa.

A Rita guardou a carta e escolheu uma lata de sardinhas ("gosto muito de sardinhas") para a Teresa.

Registos do Toon:

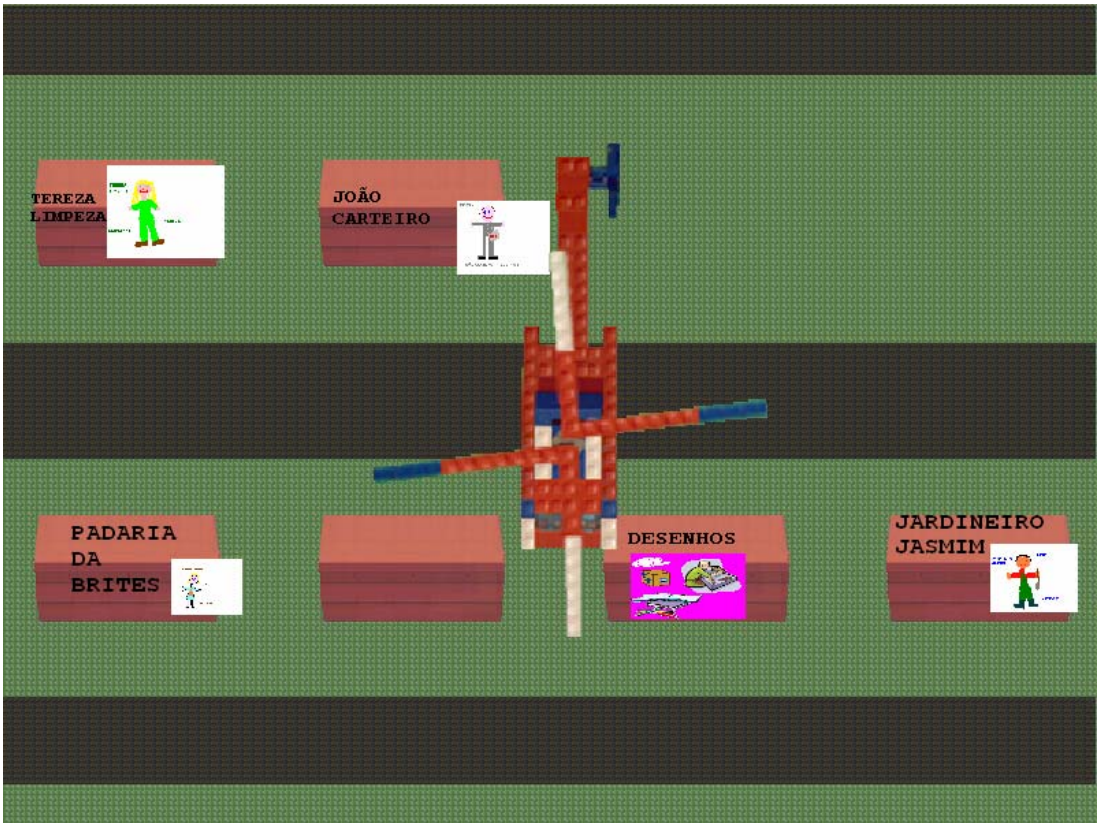


Ilustração 33 Vista da Cidade (dos telhados).

Ilustração 34 Casa da Rita (Padeira Brites) vista do exterior.

Ilustração 35 Casa do David (Jardineiro Jasmim) vista do exterior.

Ilustração 36 Casa da Carina (Teres Limpeza) vista do exterior.

Ilustração 37 Casa do Miguel (Carteiro João) vista do exterior.

Ilustração 38 Lixo para enviar à Teresa Limpeza (Carina).

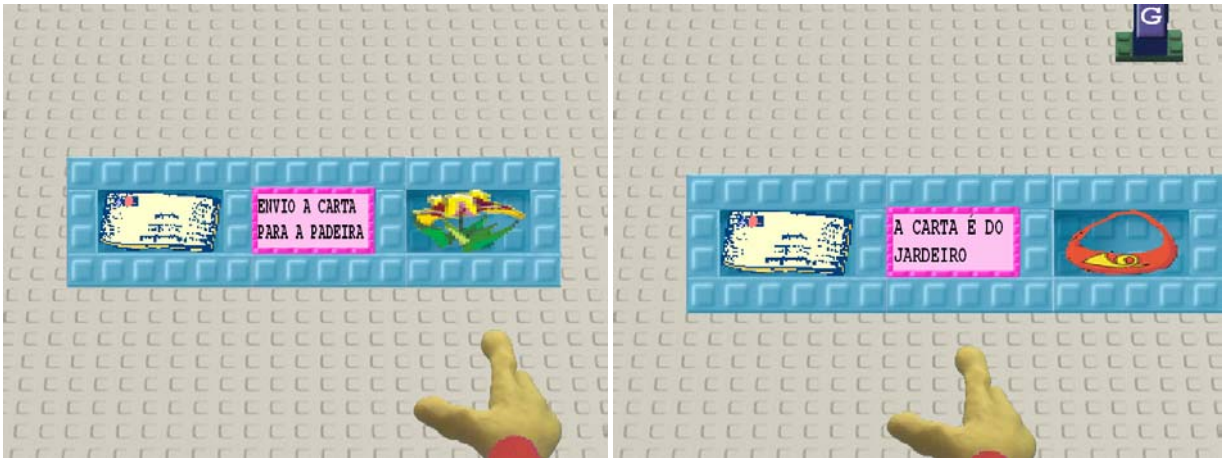


Ilustração 39 Envio da carta do Jardineiro Jasmim (David) para a Padeira Brites (Rita) através dos CTT (Miguel).

Conclusão/Avaliação

Ao finalizarmos este trabalho pensamos que conseguimos realizar um bom trabalho com as crianças, todos nós aprendemos e divertimo-nos muito. Constamos que as crianças gostaram do jogo desde o início e que não jogaram tantas vezes como o desejariam.

As principais dificuldades que sentimos na realização do trabalho foi a falta de tempo, uma vez por semana era muito pouco para avançar com o trabalho, as crianças no início esqueciam-se de algumas coisas.

Quando se trabalhou com as crianças (pequeno grupo) dias seguidos notou-se um grande avanço e com isso um crescente entusiasmo. As crianças interiorizaram mais facilmente o objectivo da Cidade, ao início ainda estavam um pouco confusas. Perto do final, já tinham memorizado as teclas F1, F2, etc. e assim chamavam os instrumentos que necessitavam sem lhes pegar; quando necessitavam de algum desenho dirigiam-se à Casa dos Desenhos e mostravam um grande à vontade de deslocação na Cidade, iam a pé de um lado para o outro sem necessitar do helicóptero para se orientarem.

Grande Grupo

- Gostámos de jogar no Toon Talk.
- Aprendemos a jogar com as teclas.
- Explodimos as casas.
- Com o helicóptero passeávamos para baixo e para cima.
- Brincámos com o pássaro e tentámos enganá-lo, mas não conseguimos.
- Com a bomba fazíamos as coisas pequenas e grandes.
- Aspirámos as coisas com o aspirador e cuspimos.
- O Rato Marretas colava as coisas.
- Gostámos do jogo e foi fácil jogar.

Pequeno Grupo

- Fizemos muitas trocas: bolos, flores, pinheiro, ecoponto.
- A Rita enganou-se na encomenda, mandou para o Jasmim, pediu desculpa e depois enviou para a Teresa.
- Conseguimos arrumar todas as casas. Ao princípio foi difícil mas depois arrumamos tudo.
- Gostávamos de continuar a jogar.
- Já sabíamos as teclas todas para jogar, só precisávamos de ajuda para escrever.

Annex X
—
Logs of the activities conducted in a home setting in Medford, MA, USA

NOTE: these reports were provided by the adult, F. Berthold, via e-mail. N is a five-year-old child, as mentioned in section 5.3.8, on p. 338.

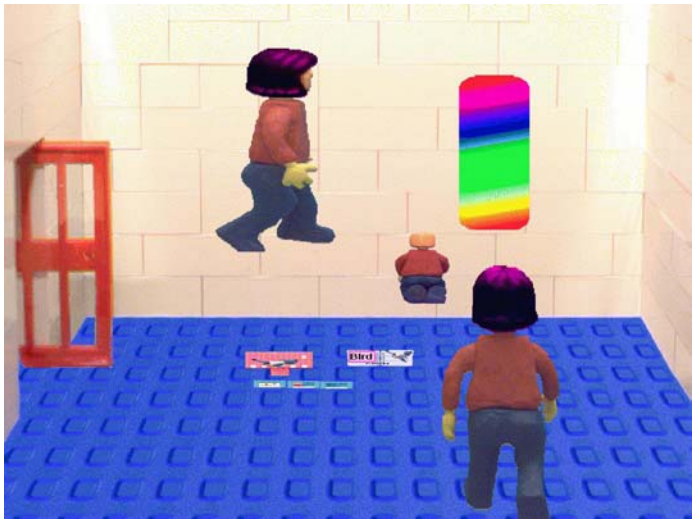
Game History:

N generally plays games such as "Blue's Treasure Hunt" and "Putt-Putt joins the Race." On weekends he also plays Zangband, "Planet Hot-wheels" and most of the Yahoo games.

Intent:

To make N. aware of and comfortable with the basic tools in toontalk by August 25, when he is to receive his own copy of TT as a gift.

7-23-04 6:40pm-7:15pm



Plan:

Introduce N. to the basic interface of toontalk. Show how houses can be decorated and decorate one with his assistance.

Reasoning:

Get N. comfortable with and interested in the toontalk.

Start:

N. was slightly irritable and tired at the start, he'd had a fairly long day.

Lesson:

In this lessen we decorated two houses together. In the first house I decorated while he suggested what pictures to place. On the second house he decorated the inside, but grew tired and wanted me to finish. I ended up showing him more than I'd initially intended. Along with the basics of adding pictures, I also taught him how to use Pumpy, Dusty, navigating a notebook and removing items from it and the basic idea of typing in letter tiles. He learned each of the basic concepts behind these fairly quickly, and had no trouble understanding that the thing you wanted to effect should be shaking. He had some trouble understanding the difference between littler and shorter. He had little trouble using the helicopter, and obviously found it intriguing, though he was somewhat uneasy dealing with navigation in general. He showed considerable trepidation regarding moving the hand past the apparent bounds of the screen and had to be encouraged to navigate the floor.

N's Questions/Observations:

Where's Tooly's mouth? <He came to the conclusion it must be on the top>

Why is the house door open when we're inside?

We're stepping on our things! <The things left on the floor, he was a touch alarmed>

End:

He was obviously excited, and was interested in continuing this form of play.

His words:

He repeatedly commented that it looked just like a game. After playing and describing it to his parents, he described Marty as the green guy who tells you what to do when you don't understand something, Pumpy as the purple guy who makes things wider or narrower, taller or shorter, big or little and the robot, who he'd seen in the picture on the CD, as being the blue guy you teach to do things (this came from my description, we'll probably start with the robot next week).

My Thoughts:

Though I showed more than initially intended none of it has any bearing on the weekend's work as presently planned. He's picking it up well and was fairly comfortable doing it with a little hand holding, it'll be interesting to see how he progresses.

7-24-04 4:05pm-5:10pm

Plan:

Introduce the mechanism for editing pictures in toontalk.

Reasoning:

Show continuity from previous styles of play to toontalk.

Start:

N was a little riled up today, he'd been rather sick the night before and slept a bit late.

Lesson:

Today we decorated another house, this time the theme was to be a garden with flowers inside, vegetables outside and implements on the top. We started out looking for flowers for the inside, but there was only one in the notebook. So I then walked him through the process of taking a black square and editing it to be a drawing of his choice. He was able to pick up the process fairly quickly. On getting three different kinds of flower I showed him how to make many of the same thing using the wand. Interestingly he didn't seem nearly as amused by this as he was by using Pumpy to make them larger and smaller.

I also showed him how to use a truck to build a new house.

I had intended for the session to only be half an hour long, but I'm obviously trying to squeeze too much into the time since it took over any hour before we were done. He grew somewhat bored with the project midway through the vegetables and had to be redirected towards it more than once.

He spent a fair amount of time exploring the limits of the world, finding how far his hand could reach on the floor, whether he could get into the house by walking through the wall next to the door, seeing how far he could walk in the world itself, and how far the helicopter could fly. He's obviously become more comfortable with the helicopter and was a little irked to find that he couldn't land on the house, but tried for about 5 minutes.

He seems to be particularly interested in Tooly, he spent a while seeing how Tooly followed him. Maggie just didn't seem to interest him.

He seemed to get a little frustrated when dealing with the mouse because it frequently required him to either pick up the mouse and put it back to keep it on the pad, or to use the entire desk as his surface for the mouse, he generally chose the latter.

He also wanted to look through each step of the process of making the new pictures in detail, wanting the screen where TT asks what you want to do with the picture in hand to be read to him from the top down. This is typical behavior for N.

End:

Still riled up.

N's Questions/Observations:

Why can the toolbox walk on the water? (at the bottom edge of the world)

Hey, I can walk through the wall!

His Words:

What did we do: We made a new house and we had to decorate it, that what our project was today.

Did you learn anything: We learned how to build a house.

My Thoughts:

He seems to be growing bored fairly quickly, part is simply is present mood, but I think I need to add more variety with what we're doing. I might add the robots tomorrow instead of waiting until next week, I'll have to think on it. Other possibilities would be to add a variety of small projects to what we're doing.

7-25-04 12:00pm-2:30pm

With a break from 12:30 to 1:15 for lunch and Gilligan's Island.

Plan:

Teach the basics of training a robot. Intended task is to make the robot count and go through the alphabet.

Reasoning:

Start a new form of play in toontalk and introduce the beginning of programming.

Start:

He was in a fairly relaxed mood today.

Lesson:

I started out showing N the basics of programming a robot by teaching a robot to count up, introduced how to deal with when the robot expected something that it wasn't holding. He understood these decently well, we then used the scale to teach it to count to 10, and then 100. He rapidly picked up the idea of how the scale works, comparing how the scale looked to holding his hands either even or with one above the other.

We then started making a 'supprise' for his mother we did this by first making a background picture and programming a robot to layer several other pictures on top of it. Despite this process taking significantly longer than the other projects he remained actively interested until the end.

End:

Somewhat excited.

His Words:

My Thoughts:

I suspect there were a couple of factors involved with his continued interest, the first was the intent to give his mother a present. Second because of the combination of putting the picture together and programming the robot there was enough variety to keep him amused despite the added complexity.

7-30-04 2:30pm-4:15pm

Plan:

Today is going to be focused more on inspiring that instructing. I'm going to demonstrate some of the small simple things that can be done in toontalk that are still amusing.

Reasoning:

The main idea is to inspire N. to continue playing with Toontalk, to insure that it doesn't become a chore.

Start:

N. was in a decent mood today, feeling alert and in the mood for playing.

Lesson:

He'd shown interest in the tutorials a couple of times, so we started with the introductory tutorial, both because it was of some interest and as a bit of a review. He was amused by most of it, though as we got to the last section he grew bored with it. I then walked him through the recursion exercise where a robot duplicates itself twice, sends them copies to make houses, then blows up its own house. He found this amusing and wanted to repeat it again midway through the session, I had no problem with this, and expect it will not be the last time we do. I then started a project to make a ball "bounce" between y=500 and y=700 using four robots, two to make the robot go up or down and two to switch from going up or down when the robot has reached the appropriate height, this ended up being significantly more time consuming than anticipated due to debugging. The final project was having two robots say "hi" to each other, initially just using birds and nests to send the word back and forth, then to also drop it onto a text to speech box so they really said the word. I demonstrated that they could do this from an extended distance by moving one of the robots to a different house.

End:

He was restless afterwards. I suspect that this was due to a combination of the heat, it's been a hot day and the room itself is hot, and the length of the session.

My Thoughts:

Overall this was a fairly productive session, I think I achieved my intended goal, but not in the way I'd planned. The surprising result was that N. actively interested in the debugging process. For a programmer this is often considered the most frustrating task, for N. it was a natural extension of play, and in fact probably the closest we have done to the play he is accustomed to. This is partly because of an average American boy's tendency towards playing fix-it with toy tools etc. Also he and I have a history of "debugging play" where, for example, a toy train will not run on the track properly, normally this is a cause for distress but we've turned this into a game of it's own by exploring what has occurred to cause it and discussing it. Because of this, it is safe to say today was one of the most successful sessions in terms of getting and holding his interest. Instead of having to judge when he was too bored to continue I had to choose to stop because of time constraints and judging how tired he was. When I ended the session we were still having trouble getting the "talking robots" working, we'd started having them say longer phrases and they were vacuuming them up prematurely, and he told me he wanted to go back to fixing them tomorrow. The major draw back to this kind of session is he was not directly involved, though he gave verbal input, and I made a point of asking his opinion in the problems he made no direct action himself, but I think at this stage it is more important that he become familiar with the system and learn to enjoy working with it than that he push though it because he should be getting involved.

7-31-04 2:00pm-3:00pm

Plan:

Continuing on yesterday's theme. Today I'm going to make a fairly complicated program with his assistance. We're going to make make a flower that grows in several stages.

Reasoning:

We're going to continue to do moderately complex, interesting programs.

Start:

Fairly alert, interested in play.

Lesson:

Due to time constraints todays session was somewhat shorter. Today we started by attempting to make a flower that grew in several stages. Unfortunately we were stopped at an early stage by a bug in toontalk. The experience in general was similar to yesterday's experience. He took a slightly more active roll, operating some of the tools and helping to edit a picture of a leaf. We then started a more organized version of the recursive program for blowing up the buildings. Here the buildings would be built one at a time and when a set number of buildings was reached they would all be blown up at once. Unfortunately this did not work out as hoped due to a lack of understanding of how the robots deal with nests on my part. We then proceeded to work on the standard recursive blowing up buildings program, which he watched until we ended the session.

End:

Neutral

His Words:

We were changing a leaf and blowing up the city.

My Thoughts:

Today's session was a strong reminder of the importance of testing out ideas before presenting them in a learning situation. Working on the fly can be ok for small parts of a project, but it is very important to practice in advance before trying it with a child. Though I think N. enjoyed the session and learned some from it, he clearly gained less than he would have if I'd been prepared both for the error in toontalk and had full understanding of how the given program would perform when run.

08-07-04 2:00pm-3:00pm

Plan:

Construct robot that will cause a rectangle to cycle through all of its colors.

Reasoning:

This is a fairly simple project to execute, but N is still likelly to find it to be amusing.

Start:

N was in a good mood.

Lesson:

We worked through the process of creating a robot that would just add to the "picture #" sensor of a rectangle repeatedly and stuck it on the back of the rectangle. N thought it was neat, but it flickered too fast for his taste, we then went about the process of putting a delay in. We had little trouble doing this and were able to change the speed to his satisfaction. He then said that he wanted to make it so it would only do it once, we decided to hold off on this project for another day.

End:

Slightly hyperactive.

My Thoughts:

I'm particularly excited by the fact that this is the first time that N has taken a creative interest in what we're doing it. It suggests that he's both become sufficiently comfortable with the medium to imagine what it can do and remained sufficiently interested to want to impliment it.

08-08-04 1:50pm-3:20pm

Plan:

Continue yesterday's project by making the color square cycle only once. I'm going to attempt to get N more directly involved in the creative process now that he's becoming more comfortable with the system.

Reasoning:

To help to focus his present interest.

Start:

He was in the mood to be paid attention to.

Lesson:

As planned we progressed through the process of making a color square cycle only once, with N doing more of the work. I made a point of at each step asking his opinion of what we should do to solve a given problem. I also had

him controlling the system directly for most of the project. Along with making the square cycle only once, we also made it so the number of the color was displayed on the square and its color was one step out of sync with the color of the square insuring that it was always visible(to some extent)

End:

Slightly more insistent on the attention.

N's Questions/Observations:

Why is everything made out of lego's? (I responded that it was because we built programs with it, to which he responded "But lego's are a toy.")

My Thoughts:

A good session, he navigated the system well, and is able to at least to some extent think in toontalk. He was noticibly disturbed by the fact that robots on the back of pictures don't always look like they're working properly. In this case, though there were two robots that were doing their job on the back of the picture they both had boxes that contained red squares.

08-09-04 3:20pm-4:10pm

Plan:

Another attempt at the robot team that distributes other teams one at a time, then blows up all of the houses at once.

Reasoning:

The amusement of both N and myself.

Start:

He was pretty relaxed today.

Lesson:

Today's was intended largely to be mostly me working and talking through as I did it. We wrote up the program with minimal trouble, other than there is presently a glitch somewhere that causes TT to freeze. Unfortunately there's still something in the program that doesn't quite work as anticipated and though we were able to produce all of the houses, the final explosion didn't occur.

End:

A little impatient.

N's Questions/Observations:

My Thoughts:

You would think that I would have learned after last weekend, you'd be wrong. I thought I had this figured out, but hadn't planned on a couple of glitches in TT itself, but these border on random. Mostly, I still need to get my own head wrapped around the TT paradigm. The previous two days were carefully planned, today was intended more to be just tossing together an amusing toy to play with, I have to remember in these circumstances, even the toys need to be carefully planned.

Annex XI
—
Sample activity sheet for preschool teachers: Storage Levels

Activity: storage levels (notebooks)

Goals

This activity aims to explore the use of notebooks for storage and organization, helping children understand the utility of such processes.

The example of integration in a preschool room is based on the "Tree Day" theme.

Source and authorship

This activity started as an original idea from students Elisabete Gonçalves, Ercília Carocha and Sónia Carvalho, of the Computers in Teaching course, from the 2nd year of the Early Childhood Education baccalaureate, University of Trás-os-Montes and Alto Douro, Portugal, 2003-2004. The adjustments, corrections, and final adaptation for use in preschool rooms were made by the lecturer, Leonel Morgado.

Initial activity

Considering that a sequence of activities based on "Arbour Day" is taking place in the garden, this suggestion is that initially a set of drawings is assembled, each drawing covering a small feature of the theme. A global, theme-wide picture can also be produced. ToonTalk is used to store all images. The final result may (and should) be somewhat disorganized, as demonstrated in figure 1.

So that the background image doesn't get in the way of the activity development, it can be glued to the ground. This initial activity allows children to develop basic manipulation activities (pick, drop, enlarge, shrink, etc.).

Figure 1 – Collection of images about the theme

Figure 2 – Separated images

Initially, the images can be separated, to make some sense out of them. For instance, in figure 2, they were separated into Animals, Tools, and Fruits.

Other possible developments

During the initial activity, it is common for children to come across several ToonTalk features that are not fundamental regarding manipulation training: number pads, letter pads, trucks, birds, etc. Under the constructionist philosophy of education supported by computer-programming, such casual interest that children may demonstrate for these elements is to be supported and explored. This may even lead to a complete redesign of this activity.

I.e., this activity must be seen as a mere strategy, aiming to conduct children on a path of understanding of the advantages of using notebooks to organize and store information.

The **ideal result for this strategy** would be that **the children themselves**, during of after the activity, would come to **use the notebooks in other (new) situations**; either directly or verbally proposing their use to solve other problems or situations.

Central activity

Storing things in notebooks

Having the pictures on the floor all the time, taking up a lot a space, isn't the most practical way to store them. Truth be told, they are more readily accessible, but they can also easily be lost, vacuumed, "ruined" (by being drop on one another), etc.

These disadvantages are examples of motives that can be used to introduce the notebooks as a storage solution. Figure 3 shows the use of 3 notebooks to save previously separated images. A piece of text was used to describe each image, but that wasn't absolutely necessary.

Figure 3 – Storing the pictures in notebooks

Developing: notebooks within notebooks

Even so, three notebooks also take up a lot of space. The next phase of this activity is to explore the disadvantage of having the images in three notebooks, in order to propose the integration of these in a single one (*vd.* figure 4), to store the full activity.

Figure 4 – Notebooks inside a general notebook

Completion: exploring complexity

The conclusion of this activity deals with the increase in complexity of this concept (which similar, for instance, to having folders within folders, in the computer's hard disk). The notebook with the full activity, for instance, can and should be saved in the global notebook (the one with the children's names – figure 5). That makes three levels of storage.

It is recommended that diverse strategies and approaches are explored, and the results observed. Can children find what they want? Can they explain where it is? Can they, in order to save the picture of a new animal, find the animals notebooks and store the new animal inside? Etc.

Figure 5 – Activity notebook inside the global notebook

The complexity can also be increased regarding notebook use, as long as there are enough images for that: the animals notebook can be split into notebooks for insects, mammals, reptiles, etc.; the tools notebook can be split into gardening and farming tools; etc. (*vd.* figure 6).

Figure 6 – Multiple notebooks

Final comments

Since an image archive is being produced, its very important for it to be used in other activities. If there is a printer available, for instance, the images can be used in other off-computer activities of the preschool room. They can also, obviously, be used in other computer or ToonTalk-related activities.

Finally, it should be stressed that the aim is to use **this activity as a method for exploration**, to try and find, **for each children**, strategies that allow her to understand "navigation" through a multi-level organizational structure, becoming **capable of creating** and **following it**, in order to obtain what she wants.

Annex XII
—
Sample activity sheet for preschool teachers: communication channels (birds)

Activity: communication channels (birds)

Goals

The activity aims to explore the use of birds to provide communication, helping children understand the utility and process of establishment of communication channels.

The example of integration in the preschool room is based around professions, and the fact that most professions are interlinked: the postman needs bread; the baker needs letters; etc.

Source and authorship

This activity began as an original idea by students Irina Brandão, Liliana Miguéis and Mónica Pereira, of the Computers in Teaching course, from the second year of the Early Childhood Education baccalaureate, University of Trás-os-Montes e Alto Douro, Portugal, 2003-2004. The adjustments, corrections, and final adaptation for use in preschool rooms were made by the lecturer, Leonel Morgado.

Initial activity

Considering an activity sequence in the preschool room, under the theme "professions", this activity proposes that all children play in the same ToonTalk city. It is assumed that each child plays the role of a professional, taking responsibility for the decoration and preparation of a house for that professional (example: figure 1).

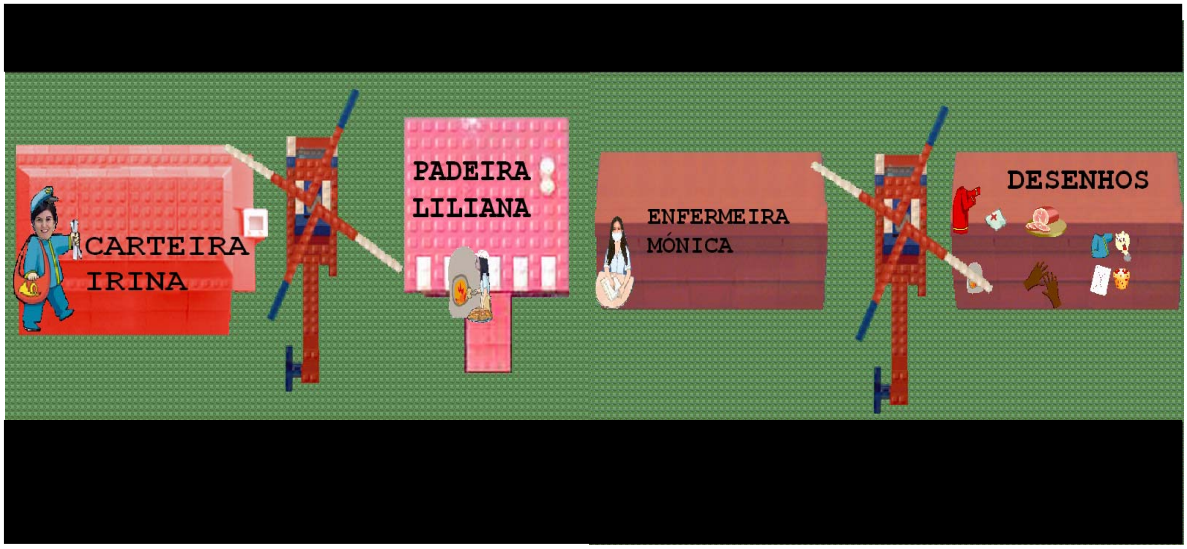


Figure 1 – City with houses for three professionals (plus a "pictures" house")

Each house should have a "warehouse" or "products" section, related to its professional (*vd.* figure 2).

Products can be generic images, photographs, children-made drawings, etc. These can be turned into infinite stacks, so that the activity can take place in a more straightforward manner. For this very reason, background images can be glued to the house floor.

This initial activity allows time for children to develop basic manipulation skills.

Figure 2 – Postman's products

Other possible developments

While the initial activity takes place, it's likely for children to notice several ToonTalk features that are not fundamental regarding manipulation training: number pads, letter pads, trucks – and birds. Under the constructionist philosophy of education supported by computer programming, such casual interest that children may demonstrate for these elements is to be supported and explored. This may even lead to a complete redesign of this activity.

I.e., this activity must be seen as a mere strategy, aiming to conduct children on a path of understanding of the advantages of creating and using communication channels (using birds).

The **ideal result for this strategy** would be that **the children themselves**, during or after the activity, would come to **use the birds in other (new) situations**; either directly or verbally proposing their use to solve other problems or situations.

Central activity

Announcing services

When a professional is ready to provide services to the community, the activity can move on into this phase. It's the teacher's role to decide when that time comes. Examples: when there are enough products; when, besides having products, the house is properly decorated; when besides products and decoration, there is a uniform; etc...

In order to provide services to the community, it is proposed that in this activity the children playing a professional's role will ANNOUNCE those services, using **his/her** bird. *I.e.*: he/she creates a bird, makes copies of it, and distributes them around town. So that its purpose is understood as a service-request point, the placement of a bird can be played, theatre-like, by saying, for instance "*I'm the baker. For when you want to order something from me, here's a bird: use it to place an order.*"

As each professional announces available services, at each house several birds will become available (figure 3).

In this picture, birds have the name of a profession on their shirts. Actually, it's preferable for birds to have an image identifying the service, instead of text.

Figure 3 – Performing an order

Performing an order

When, due to an activity taking place in the preschool room, a product becomes necessary, any child can order it from the adequate professional. Each order is composed, at a minimum, by a box with a description of the request, and the identification of the requester.

Figure 3, shows an order, which is being placed by the mailman (identified by the mailman's hat): the mailman is requesting "1 bolo" (1 cake). So probably the request will be handed to the baker's bird.

Notice that the text "1 cake" could be replaced by the Picture of a cake (drawn by the requesting children, for instance).

Receiving requests

In figure 3, a nest is visible. In that figure, it was the mailman's nest, since he/she was placing the order. If we had been at the baker's, that nest would belong to the baker.

When, at a later time, the baker visits the ToonTalk city, he/she will find in the nest the other children's requests: a cake, half a dozen breads, etc.

The strategy for organizing these requests is up to the teacher and child. In fig. 4, the choice was to place all requests in a cupboard, to fulfil them afterwards. The nurse ordered a pastry (the second request). But there are requests nor meant for the baker: a package, a stamp, etc...

The child and the teacher must decide how the orders will be satisfied. For proper requests (for the baker: bread, cakes, etc.), the necessary products must be gathered. For inadequate requests, a decision must be made: ignore them? Reply with text, stating that they can't be satisfied? Forward them to the adequate professional? The decision process is a nice opportunity for reflection for exploring...

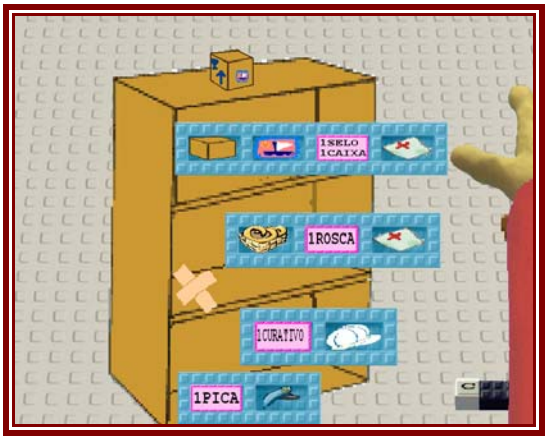


Figure 4 – Requests after distribution

Order delivery

Thus, the moment of delivery is reached. This can occur in two ways.

1. If the child that issued the order (the nurse, for instance) still hasn't distributed his/her birds, the only way is for the baker to walk up to the nurse's house, to deliver the order.
2. If there's a bird available, from the ordering child, it can deliver the order (the fastest and most convenient way) or still make the delivery by foot.

Potential developments

After integrally exploring this process, there is no mandatory need to abandon it. Several ideas may have arisen throughout the activity, worth exploring.

A few sample possibilities were considered:

- **Buying and selling.** Parcels may require a payment, which in turns allows the introduction of sending and receiving money; a record of outstanding debts may be kept; a record of payments made and settled accounts; a bank can be created in the city; etc.
- **Automatic dispatching.** The nest where requests arrive may be handled by a robot. This can be taught by the children, to perform some activity automatically, for each received request, for instance: update a request list; provide automated replies; etc.
- **Quality control.** The delivery of a parcel can include a "copy of the request", so that the receiver may check the order contents, to see if they are right. For instance: a hole for the order, another hole for the original request.

It should be noted that except for the "automatic dispatching", the remaining activities can also be made in the preschool room, without using ToonTalk or the computer; they are way of exploring the computer activity by linking it with more physical activities, for instance.