The Human Controller: Usability and Accessibility in Video Game Interfaces

by

Eitan M. Glinert

Submitted to the Department of Computer Science and Electrical Engineering
on May, 2008 in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Computer Science and Electrical Engineering
at the Massachusetts Institute of Technology

ABSTRACT

Despite the advances in user interfaces and the new gaming genres, not all people can play all games – disabled people are frequently excluded from game play experiences. On the one hand this adds to the list of discriminations disabled people face in our society, while on the other hand actively including them potentially results in games that are better for everyone. The largest hurdle to involvement is the user interface, or how a player interacts with the game. Analyzing usability and adhering to accessibility design principles makes it both possible and practical to develop fun and engaging game user interfaces that a broader range of the population can play. To demonstrate these principles we created *AudiOdyssey*, a PC rhythm game that is accessible to both sighted and non-sighted audiences. By following accessibility guidelines we incorporated a novel combination of features resulting in a similar play experience for both groups. Testing *AudiOdyssey* yielded useful insights into which interface elements work and which don't work for all users. Finally a case is made for considering accessibility when designing future versions of gaming user interfaces, and speculative scenarios are presented for what such interfaces might look like.

Thesis Supervisor: Eric Klopfer
Title: Associate Professor and Director of the Scheller Teacher Education Program

Contents

Acknowledgement

I'd like to thank several groups for their help making this thesis possible: the MIT EE/CS department for their support, the IGDA accessibility special interest group for always being helpful and answering my stupid questions, Eric Klopfer and the entire Education Arcade for providing excellent guidance and advice on my research, Team Reverb for their hard work and dedication to an insane project, and the Singapore-MIT GAMBIT Game Lab for sponsoring this work, providing me with the resources to make AudiOdyssey, and pushing me to new heights.

Most of all I would like to thank Rinat, Keren, Elana, and Ephraim for being such a wonderfully loving and supportive family, even though they never bought me a game system growing up since it would rot my brain.

Chapter 1
The Case for Gaming Accessibility

Accessibility refers to who can play a game, and is generally used to describe opening games up to the disabled. In contrast, usability refers to how well a user interface can be used by the target audience(s). While most developers agree that usability is important, many also feel that making accessible games is an altruistic mission, and that the benefits of accessibility do not outweigh the added costs [1]. *This is not the case!* Aside from humanitarian reasons here are many important justifications for making games accessible:

- Any added cost is certainly offset by an increased potential market, as an accessible game is one which impaired individuals can purchase. Furthermore, accessibility design themes tend to make games more usable for everyone, resulting in a game which will be easier to use for a broad section of the population.
- While making a game accessible may incur extra costs it is likely not as high as one might expect, especially if accessibility is considered from the outset. This thesis outlines several design principles which, if kept in mind from the beginning of development, can have big payoff for little investment.
- Any United States federally funded public computer work (like games) must have accessibility measures built in [2]. Other countries are adopting similar legislation as well. The result of these laws is that without thinking about accessibility it may be impossible to get government funding.
- Even those who are not disabled now might be someday. The sad fact is that impairments tend to be acquired as people grow older, and as the age of the gamer increases, so does the likelihood that he or she has accessibility concerns [3].

Of course, it is not possible to make every game user interface accessible to everyone, nor is it advisable to attempt to do so in all cases. Rather than implying that this is the case, the goal of this thesis is to impart two key ideas onto the reader:

Key Idea # 1) When developing a game one should think about which user groups could play an accessible version, and which interface changes could help achieve that end without changing the core game aesthetic or incurring huge added costs.

Key Idea # 2) Even if it is not clear how to make a game accessible, there are certain design principles which can be followed that tend to increase usability across the board. This increase in usability may in turn lead to accessibility.

This chapter stresses the importance of the user interface (UI) in the gaming experience. Like all software components, UIs have evolved over time. This evolution has coincided with an expansion of user demographics. However, ill-conceived UIs can be detrimental and serve as a barrier to playing, as games that are inaccessible to certain disabilities leave that portion of the population unable to play like everyone else. By observing

certain design rules (which are covered in depth in Chapter 2) it is possible to create highly usable UIs that work for a larger cross section of the population.

1.1 Video Game User Interfaces

With so many arguments for making accessible user interfaces an obvious question arises: if it's such a good idea, why aren't more people doing it? Perception is part of the problem: the cost is too high, it's too difficult to do correctly, it's not that important, etc. Perhaps a bigger problem is that UIs are frequently more a product of evolution than design. Oftentimes developers do not spend much time determining how a UI should work, instead choosing to copy previous efforts and using available toolkits and standards, even if such a solution may in fact be sub-optimal for some players. It is therefore worthwhile to explore a brief history of how the games industry has arrived at its current state.

Video games, or games in which an individual (*user*) interacts with an electronic device (*system*) through a controlling mechanism, are possibly the fastest growing communication medium in human history. In 1958, the world was introduced to the first graphical video game, *Tennis for Two* [4]. Forty eight years later, Nintendo releases *Wii Sports*, a set of sports games including tennis. What has changed in the interim? Certainly other tennis games have been released between 1958 and 2006, each one in some way improving, or attempting to improve, on previous incarnations. Over the years tennis games have seen the addition of 2D followed by 3D graphics, semi-realistic avatars, and ball physics. Perhaps most interesting, though, have been the changes to the game's UI. *Tennis for Two* was played on an oscilloscope with dials and a button, and allowed two people to play a simulated tennis match against each other using a simplified graphical representation of a court, net and ball. In contrast *Wii Sports* is controlled via a motion sensing device (the Wii Remote) and played on one's television set.

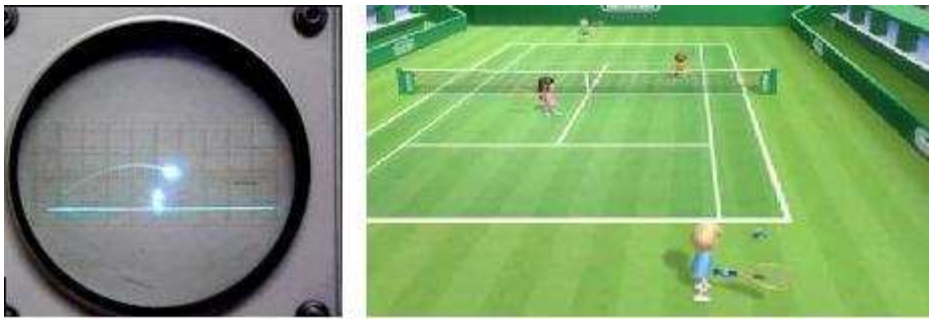

Figure 1 – *Tennis for Two*, the first graphical video game, used an oscilloscope and dials to control the game. 48 years later *Wii Sports* uses a television and a Wii Remote.

What happened in between these cases? In the late 1970's the world was faced with a proliferation of video games, first in the arcades, and then at home. The introduction of console systems from Magnavox, Atari, Nintendo, and Sega allowed millions of users to play video games at home on their televisions. A diverse set of controllers bearing joysticks, directional pads, and buttons were used to race cars, play sports, solve puzzles,

and beat levels. The 1990's saw further growth in this market spurred in part by the introduction of portable gaming systems[1]. Controller configuration changed as shoulder buttons were added and thumbsticks replaced directional pads, but the overall theme of "joysticks and buttons" stayed roughly the same.

Perhaps the driving force behind why joysticks and buttons were the standard for video game input for so long is their high level of robustness. The Swiss army knife of video game UIs, they fit myriad games with regularity. Joysticks usually correspond to spatial movement and/or directionality, and buttons correspond to actions the user can take. They tend to be fairly simple, limiting the user's actions to a few well understood options (though complexity can increase with the number of buttons and context-sensitive buttons). They are efficient, as the user can rapidly and repetitively enter game commands with the same muscle movements. Finally, such interfaces are familiar and comfortable after having been the standard for so long.

For all these reasons the joystick and button interface has been the de facto input interface for games for decades. This is a curious phenomenon, as just about every other aspect of gaming changed radically over this time period: processing power grew logarithmically according to Moore's law, and graphical representation of images improved from simple pixel artwork, to sprites, to dynamic polygonal models with cinema-quality cut scenes. Despite these dramatic changes the user interface remained mostly unchanged, at least until 2000's.

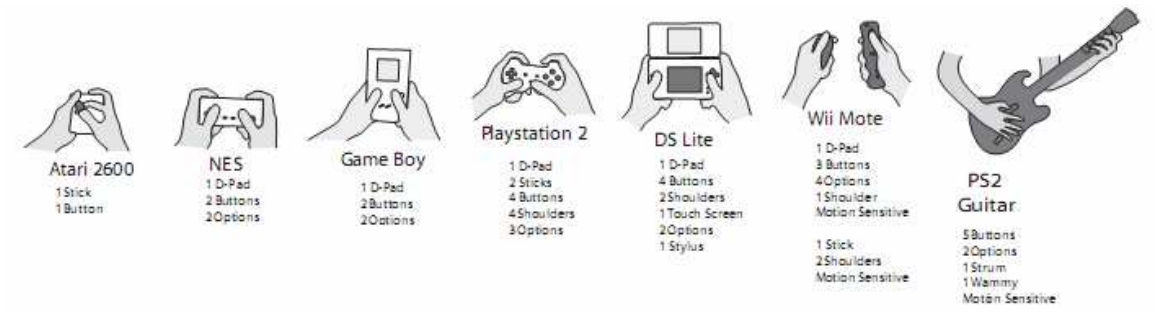

Figure 2 – Critical points along the evolution of game controllers (image by Damien Lopez [5])

This dominance can be contrasted to a new trend over the past few years, in which games have begun to embrace a new wave of user interfaces which focus on more natural forms of input[2]. Highly successful titles like *Dance Dance Revolution* (Arcade, 1998), *Guitar Hero* (Playstation 2, 2005), and *Rock Band* (Xbox 360 and Playstation 3, 2007) have shown consumer willingness to purchase external hardware for use with a single game series. System-specific generic user interfaces have been successful as well, as Nintendo has shown with the touch screen DS and the pointing/motion sensitive Wii Remote, and

[1] The Nintendo Game Boy, the first massively successful portable interchangeable game system, was released in 1989.

[2] While there were some earlier noteworthy efforts like Nintendo's Zapper and Super Scope, they generally did not catch on with game developers and had few compatible titles.

Playstation with the vision based EyeToy and SixAxis accelerometer controllers. This proliferation of alternative UIs for games does not herald the death of the joystick and button interface, rather it reflects the growth and diversity of the games market. Joysticks and buttons continue to exist because of their versatility and efficiency, however they are now being combined with other forms of input to give the user a more compelling, and immersive[3], experience.

As the number of new interfaces expands the range of what actions can control a game grows as well. Instead of moving a dial to cause a paddle to hit a ball, players can now swing a Wii Remote like a tennis racket. Internal vibration can give controllers a small amount of haptic (touch-based) feedback. Cameras can detect motion and display images of the user on screen, while microphones can allow voice chat, translate speech, or detect pitch while singing. Touch sensitive screens can read finger presses or strokes and writing from pens. Indeed, gamers today have more ways of playing than ever before (for predictions on future UIs, see Chapter 4).

Of course input and output are not the only aspects of user interfaces, as the graphical component (what is displayed on the screen) and game mechanics are critical as well. While this thesis will touch on these UI aspects the focus will center on input and output methods.

1. 2 New Demographics in Gaming

Just as game UIs have evolved and diversified over time, so have the genres. An industry once dominated by simple puzzle games, text adventures, and reflex challenges has grown to include platformers, sports simulation, first/third person shooters, open world sandboxes, and causal games. There has also been a proliferation of new game mechanics (the core game play elements of a title). This evolution has been spurred by improvements in processing power, graphics, and of course, UIs.

These new genres have drawn in new users, expanding the base of those of who play games and changing the notion of the typical gamer. What was traditionally dominated by young males has opened up to many other groups over the years. Many of the same kids who grew up playing *Pac-Man* (Arcade, 1980) are now adults playing *Halo* (Xbox, 2001). Titles like *The Sims* (PC, 2000) and *Bejewelled* (Internet, 2001) have drastically increased the number of women who play games, and offerings like *Wii Sports* and *Brain Age* (DS, 2005) are bringing the elderly into the fold. The widespread availability of computers, coupled with the ever-increasing number of free online games, has resulted in more users having access to video games than ever before.

[3] Immersive here means how much the user is drawn into the game's world and feels as though they are actually in the gaming environment.

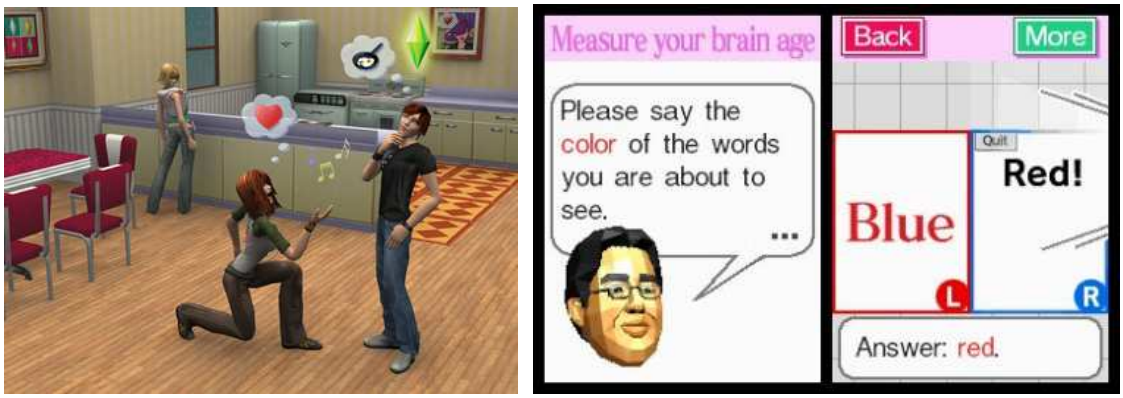

Figure 3 – New demographics to gaming – *The Sims* (left) is wildly popular among women, while *Brain Age* (right) is played by older gamers

Much of this growth has been dependant on new types of games: obviously, if the subject matter isn't interesting, the audience won't play it. So making new games brings new gamers into the market, but certainly this isn't the only factor. Another key element has been the development of new, meaningful, and intuitive UIs. Novel interfaces get attention (i.e. hey, how are you playing that?") and make experiences more immersive and exciting – consider the tremendous popularity of *Rock Band*. The combination of new types of games and new UIs has been a driving factor in the expansion of gaming demographics.

1.3 Barriers to Gaming

Despite the advances in user interfaces and the new gaming genres, not all people can play all games. This may seem unsurprising as no media are consumed by or intended for the entire population, but at least in most cases the consumer has the *choice* of using it. What barriers prevent certain groups from playing certain games, and why?

Certainly, the appeal of the genre has a large impact on who will play a game. Many games are explicitly designed for and marketed to certain demographics, for example *Brain Age* is a game conceived and intended for an older audience while the *Pokemon* series (first release Game Boy, 1996) sits on the opposite end of the spectrum for younger children. Here, the genre serves as a barrier to entry based on aesthetic. If a certain group doesn't find a type of game mechanic appealing, they won't play it.

While it may be obvious that low appeal acts as a barrier to gaming, it is perhaps less obvious that poorly conceived or difficult to use UIs can be just as much of a barrier. **Simply put, if one can't use the UI, one can't play the game!** If the controls are too difficult to learn or use or if they detract from the gaming experience users will decide not to play the game. This effect is sometimes seen when one is fighting the controls more than the game mechanic and gives up in frustration.

Disabled individuals may be the largest group of would-be gamers who have difficulties interacting with UIs. Most of the popular mainstream games have serious accessibility

issues that result in UIs that are unusable by certain segments of the population. For example, a deaf person will be at a huge disadvantage if they can't hear gunshots in a first person shooter, and a veteran with only one hand might be unable to play a game that requires the use of both hands for controls. While it may seem strange, this adds to the list of discriminations disabled people face in our society.

1.4 Impairments that Require Accessible UIs

Good user interface design is critical for all users, even more so disabled users as inaccessible interfaces can actively prevent use. Most people who need accessible UIs fall into one (or more) of four categories: visual, audio, motor, and mental impairments.

Visual impairments are sight conditions which cannot be fixed with any corrective lenses. The most extreme visual impairment is blindness, affecting 0.8% of the U.S. population[4]. An additional 2.7% suffers from low vision which can result from a number of conditions, such as extreme myopia, tunnel vision, cataracts, and retinal degeneration [6]. Sight aberrations are conditions which arise due to a defect or malfunctioning eye tissue, the most common of which is red-green color blindness, which affects roughly 8% of male population[5] [7].

Hearing impairments are conditions in which the individual has substandard hearing. This includes deafness, single ear hearing, partial hearing loss, and low/high frequency insensitivity (often associated with aging). While easier to accommodate than visual impairments as people tend to be sight oriented it is still important to consider hearing impairments as they affect 3.3% of the U.S. population[6] [8].

Motor impairments are conditions wherein individuals have limited function in muscle control or movement or a limitation in mobility. This is often a difficult set of disabilities to accommodate as the range of disabilities is so large. It includes missing digits/hands, degenerative diseases like arthritis, Cerebral Palsy, and Multiple Sclerosis, and debilitating spinal injuries such as para- and quadriplegics. This group also has the highest number of transient impairments, as many non-permanent injuries like spraining a shoulder or breaking a wrist results in being temporarily disabled.

Mental impairments are mental and psychological disorders such as mental retardation, organic brain syndrome, emotional or mental illness, and specific learning disabilities [9]. A complete analysis of the accessibility concerns raised by this group falls outside the scope of this thesis; however several of the usability and accessibility design principles contained herein are applicable.

This list of impairments is not comprehensive, as the range of disabilities is quite broad. It is quite possible to have multiple disabilities, as well as ones which don't affect

[4] Percentage refers to people 15 years and older.
[5] It affects significantly less women (~.5%) as the defect is linked to the Y chromosome.
[6] Percentage refers to people 18 years and older.

gaming. A loss of one's sense of smell or taste, for instance, doesn't affect one's ability to play any games (at least not any the author is aware of).

It is also worth noting that the age is strongly correlated to the likelihood of disability. In America, less than 6% of people under the age 15 are impaired, while 18% of 16 – 64 year olds suffer from disability, and over 41% of senior citizens over 65 have a disability [6]. It is therefore much more important to consider accessibility with older age groups, and to keep in mind that the average gamer is getting older each year.

1.5 Prior Accessible Gaming Efforts

Gaming accessibility is certainly not a new problem, and over the years several projects have addressed the issue. The two most common methods for making games accessible are either to take existing inaccessible games and remap new accessible UIs, or to create games from the ground up with accessibility in mind.

Remapping UIs involves overriding a game's input, output, or both. It is more common to override input with a specialized controller or control scheme which certain disabled groups can use. An example of this is a sip/puff controller, created for quadriplegics unable to move their body from the neck down. The user sips and puffs air out of straws, which maps directly to button pushes of a controller (in the case pictured below to the Xbox 360). Overriding a game's output is rarer, and is most commonly used by the visually impaired. Screen readers and optical character recognition systems scan and convert text to speech so that it can be heard.

Figure 4 – A quadcontroller allows quadriplegics to play games by remapping air puffs and sips in straws to buttons

The biggest benefit of remapping controls is that it opens up popular mainstream games, which is critical as many impaired people want to play the same titles as their non-impaired friends [10]. Unfortunately, it is not always possible to remap UIs in this sense. Text can be converted to speech, but images cannot. Games that require entering rapid sequences of button presses may be impossible to fire for certain motor impairments (at least, without controller assistance). Specialized inputs, like pointing with the Wii Remote, may not have simple remapping methods that work for certain disabilities.

Even if it is possible to remap controls it is not always advisable to do so. Frequently part of the fun of a game is the interface, and changing it without forethought is potentially detrimental. In the pervious example of *Wii Sports* tennis part of the fun is actually swinging the controller as if it were a racket. If this functionality were changed to pressing a button then much of the game's charm and fun would be lost.

As remapping can potentially kill the fun in game a better solution is to design and create accessible titles from the ground up. Many have been created, for example there are over 200 visually impaired accessible titles [11], and countless more for the hearing impaired. A shortcoming of creating games in this manner is that sometimes the UI design is directed *only* at a certain disabled group, and the resulting game actually becomes *inaccessible* to other groups. An example of such an effect can be seen in *Shades of Doom* (PC, 2001), a first person shooter that is playable by the visually impaired [12]. The game is audio only, as blind individuals don't need graphics. This lack of graphics actually makes the game inaccessible to sighted audiences, as most sighted users are unwilling to tolerate a game without graphics.

Design for only one group is problematic as it tends to lead to gaming segregation, a notion of separate games for separate groups**.** Such design invariably leads to accessible games that, by and large, are lower quality than their inaccessible "mainstream" counterparts. Disabled gamers themselves dislike this model, as again they want to play the same games as everyone else [10].

The other common pitfall for accessible games is that without serious forethought it is very possible that resulting game will have drastically different play experiences when played by different user groups. For example, in a standard first person shooter the challenge is to shoot others before getting shot. A visually impaired variant which changes all output to audio shifts the focus as movement requires sonar (or some variant thereof) and the user is forced to keep a mental map of the surrounding area. The game's central challenge changes from shooting others to spatial navigation and as such becomes less enjoyable for sighted players. The problem is even more irksome when sighted users realizes that a simple visual feature (like a map) would fix the issue.

There are a few noteworthy examples of games that manage to be accessible to multiple groups while avoiding these pitfalls. *Terraformers* (PC, 2002) is a game which works for both sighted and non-sighted users by having fully developed audio and visual outputs, allowing both groups to have similar play experiences[7] [13]. *UA (Universally Accessible) Chess* (PC, 2004) is a version of chess that has multiple input and output methods, allowing many different types of disabled users to play the game without changing the game experience [14].

[7] Terraformers was so good it won the Independent Games Festival award for innovation in audio in 2003.

Figure 5 – *UA Chess* allows many types of disabled gamers to play chess by offering several inputs and output schemes

1.6 Summary

Games have evolved tremendously over the past few decades, as advancements in technology have led to amazingly realistic and engaging offerings, while shifts in player demographics indicate the widespread popularity of video games. Despite these changes many different disabled groups are still unable to play most titles due to inaccessible UIs. Developers are frequently hesitant to attempt to fix the issue due to a perception that such endeavors are difficult, costly, and/or fruitless. However, with some careful thought and planning it is possible to create UIs that make sense for the game and are usable by a wide segment of the population.

As a closing example, consider *Brain Age*. Five years ago who would have believed that older individuals would be willing to yell colors out loud at a portable gaming device[8]? Yet after selling millions of copies the game has met broad market acceptance, and the thought of talking to a game no longer seems so strange. Accessibility in games is a similar challenge – the argument that it can't be done due to a lack of prior success is weak at best. **We can make new, accessible UIs, and we can make them interesting**.

The next chapter will build on this notion of creating accessible UIs, and will illustrate examples of games that do and don't do a good job of achieving this goal. Following this is a case study of the development of *AudiOdyssey*, a music/rhythm game created with accessibility principles in mind.

[8] This is actually part of Brain Age's game play!

Chapter 2
Designing Game Accessibility and Usability

Chapter 1 states that poorly designed user interfaces can prevent people from playing a game, and that great UIs can make games more fun and accessible to many users. But how are great UIs created? At first glance it might appear that the design is something of an art form; developers draw on their years of experience and their gut instincts to create a UI which is both highly functional and aesthetically pleasing. While this is true to some degree one can do even better by learning from both positive and negative examples of existing game UIs and by analyzing their usability.

The goal of this chapter is to present guidelines for creating accessible user interfaces in keeping with the two key ideas. To do this, usability analysis principles are presented and then related to accessibility. Following this is a discussion of some common accessibility design themes with a focus on tradeoffs and potential pitfalls. The chapter concludes with suggestions for game genres that are ripe for accessible titles. Chapter 3 builds on these guidelines by discussing the development process of *AudiOdyssey,* an accessible game created by the author.

Key Ideas to Remember:
 # 1) When developing a game one should think about which user groups could play an accessible version, and which interface changes could help achieve that end without changing the core game aesthetic or incurring huge added costs.

 # 2) Even if it is not clear how to make a game accessible, there are certain design principles which can be followed that tend to increase usability across the board. This increase in usability may in turn lead to accessibility.

2.1 Analyzing Usability

To analyze usability in a video game's UI it is helpful to first define a common terminology. This paper uses four dimensions of usability: *Learnability, Simplicity, Efficiency, and Aesthetic*[9] [15]. The purpose of defining such usability dimensions is to provide a methodology for talking about UIs, i.e. user interface *X* is more learnable, but less efficient, than user interface *Y*. Keep in mind that these dimensions are not a rating of how "good" a UI is, as sometimes it makes sense to disregard certain principles for the sake of gameplay. What's important is in these cases the game developers are conscious of this choice, and it is not an arbitrary decision.

[9] This list is based on Miller's dimensions of HCI usability, but only focuses on the four that are especially relevant to video games. For more information on usability dimensions the author recommends references from design experts like Neilson [16], Norman[17].

Learnability *– How easy is it to learn how to use the system?*

This dimension mainly concerns novice users. Most developers prefer straightforward UI learning and to this end tutorials are commonplace, as they provide a safe environment for learning controls and mechanics while providing meaningful assistance and guidance. Once learned using the UI should be based on recognition rather than recall. Generally one should not be expected to remember every game control, rather there should be easily accessible reminders where appropriate. The exception to this rule comes from games where learnability is part of the game mechanic. The fighting genre is a classic example of this exception, as players are rewarded for learning difficult to remember controls.

Figure 6 - *Street Fighter 2* (Arcade, 1991) has learnability at the core of its game mechanic as players must learn complicated input sequences for attacks

Even within fighting games there is a hidden learnability method, namely feedback. Obvious, immediate, and repeatable feedback between input and output are critical for learnability, and thus trial and error in a fighting game will likely teach the user much about the mechanic. Relying only on feedback is dangerous, though - how will users learn non-obvious or complicated input methods? Perhaps more importantly, will such a game be fun for someone who has yet to learn these expert inputs?

Learnability also dictates that controller mappings should be consistent; if the **X** button is used for "yes" in one area of a game, it should always be used for confirmation throughout. Likewise, controls should match real world expectations; if most other games in a genre have **X** as the standard button for yes then a new game within that genre should conform for maximum learnability[10]. Games with highly learnable interfaces are inviting to new users as they tend to be easy to pick up and play, while novel user interfaces frequently have difficult learnability issues as all users are effectively novices.

Efficiency *– How quickly can tasks be accomplished?*

In contrast to learnability, efficiency mainly concerns expert users who want to accomplish tasks quickly. The interface should allow users to move swiftly and

[10] However one must be wary of slavish adherence in the face of new, superior control schemes.

effortlessly through the game. Efficient controls become an extension of the user's body, with the user being able to rapidly react to the game without even thinking of how to command the system. There are often tradeoffs between efficiency and learnability and simplicity which game developers must be conscious of. When making such tradeoffs the rule of thumb is to think about the player demographics. If the game will cater more towards expert users, it may be worthwhile to sacrifice some simplicity and learnability in favor of more efficient controls. For example, *Starcraft* (PC, 1998) has a relatively efficient UI that allows users to quickly accomplish many actions by making use of the many buttons on the keyboard. At the same time the game controls have serious learnability issues, and this is reflected by the fact that a large segment of the single player campaign is effectively an extended tutorial.

Figure 7 – *Starcraft* has efficient controls that allow the user to quickly perform a wide variety of actions

Simplicity *– Are the controls as simple as possible?*

Game controls should be as simple as possible, but no simpler. This seems straightforward on the surface, as humans only have so many fingers which they can only move at a certain speed. However, this frequently results in serious tradeoffs with efficiency. The best UIs are ones that determine ways to keep the game mechanic efficient and engaging but still only use simple controls. A great example of simple controls is EA's "Family Play" mode for the Wii version of *Madden NFL* (Wii, 2008) football. The basic idea in Family Play is to trade off some functionality in exchange for a simpler control scheme which allows the user to focus on more critical game elements. Due to obvious tradeoffs in player freedom a less simple "Advanced" mode is available for expert users.

Figure 8 – The newer version of *Madden NFL* has two control schemes, a simple novice-friendly version (left) and an advanced expert-friendly version (right)

Aesthetic *– How pleasant is the user interface experience?*

Arguably the most important dimension for video games, a user interface should have a pleasing aesthetic that makes it enjoyable to use. Game actions should be clear, obvious, and easy to select. Graphics and sound effects should be agreeable, consistent, and informative. The style of the UI should match the tone of the game. This is the dimension of usability that game developers tend to focus most heavily upon, and for good reason! A game's UI can define the user's experience.

Figure 9 – The Wii Menu system has a calming, subdued aesthetic that makes it enjoyable to use

Of course, it is generally not possible to excel in all categories without seriously changing the UI's impact. Larger games tend to have difficulty adhering to optimal design due to their inherit complexity, and frequently one dimension will be sacrificed to lead to increases in other dimensions, or guidelines will be ignored for the sake of supporting a game mechanic. These sorts of decisions are acceptable as long as they are made consciously by the developers. However, poor decisions will likely make the interface much more challenging for several demographics.

2.2 Usability Tradeoffs

Usability analysis is well and good, and provides a good starting point for talking about UIs, but again it is rarely possible to follow all principles. Improving one dimension may diminish another, thus it is always important to keep tradeoffs in mind. Should a game have a more efficient user interface that is harder to learn, or one that is less efficient but more intuitive? Such questions frequently arise during development, and reaching meaningful solutions generally relies on thinking about how the user will be most likely to interact with the system.

Likewise adhering to all of these guidelines may not be realistic in terms of cost, time, and manpower. As always it is necessary to weigh the costs of implementing features against the benefits gained from the implementation. On the other hand increases in usability can potentially reap large rewards in terms of customer base, hence improvements to any dimension is highly advised if possible.

Finally, it is perfectly reasonable to deliberately go against these design principles if there is a good reason to do so. Game mechanics inherently have built in inefficiency and learnability challenges as the fun lies in overcoming these obstacles. It is fine for UIs to follow suit if that is part of the gaming experience. For example, it may be desirable to take control away from the user to force watching a critical cut scene (however, that raises the question of whether the cut scene is the only way to relay the critical information.) Obstruction mechanics seriously decrease learnability and efficiency, but if is done in the name of aesthetic such a tradeoff may be worthwhile. *Space Giraffe* (Xbox 360, 2007) is a great example of this. As a psychedelic space shooter the game relies on obfuscation to confuse the user and make the play experience exciting. Shifting backgrounds, constantly changing colors, morphing camera lens views, and no form of help or assistance make the game extremely challenging, but therein lies the fun. The game sacrifices several usability dimensions for the sake of aesthetic. Good interface designers know when to break best practices in the name of the overall experience.

Figure 10 – *Space Giraffe* deliberately makes serious tradeoffs in usability to support the game's aesthetic and mechanic

2.3 Relating Usability and Accessibility

Whether or not they realize it, many game developers treat usability and accessibility as separate and distinct topics. Conventional wisdom dictates that usability deals with how to make a UI straightforward for the population in general, while accessibility deals with making a UI accessible to one or more groups of disabled users. This thought process leads to a focus on making games usable (since that covers a large part of the market), but accessibility is often an afterthought or not even considered during the design process (since that covers a small part of the existing market.)

This mentality is flawed, as **accessibility is an extreme form of usability, and making a game accessible to one group usually makes it usable to many!** In fact, accessibility can be viewed as an extension of usability – UIs with highly usable dimensions tend to become accessible, and accessible UIs tend to be highly usable by many different user groups.

A good way to think about accessibility is to consider real world examples, like the wheelchair ramp which grants handicapped individuals access to structures with stairs or sidewalks with curbs. While designed with handicapped accessibility in mind, the ramps actually work for a much greater number of people – the mother with a stroller, the elderly individual with a cane, the teenager on rollerblades, and anyone else who just doesn't want to use stairs for some reason. Another great example of a real life feature which is highly usable by everyone is handicapped accessible bathrooms.

Accessibility is not always the answer though, and does not work for all groups in every situation. A real life counterexample can be seen in motorized ramps in buses. While these ramps do make buses accessible to those in wheelchairs, they do not help the general public. Furthermore they cost a good amount of money and take up space that could be used for seating. This isn't intended to diminish the value of such ramps, just to illustrate how accessibility does not always translate into usability.

Just like in the real world, making a game accessible to one group often results in usability gains for other groups. While closed captioning may be most effective for the hearing impaired it adds extra usability for other groups. Older gamers might find it easier to follow the conversation with subtitles, expert gamers could use the feature jump through dialog quickly by reading instead of listening, and foreign gamers can use localized, translated closed captioning. This is a case where a small amount of extra effort on creating accessible user interfaces can result in large payoffs for the many users.

Usability principles are also important when addressing *situational disabilities. Situational disabilities* refer to situations in which a non-disabled individual is unable to use an UI normally due to a disruptive environment, effectively rendering them temporarily disabled[11] [18]. For example, playing a handheld gaming device in a noisy cafeteria without headphones results in being situationally deaf since the user is unable to hear any audio output from the system. Playing in public may introduce social constraints

[11] Situational disabilities are also sometimes referred to as functional disabilities.

rendering a person situationally mute, and drinking alcohol while playing a game can result in being situationally mentally and motor impaired.

Making user interfaces accessible to the disabled also makes them accessible to the situationally impaired. So in the previous example if a portable game system has closed captioning it not only becomes accessible to the hearing impaired but also to the student playing in the noisy cafeteria. By removing reliance on audio and adding redundant output channels, the portable game becomes accessible to all users regardless of ability to hear.

2.4 Central Design Themes for all Groups

The wide range of impairments makes compiling a list of universal design principles difficult, if not impossible. What is usable for some may be unusable for others, even within the same disability group! Consider visual impairments – some people have trouble viewing high contrast elements, while others are unable to view low contrast details. Despite these difficulties, it is still possible to follow some general rules of thumb which tend to lead to highly usable and accessible interfaces.

Probably the easiest rule to remember is the importance of simplicity. Keeping the output simple is helpful as it reduces confusion and makes it easier to pick out critical information. For many impaired individuals the interface bottleneck lies in discovering what the system is saying – legally blind people tend to slowly scan the screen for information, the completely blind use screen readers to read text, and mentally impaired individuals might need longer to parse given options. A simplified output helps reduce the time spent in this phase.

On the input side simplicity is still important, but even better are configurable or alternate control schemes. Configurable control schemes are especially important for motor impaired individuals as frequently they are unable to use all of the elements of an interface controller. Some motor impaired individuals have specialized controllers (like the sip-puff controller in figure 4), and having configurable controls makes remapping easy. Impaired individuals are also generally willing to spend more time configuring controls. Many computer games offer such functionality, but consoles titles seldom do.

Alternate controls tend to make the largest difference when the control schemes are highly varied. For instance, inverting camera tilt in a first person shooter has only marginal accessibility payoffs, while *Earthbound's* (Super Nintendo, 1994) alternative control schemes are much better as one can either play with two hands or just the left hand.

Figure 11 – *Earthbound* has excellent alternative interface control schemes, and can be controlled with two hands or just the left hand

Even better for some people than configurable controls are partial artificial intelligence (AI) controls. While rare now, there are several such schemes on the horizon, perhaps most notably EA's family play controls (figure 8). Passing sections of control over to the AI not only helps impaired users but also novices who haven't had a chance to learn how to play the game. Another great example is *Gordon's Trigger Finger* (PC, 2007), a *Half Life 2* modification that has AI auto-movement and aiming, while the user just worries about shooting. The result is a first person shooter which motor impaired users can play with only one switch [19].

All modern games have two main forms of output, audio and video[12]. Two of the broad disabled groups are visual and hearing impairments. Therefore any system that outputs information in only one format will always be inaccessible to one of these groups. Redundant audio for all visual effects, and vice versa, is the ideal way to overcome this problem. Closed captioning for all audio can make most games accessible to the hard of hearing, while sound effects and speech output can make a large number of games usable by the blind. An added benefit of redundant audio and visual output is that the game feels more natural to all users, as humans are used to hearing a noise when an action takes place; think how odd it sounds watching fireworks to see the explosions, but only hear them a split second later.

There are several common game elements and mechanics that tend to hurt accessibility. Mandatory timers that cannot be disabled greatly reduce usability as they require the user to quickly uptake and process information, and punish those who cannot do so rapidly. Complicated controls with large numbers of commands are highly problematic, but can be mitigated through menu browsing as the user won't need to mentally recall all the options and fewer buttons are required for action selection.

Two disabilities that affect large portions of the population and are relatively easy to accommodate are hearing impairments and colorblindness. Closed captioning of both

[12] Some games also have haptic feedback, usually in terms of a vibrating controller, however such schemes are rarely rich enough to provide much meaningful information.

speech and sound effects removes reliance on audio, and has many benefits for groups beyond the hard of hearing. As for colorblindness, games should avoid relying on color alone to convey information, and instead should also use secondary cues such as position, shape, and texture for differentiation purposes. Red and green with the same saturation should especially be avoided, as these colors are generally the hardest to tell apart.

User centric design and development, or having people who are actually in the targeted user group involved in the development process, is critically important and cannot be overstressed. When designing accessible UIs it is crucial for the developers to remember that they are not the users, and to actually get impaired users involved from the beginning. Design advice from these users is generally valuable, and can save time and money by pointing out accessibility issues before they are even implemented. Once the UI actually exists, it is just as important to conduct broad testing across all potential user groups who might want to play the game. Testing always brings the worst usability bugs to light, and once identified the developers can make appropriate decisions about the value of implementing changes.

While these general rules are useful for developing highly usable games which tend be accessible to many groups, they certainly do not cover every nuance of game design for all user groups. For more on universally accessible game design the author highly recommends reading Unified Design of Universally Accessible Games or playing *Game Over!* (PC, 2006) which teaches several of those design points [20, 21].

2.5 Potential Pitfalls

Bear in mind that accessibility is not a silver bullet, and cannot be used in every situation. It is not possible to make every type of game accessible, as some games and genres just don't lend themselves to certain disabilities. For instance, a rhythm title like *Guitar Hero* which focuses on music will not work for the hearing impaired, and it is probably not possible to make an accessible version. This doesn't mean that *Guitar Hero* can't be accessible at all, as it may well be possible to make versions that work well for the colorblind or motor impaired.

Here are some of the main pitfalls of accessible game UI design, and some partial solutions which may help developers think about how to counter these issues.

Complexity or special controls may preclude accessibility

Generally speaking, the more complicated a game, the greater the range of actions a user can perform. This range of actions is frequently a big part of the fun, so any UI or control scheme that limits these actions is likely to limit the amount of fun for the player as well. Furthermore control schemes are frequently meaningfully mapped, and changing them might ruin the experience – again, imagine *Wii Sports* tennis with a different input.

Partial Solution: If it would really hurt the game to make the controls simpler sometimes a good solution is to make two modes, one for expert users and one for novices. The

novice mode might sacrifice efficiency (and possibly some fun) in exchange for increased learnability, simplicity, and accessibility. Even this is tricky though, as adding modes and removing functionality can really alter the experience. For instance, in *Gordon's Trigger Finger* all functionality is removed except for shooting. While this is perhaps less fun than playing normally, it works well as it allows novices and the motor impaired to play against experts. However what if the game instead had automatic shooting, and let the user handle navigation? This actually would have been easier to implement, but it probably would have actually been much less fun, as there is much enjoyment in "pulling the trigger" in first person shooters. Thus game developers who make different control modes must be very conscious of tradeoffs.

Additional time and cost to develop

Sometimes accessibility costs outweigh the benefits. The unfortunate reality is that most games are made on a tight schedule with limited funds, and as a result the final version almost never has all the features the developers wanted to add. The added time and cost of creating an ideal UI may not be worthwhile in terms of increased revenue.

Partial Solution: Thinking about usability and accessibility from the beginning of development (rather than considering at the end) will drastically reduce implementation costs. Common toolkits may also be available for certain usability features, especially for computer (non-console) games. Finally it is important not to underestimate the added value of an especially slick and usable UI. Customers tend to be drawn to games that are easy to pick up and play, and sometimes usability features end up saving money in other areas, like closed captioning helping with localization (see chapter 2.6).

Accessibility for one group may cut out others

Accessible doesn't always mean more usable for everyone, and may not have the broad benefit desired. As with all software development, ill-conceived features that aren't carefully thought out have the potential to actually make the game worse. Consider an example where a multiplayer game requiring fast-paced reactions creates an accessible version. Impaired users may lose the game if it takes them longer to parse the game's output, or might cause their team to lose if the game is cooperative.

Partial Solution: Again, a potential solution is to have multiple input and output schemes which work for different groups (consider *UA Chess*). Alternatively it may be possible to make changes the game mechanic to accommodate different groups. In the example above handicapping or partial AI control may fix the multiplayer issues.

Of course, it is important to keep in mind that most games can't be accessible for everyone. However, by thinking carefully about accessibility issues from the beginning it is more often than not possible to find a full or partial solution that increases usability, doesn't cost a huge amount, and opens the game up for additional groups of people.

2.6 Specific Game Examples

While design guidelines are helpful perhaps the most illustrative examples are titles that follow accessibility principles. A great first example is *Half Life 2* (PC, 2004), a sci-fi first person shooter. The original *Half Life* (PC, 1998) had segments which relied on audio and did not have any redundant visuals to help hearing impaired users. Reacting to complaints from the deaf community, the sequel features full closed captioning not just for speech but also for sound effects.

Figure 12 – *Half Life 2's* closed captioning system makes the game more usable, and accessible to the audio impaired

Closed captioning is a great usability feature for several reasons. It makes it easier for a user to understand dialog (regardless of accents, voice volume, etc.) and it decreases the likelihood of error due to misconstrued character speech. Even better, it is useful for localization (porting the game to different languages for different countries) as new language files can be cheaply and quickly swapped rather than recording dialog in a new language.

Another game made with accessibility in mind is *Strange Attractors* (PC, 2005), a bizarre space navigation game in which the user controls a small spaceship trying to pass through a set of circular goal posts [22]. The twist is that the spaceship cannot be flown, and instead the user can only toggle gravity's effect on the craft. Thus navigation is performed by turning gravity on and off at the right moments, causing the ship to bounce off interstellar bodies. *Strange Attractors* is noteworthy in that the entire game is controlled with only one button, and as a result is highly accessible to the motor impaired. Furthermore, as a PC game the input can easily be remapped to a number of different controllers.

Figure 13 – *Strange Attractors* is highly accessible and usable thanks to its super simple one button interface

A third example of thoughtful UI design is *Peggle* (PC and Mac, 2007), an extremely popular casual game. Users clear multicolored blocks by launching pinballs around the screen. The game has a "colorblind mode" which adds symbols to the blocks making them easier to distinguish. The mechanic is fairly accessible to low vision individuals as there is no timing element, and once fired the user is not required to quickly respond to any action as the pinball bounces around.

Figure 14 – *Peggle* is highly accessible to low vision players thanks to different color modes and a game mechanic which doesn't require instant action

Beyond these examples there are still many titles and genres that are ripe for accessible designs. Casual games are often PC based and rely on simple and highly usable control schemes. Several are already accessible to the motor impaired, and it would not be a stretch for others to go the extra step. Text adventures (and their modern day counterparts like *Phoenix Wright* (first release Game Boy Advance, 2001)) can often be opened up to the visually impaired by adding audio files for all text. Such a change would actually make the game more fun for all users. Portable games for cell phones and handheld consoles are another prime target, as they are often played by users with situational disabilities. Redundant audio and visual output can help assure that the majority of people will be able to play these games regardless of the environment.

2.7 Summary

Accessibility and usability are linked in that highly usable interfaces tend to be accessible, and vice versa. By following some central design themes and thinking about usability from the start of development it is possible to make UIs that work for a broader range of people. However, not all accessibility solutions work for all games and one of the biggest challenges is determining which principles are correct for each game.

Recap of design themes:

- Simplicity
- Alternate and configurable control schemes
- Redundant audio/visual output
- Partial AI control where possible
- Browse and select for actions
- No mandatory timers
- Closed Captioning
- Never rely on color alone, especially red/green
- User centric design
- Broad user testing
- **Think about usability and the UI from the beginning**

Chapter 3
AudiOdyssey: A Case Study

AudiOdyssey is an accessible rhythm video game created in the summer of 2007 in an attempt to illustrate the design themes and concerns laid thus far. It is intended to be played by the general gaming audience, yet is accessible to non-sighted users thanks to a highly usable interface with meaningful audio output. The challenges encountered during development underscored the importance of considering usability concerns and planning a game's UI from the outset.

The creation of the game itself was a collaborative effort involving many people. The author and Prof. Lonce Wyse of the National University of Singapore created the game along with a team of seven undergraduates from MIT and various Singaporean universities: Dominic Chai, Bruce Chia, Paviter Singh, Mark Sullivan, Edwin Toh, Jim Wilberger, and Yeo Jingying. The game was supported by the Singapore-MIT GAMBIT Game Lab, and several members of the lab contributed to development as well. The underlying accessibility research behind the game is the author's.

3.1 Motivation

AudiOdyssey was initially conceived to address the issues discussed in the previous chapters, especially segregation in gaming due to inaccessible UIs. Despite the advent of new gaming mechanics, intuitive user interfaces, and increasingly sophisticated technology disabled groups were generally still not able to use or play most games. The few accessible titles that did exist were generally separate from mainstream offerings, and a small but vocal group of disabled gamers were clamoring for developers to create control schemes that would let them play the same games as everyone else [10]. Furthermore, many of the accessible games that were being released did not make the best use of their genre's affordances and as a result were frequently inaccessible to non-impaired gamers due to their lack of features.

These problems led to the decision to make a game that would be playable by both sighted and non-sighted audiences. By analyzing discrepancies in past titles, a short list of four core research goals was chosen for the project:

- Implementing a game design that allows visually impaired and sighted users to play the game in the same way, with the same level of challenge, and share a common gaming experience.
- Designing an alternative control scheme with improved accessibility for the visually impaired specifically using the Nintendo Wii Remote.
- Creating a fun, engaging game that relies on audio more than visuals to produce an exciting experience.
- Designing online multiplayer that allows for identity masking, at least in the sense that users in remote locations should not be aware of the visual status of their

gaming counterpart. (Unfortunately, we did not succeed in reaching this fourth goal)

Of course, any game with these features would have to be highly usable. Learnability and simplicity would be paramount, as the both the Wii Remote was relatively unknown to the visually impaired community. The game's UI aesthetic would have to center around audio output to ensure both groups could share a common experience, but lesser visual feedback would be necessary for sighted players.

3.2 Early Concept Prototype

As creating a highly usable UI was the key to achieving the research goals an early prototype example was made to learn more about designing games for multiple audiences. Designed to be a "throw away" prototype that would test certain ideas but not actually be used in the actual game's development, the predecessor was a small, computer controlled "wrapper" that would allow one to fight a *Final Fantasy X*[13] (Playstation 2, 2001) battle with audio only. The battle component was chosen due to its affordances: The lack of a timer allowed players to take as long as desired to make decisions, a limited, repetitive command menu with option browsing and selection meant users could become experts relatively quickly, and frequent, varied, and informative sound feedback notified players of in-game events.

The wrapper itself was extremely limited, but it was enough to test out certain UI mechanics. Directional audio cues indicated the user's options on a four button radial menu (see figure 15). These cues consisted of chimes coming from stereo speakers (left for left arrow, right for right arrow, both for front, and both at 50% volume for back), followed by verbal cues with the same directionality. A special button verbally reminded the users where they were in the menu at any moment, and a second special button returned the user to the top level of the menu at any time. This keyboard served as the input for the game, while output through the game was relayed through audio sound effects and the manual vocalization of all visual-only game information (i.e. "Enemy two hits Wakka for 300 damage. Wakka now has 400 hit points remaining.") While the wrapper was not descriptive enough to play the entire game, it made the battle section playable by non-sighted individuals.

[13] Final Fantasy X is a popular role playing game for the Playstation 2 in which the user controls a small band of warriors attempting to rid the world of an evil magical creature.

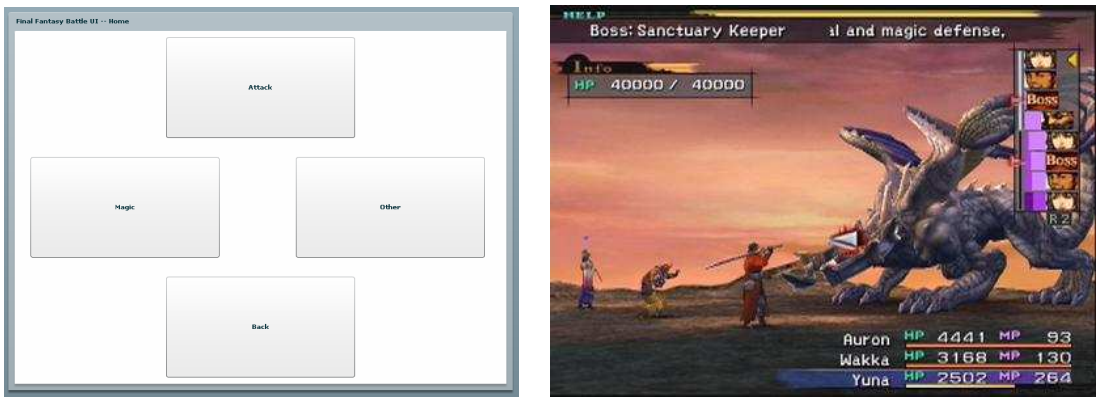

Figure 15 – An early prototype wrapper (left) allows blind players to fight battles in *Final Fantasy X* (right).The output is almost entirely audio – players listen for spatial sound cues to indicate menu options, then input commands with arrow keys in the appropriate direction

Formal testing of the wrapper interface was conducted with two female and three male visually impaired individuals, aged 17 – 53. The feedback yielded was valuable. Testers listened to the speech more than the chimes, they wanted more audio feedback, they thought the menu selection was too slow after using it a few times, and they liked the spatial sound output. Testing with sighted players yielded similar feedback and allowed for usability comparison between the two groups. Despite these concerns users were unanimously excited about the prototype. Overall reactions were very positive, with one user exclaiming *"Oh cool, this is just like when my friends and I [fantasy] role play!"* Another, older tester went through the whole battle session with a look of concentration on his face, and upon hearing the winning music was grinning ear to ear. *"When can I play the whole game?"* he asked.

Tester feedback and usability analysis of the playing sessions led to strong conclusions about how well the various components meshed together, and what tweaks and changes could be made to improve overall usability. The chime sounds were nice, but not informative enough to keep since most users waited for verbal confirmation anyway. Spatial sound output and verbal recordings (as opposed to text to speech synthesis) were both a hit, and would stay in the UI. While the radial menu system received more positive feedback than negative, it became apparent that a better UI would just have a simpler menu with fewer options. With this information in hand work began on a new game, which would eventually be named AudiOdyssey.

3.3 AudiOdyssey: Development Themes

The bulk of AudiOdyssey's development took place over two months during the summer of 2007. This section reviews some of the more important themes during this time period; for a complete week by week breakdown of the development process please refer to the appendix.

Picking a Suitable Game Genre

While analysis of the *Final Fantasy X* wrapper prototype gave a solid grounding for the game's UI, a suitable genre and concept was still needed. To guard against gaming segregation (separate games for separate groups) it was imperative to focus on audio's strengths and not attempt to substitute audio in the place of what should be video, as that would turn off sighted users. The music/rhythm genre was selected as both sighted and non-sighted audiences have similar expectations and knowledge of music, and it would allow the team to focus on making an engrossing audio experience with minimalist graphics. A DJ-based game seemed ideal as DJs can play a variety of music, have expressive freedom, and most importantly spinning records is pretty damn fun. The expressive freedom in particular allowed the team to experiment with new sound generation software created by Wyse [23].

Iterative design cycle

AudiOdyssey's development consisted of four two-week rapid build cycles. Each of these cycles started with deciding on a goal for the end of the period, specifying designs, creating and combining assets into a new build, and then testing that build with users. Feedback from testing was then incorporated into the goals for the next cycle. This iterative process resulted in catching bugs and usability issues early on. It also helped to keep the team on track pursing fun game mechanics and throwing out faulty components.

Constant Accessibility Consideration

Choosing to make an accessible game from day one rather than trying to tack it on at the end of the project allowed accessibility themes to permeate all aspects of game development. The design team made sure the central game mechanic would have the same challenge level for all users, the art team created graphics that complemented the audio without being overly distracting, and the sound team worked on conveying useful information through sound effects. Along with iterative development and testing, this helped ensure the final product would meet all of the team's accessibility goals.

User centric development and testing

The entire team was sighted so we sought out the assistance of a visually impaired consultant to help us make AudiOdyssey. Alicia "Kestrell" Verlager helped by offering feedback on design ideas and testing intermittent builds. Her comments were extremely valuable, identifying pitfalls early on and leading to the development of useful help features like the "how to use the Wii Remote" tutorial.

3.4 AudiOdyssey: Gameplay and Features

AudiOdyssey was completed at the end of summer 2007. A fully functional accessible game, AudiOdyssey has dual input control schemes via Nintendo Wii Remote or keyboard. The user stars as Vinyl Scorcher, an up and coming DJ trying to get people in a

nightclub dancing. This goal is accomplished by either swinging a Wii Remote or pressing keyboard keys in time with a set of repeating audio-visual cues. Each time a set is matched, a different instrumental track is added to the song. As more tracks are added, the song gets more complex and audience members start dancing. Once all the tracks are added a short twenty second bonus section ensues in which random sound effects can be generated by waving around the Wii Remote or pressing directional arrow keys. At the end of this bonus section one of the audience members (the "Scrotch") accidentally bumps Vinyl's table, causing some of the previously matched tracks to drop out and forcing the user to beat match again. This cycle of beat matching and bonus sections continues until the level ends, at the song's finish.

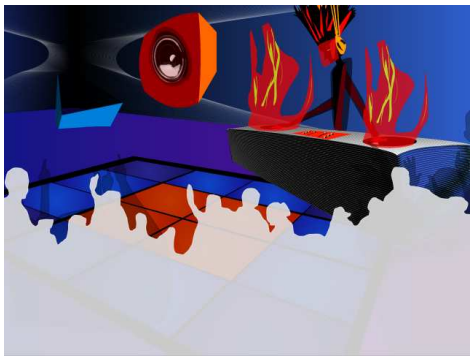

Figure 16 - Vinyl adds a new track to his song, and more dancers take to the floor

Chief among AudiOdyssey's features is its two control schemes. Wii Remote play allows users to swing the Wii Remote in time with the beat, and relies only on the controller's accelerometers (since pointing is inaccessible for blind gamers). Opening up this device to the visually impaired was one of the project's main accomplishments. Keyboard controls are available as the majority of users do not have access to a Wii Remote and are much more familiar with keyboard input.

Figure 17 – *AudiOdyssey's* main menu, with high contrast text and option vocalization making selection easy for low vision and non-sighted players

Since the Wii Remote is a new device for most users, AudiOdyssey focused heavily on UI learnability. A section called "How to use the Wii Remote" explains how the device works and allows visually impaired users to explore functionality at their own pace. The

game has two complete levels; the first is a tutorial which has no timing element and a gentle learning curve, while the second is an advanced level with a more complicated song and steeper curve. As of spring 2008, the game is available for free download at http://gambit.mit.edu/loadgame/audiodyssey.php

Figure 18 – Victory screen at the end of the advanced level

3.5 Testing and Reception

The final version of AudiOdyssey was tested with two small groups, one with four non-sighted people aged 27 – 49, and one with four sighted people aged 19 – 28. Testers were asked to fill out short questionnaires before and after playing the game, and were monitored during game play to track how long they spent in each section of the game. Testing started with the Wii Remote UI, and users were free to switch to the keyboard UI at any time.

Of the non-sighted group only one played video games on a regular basis, and none had ever used a Wii Remote before. All of the testers in the sighted group played video games at least on a sporadic basis with casual games, and three had played Wii Remote games before. Everyone who had not used the Wii Remote beforehand selected the "How to use the Wii Remote" tutorial before attempting the main game. Despite this tutorial, all of the blind users found using the Wii Remote difficult, with one tester commenting "*The Wii Remote was hard to understand and move around, and hard to understand what the movements did*", indicating serious feedback and learnability issues. However, we do have reason to believe the Wii Remote controls are fun as three of the four sighted testers were able to play the game with this scheme, and one sighted tester never switched to the keyboard controls because "*they looked boring compared to the Wiimote"*.

Upon switching to keyboard inputs all users performed better, especially in the non-sighted test group. Most testers played multiple rounds of the game, either repeating the tutorial level or challenging the advanced level. One blind tester who was unable to progress in the tutorial complained that the game was too frustrating, and gave up after a short time. It is interesting to note that people with musical experience tended to perform better than those without.

The testers seemed to enjoy the game experience overall. Testers offered lots of positive feedback, such as "*Cool, it's kind of like Guitar Hero*"[14], and suggested future features for the game, like "*It would be great if it featured well known songs and beats*" and "*I'd like to see a more competitive [multiplayer] game*". The sighted testers were surprised to learn that AudiOdyssey was accessible to the visually impaired, and one person tried playing again with his eyes closed upon learning this. General consensus indicated that the game was fun and that with some more work and perhaps a multiplayer element the game would be very engaging.

3.6 Development Post Mortem and Reflection

Looking back at AudiOdyssey creation it is impressive how well development went. Team members were not only talented but worked well together, had excellent communication, and were thoroughly dedicated to the project. The team's willingness to experiment and try new designs was extremely positive as we were able to rapidly change game mechanics in concert with feedback from testing.

While working with the Wii Remote was problematic as it took up much of our programming manpower, the final result more than paid off as it is somewhat more fun, and more importantly opens up the device to a disabled group that could not use it before. However, we might have done a better job making the Wii Remote accessible had we analyzed the feedback affordances of the device, as it was relatively unknown to part of our target audience. In retrospect a haptic device with force feedback might have worked better, however from a financial standpoint it would be unrealistic to expect users to purchase such controllers.

While the team also made several mistakes, we were lucky that none proved fatal. The most serious oversight was the eleventh hour addition of a third, untested song level. In addition to not sounding as polished as the other two levels, the addition actually introduced a game-crashing bug, and AudiOdyssey's release date was delayed by two weeks as we had to go back to an earlier build.

Another mistake was the choice of development environment; we felt Flash would allow us to put the game online as a web browser plugin, but as we had synchronization trouble and were forced to switch from .mp3 to .wav format to fix the problem. The result was that our sound files had a tenfold increase in size, and as we started making the game more complex and adding more sounds the file sized ballooned. Finally we realized that game would have to be downloaded and installed on the user's computer.

Perhaps the biggest disappointment during AudiOdyssey's development was the triage of multiplayer. Cutting the feature was especially hard for the team, as it was one of our original development goals and the design for multiplayer had been almost entirely specified. However once we saw how much time we were going to have to spend making the single player game we realized that it was not feasible to implement multiplayer as well in our short development period.

[14] Though she later went on to comment *"I like Guitar Hero better"*.

The final version of AudiOdyssey is not without its share of problems. The largest issue is the Wii Remote UI, which is not especially robust and has severe learnability deficits for the non-sighted. Swinging motions are not always recognized, feedback is not as obvious as is should be, and the interface is too difficult for all but the most adept users. As a result few visually impaired people were able to succeed with the Wii Remote. In hindsight we should have made an easier to understand and more interactive Wii Remote tutorial for the non-sighted audience. Luckily the alternative keyboard controls are quite straightforward, and most users who download the game tend to use this input scheme. Another usability problem is that the player must act a split second before the sound cues are fired, which means anticipating events rather than reacting to them. Because we don't rely on feedback in this regard frustration tends to rise when users miss series of notes they feel they should have triggered.

Despite these issues the majority of interviewed testers agree that the game is compelling and fun to play. Creating a rhythm game with silhouetted graphics turned out to be exactly the right decision. Sighted users felt the aesthetic worked since it put emphasis on the music, while non-sighted users didn't feel they were missing a crucial part of the game since the visuals aren't descriptive. The mechanic of building a song from component tracks turned out to be very compelling. Most surprising, though, was the popularity of the random sound generation during the bonus phase. Users really enjoy this feature, and it should probably have been featured as an even more prominent game mechanic.

One final interesting and unexpected result is that AudiOdyssey ended up actually being (technically) accessible to hearing impaired individuals thanks to redundant visual output for all critical audio cues. However being unable to hear the music likely makes the game much less fun, and we suspect that deaf players would find the title quite boring.

3.7 Next Steps

While the current version of AudiOdyssey successfully demonstrates many of the research project's goals there is still plenty of room for future work. Of the four research goals we were never able to implement remote multiplayer with identity masking. Luckily there are plans to make a spiritual sequel which will implement this feature, though likely with different game play and mechanics. It is conceivable that later versions may shift to consoles as well, a move that could greatly benefit the accessible gaming community since it is possible that the success of an accessible console game could open the floodgates to similar offerings.

Beyond AudiOdyssey and games for the sight impaired there are still many opportunities for accessible game development. There is still a relatively untapped market for accessible games, including both the creation of new games and adapting existing games to accessible control schemes. It is the hope of GAMBIT and the development team that AudiOdyssey will help illustrate how such design is feasible and worthwhile.

Chapter 4
Next Generation User Interfaces

Hopefully by now the importance of user interfaces in games is apparent. To wit: when Nintendo realized the Gamecube wasn't selling as well as they had hoped, and that Microsoft and Sony were about to release updated and powerful new consoles, they decided to try something different. Instead of compete in terms of processing power, they bet that consumers would rather have a system with a novel new UI. The result was the Wii, and as history has shown thus far the gamble seems to have paid off. An otherwise unimpressive system was made to shine thanks to a well designed UI.

The success of the Wii reflects what the author suspects is a shift in focus centering on the importance of a game's UI. For example, computing has become powerful enough that some games approach cinema-like quality, and current generation consoles are approaching maximum resolution for our display devices, but the result is that *new* improvements in graphics seem incremental compared to the huge leaps taken in the 1980's and 1990's. UIs however still have plenty of room for improvement, and it is likely that there will be much development in this realm over the next few years.

How user interfaces will evolve in the future is difficult to predict, but it is clear that there are currently many technologies which are just starting to be used in games, namely direct manipulation inputs, feature tracking and information lookup for context aware interfaces, and cross platform functionality. All these technologies currently exist, and there is no reason any of these couldn't be incorporated into current (or at least next) generation systems.

Will these new features translate into increased usability and accessibility? This question is even harder to answer than how UIs will evolve; if it is unclear exactly how UIs will improve in the future, it is nearly impossible to guess how games will make use of these advancements. But there is reason for hope. The main difference is that while past UIs tended to gravitate around a set of standards, future UIs look much more diverse and open.

Recall key idea #2: *Even if it is not clear how to make a game accessible, there are certain design principles which can be followed that tend to increase usability across the board. This increase in usability may in turn lead to accessibility*. As developers are forced to grapple with how to create and design these new interfaces they can (and hopefully will) keep the principles established thus far in mind. After all, incorporating accessibility into games and UIs from the outset makes the challenge much more manageable!

4.1 Direct Manipulation Interfaces

Perhaps one of the more interesting trends in the past few years has been the rise of direct manipulation control schemes, or the use of intuitive movements and motions for actions

in games[15]. Examples include strumming on a guitar controller to simulate a rock star in *Guitar Hero*, steering in a racing game by tilting the controller, and calling out your dog's name to get his attention in *Nintendogs* (DS, 2005). Direct manipulation can be contrasted with more classical indirect manipulation schemes in which an arbitrary action by the user translates into a different in game action, for instance when pushing a button causes a character to punch in a fighting game. Of course there are gray areas as well – under which category does tilting a joystick to cause *Pac-Man* to move fall?

Why have these control schemes recently seen a surge in popularity? Generally speaking they lose out in efficiency compared to general button and joystick UIs. In some ways they may be easier to learn if the manipulation is analogous to the real world, however they may be less learnable if feedback is lacking or non-obvious. Perhaps the most motivating argument is the increased sense of immersion the user feels, creating the illusion of being part of, or at least closer to, the game environment. Direct manipulation schemes frequently have a novelty factor leading to more memorable game play experiences. They also tend to attract attention from nearby viewers, as there is a "Hey, let me see that" moment when people see a new UI. The Nintendo Wii and DS are good examples of systems which benefit from these effects.

While current direct manipulation relies upon knowing where the controller is and how it is moving, future variants may also know the user's orientation and motion. Most people probably won't tolerate wearing clunky monitoring devices, so it is likely that much of this functionality will come from mounted cameras and microphones, or arrays thereof. Indeed, there has already been some preliminary commercial success in this realm; Sony's *EyeToy* is a camera system for the Playstation 2 that sold over 6.7 million units, and the new Playstation 3 *Eye* version could conceivably do much better [24].

Figure 19 – The Playstation 3 *Eye* has a camera and microphone for audio and visual input, and can be used as a direct manipulation interface

Despite this initial success, these audio/vision systems have not truly entered the mainstream yet, likely due to a lack of compelling games that use the UI meaningfully. If the game's controls feel awkward, have learnability deficits, or are less efficient than other available technology then the UI will likely hurt overall sales. Most of the EyeToy

[15] Examples of direct manipulation controls did appear earlier, like Nintendo's *Zapper* for shooting games.

and Eye games suffer from this problem as they are effectively elaborate tech demos showing off the new interface.

One solution to this is to augment vision and sound UIs with more standard button controls, and to use each component where appropriate. Vision systems have already shown the potential for personalized pseudo 3D environments using head tracking, and this feature could tie with standard controls for a highly immersive gaming experience [25]. This technology currently only works for single players as the effect breaks down when viewed from multiple vantage points, however there is reason to believe such obstacles may be overcome in the future (for instance through the use of personalized displays.)

Direct manipulation also concerns tangible interfaces and haptic feedback. Tangible interfaces are systems in which the user manipulates physical objects which also have a digital analog, which somehow augments the experience [26]. For example, a child playing with special blocks with embedded electronic ID tags could have a system read the position of the blocks and make an on screen visual representation of the structure. Haptic feedback is output from a system which stimulates the user's sense of touch, like rumbling vibrators in today's controllers. Future systems might be made to simulate touching different surfaces and objects, adding a third output channel (in addition to vision and audio) [27].

New haptic feedback systems are the most promising accessibility prospect of direct manipulation interfaces, as they can add complexity to interfaces for people accustomed to only one output. For instance, a blind user could get much more information out of a system with both audio and touch feedback[16]. The other variations of direct manipulation also tend to increase usability since systems based on them will likely be simple and highly learnable, therefore they may also increase accessibility.

4.2 Context Sensitivity

Modern computing systems are able to track, analyze, and store huge amounts of user generated data. Large companies like Google have made fortunes tracking user movement through websites and searches. What makes these data so valuable is that they are then used to generate advertisements and suggestions that are uniquely suited to the user. These ads and suggestions are context sensitive, as they draw on knowledge about the user's past actions, typical behavior, and likely goals to offer meaningful advice and advertisements.

Games can use similar tactics to improve playing experiences, and some current generation games offer minimal context sensitivity [28]. For instance, *SiN Episodes: Emergence* (PC 2006) is a first person shooter that tracks a number of player statistics to make guesses about the user's ability level and playing style [29]. It then uses these statistics to dynamically adjust the difficulty level of the game and ensure a natural sense of flow, hopefully keeping the experience from becoming too frustrating or too boring.

[16] This is one of the dimensions we would have liked to explore with AudiOdyssey.

The *Super Smash Bros*. series (first release Nintendo 64, 1999) has an auto-handicap feature in which player's abilities are monitored and weighted handicaps are handed out to achieve game balance.

Figure 10 – *SiN Episodes: Emergence* tracks the player's ability level and dynamically adjusts difficulty to make the experience more fluid. Can future games use context cues more meaningfully?

Theoretically, a system with a built in camera and microphone could use facial, gesture, and sound recognition software to provide context sensitive inputs by making guesses about whether the user is becoming frustrated, bored, distracted by others in the room, or immersed in the game play. Such information could serve as an invaluable set of inputs, allowing not only for adjusting difficulty level but aesthetic and mechanics. The system could also be expanded to allow for new forms of direct manipulation, like natural speech input. For example, if a player loses a level several times and then says "What the hell?" the system could guess from context clues that the game is too frustrating and adjust difficultly accordingly.

A potentially source of context sensitive user information is the internet. Systems drawing upon outside world knowledge of the user can then incorporate that information into the gaming experience, perhaps dynamically modifying levels or emphasizing game elements that particularly suit an individual's preferences based on previous experiences. Another likely result of such information will be the presence of targeted in-game advertisements. Of course, any data sent or received through the internet will have to be subject to security protocols to prevent the transmission of sensitive information [30].

One particular area that would highly benefit from context sensitivity is the educational and serious games genre. By making guesses about how much the user understands the game's subject content it should be possible to create games that will dynamically tweak both the difficulty level and presented content to suit the user, or student's, needs[17].

Context sensitivity could yield tremendous payoffs with mentally impaired gamers. One of the chief difficulties of designing games that are accessible to this group is the sheer range of disabilities. Context sensitive games and systems can get around this problem by

[17] This approach works with non-game educational software too, but wouldn't the game be more fun?

learning about the user and adapting to their needs dynamically, eventually adjusting enough to accommodate their playing style.

4.3 Cross Platform Functionality

As computing abilities proliferate both in terms of power and platforms we have more devices today that can be, and are used, for gaming than ever before. Modern home entertainment consoles, portable gaming devices, computers, music players, and cell phones are all either primarily intended for playing games or have strong support for such activities.

At the same time players are becoming heavily invested in certain games, especially massively multiplayer online (MMO) titles. One's in-game avatar becomes a digital extension of themselves, and large amounts of effort is spent on customization. Meaningful relationships are formed in these virtual worlds, with guilds, clans, and communities populated with virtual avatars proliferating. These interactions help keep players engaged and invested in the game [31].

Future games can take advantage of these trends by creating titles that work across different platforms. Games for the home console or computer can also have functionality built in for mobile devices to allow users to play wherever and whenever is most convenient. Such cross platform functionality will likely work best with MMOs due to the high level of investment, but it is possible that other games can use such functionality as well [32].

It is worth noting that when creating cross platform games the interactions will likely vary according to the affordances of the device and the environment. Portable device use suffers from potential situational impairments and a limited UI, therefore the game mechanics will have to make sense given these constraints. Complex, involved interactions will likely take place at home (like a raid on a rival guild), while simpler interactions will be more common through portable devices (like customizing one's avatar) [32].

A potentially problematic issue will be cheating due to illegal player modifications. It is quite possible that users may attempt to exploit data sent via servers by uncovering extra "hidden" information to gain a competitive edge. As a result cross platform games may have serious security concerns.

Cross platform technology could have terrific accessibility benefits due to the alternative modes that games would introduce. Different legitimate ways to play that work for functional disabilities would work just as well for permanent disabilities. Thus while impaired users might not be able to join every aspect of the game they should be able to collaborate with others in meaningful ways and play accordingly. If social interactions are constructed correctly these games could be quite compelling.

4.4 Next Generation Game Systems

All of the technology described thus far already exists, either commercially or in laboratory research settings. Current consoles and computers are either powerful enough or almost powerful enough to run the algorithms that would be required to process and output the information required to make these systems reality. Hopefully the next generation of game consoles will make use of these features for their user interfaces, or go even further into new unexplored realms.

What follows is a vision of such a future gaming experience. A MMO with a first person shooter mechanic is played using a Wii Remote-like device to aim. A video camera does body and head tracking, and surrounding speakers produce spatial sound output. The game's 3D environment responds to the position of the user's head and eyes, panning in the appropriate direction as the angle of the head and eye focus changes. On easier difficulty settings the game also uses these inputs to assist the user with aided aiming functionality. Spatial sound cues help the user react accordingly, with sound effects like gunshots coming from the appropriate direction.

When a second player joins the game asks for a username, and once identified the system automatically downloads information about the new user. The screen is split down the middle for two separate displays, with each half reacting to one user's head position to create a pair of "personalized" 3D environments. This has the benefit of obscuring each player's side from their opponent[18]. The game could then offer a competitive challenge that suits the style of interaction these players are known to have enjoyed in the past. Monitoring their facial expressions and gestures, play would continue with no major changes if the players are sufficiently immersed, or if it sensed the players are getting frustrated it could quietly tweak mechanics or suggest a new game setting.

After the play session is over, one of the users accesses the same game on his cell phone while riding home on the bus. This is not the same first person part of the game that was played before, but rather a special segment that works well for the cell phone's smaller screen and the noisy environment, namely purchasing new items for his avatar at the store and chatting with other friends in a guild.

This game could be very accessible to a wide range of disabled users as well. High fidelity audio and video would make the game more usable for those playing. Context sensitivity could detect certain shortcomings and compensate, perhaps by taking over partial control for the player and leaving a simpler set of actions for the user to handle. With thoughtful planning future UIs can be very accessible indeed.

[18] Generally, obscuring screens in competitive games is good since it allows giving each player "private" information.

4.5 Summary

It is likely that many of the upcoming innovations in gaming will come from new developments in user interface technology. Direct manipulation UIs, context sensitivity, and cross platform titles will all result in new ways to play. Combining these technologies can lead to a new generation of games and experiences that are fun and exciting for a broad user base. If thoughtfully conceived these technologies can lead to added usability and accessibility for systems as well.

It is the author's hope that future user interfaces will allow all people, regardless of ability, to play the same popular games. Hopefully the guidelines presented herein will help developers achieve this goal. Thanks so much for reading this thesis.

References

[1] J. Ferrell, "Re: Designing Games that are Accessible to Everyone," Gamasutra article response forum, Februrary 14, 2008, http://www.gamasutra.com/view/feature/3538/designing_games_that_are_.php.

[2] United States Legislation, Section 508 of the Rehabilitation Act, August 7, 1998, http://www.section508.gov/index.cfm?FuseAction=Content&ID=3.

[3] United States Census, "Prevalence of Disability by Age, Sex, Race, and Hispanic Origin," United States Census Bureau, Washington D.C., 2002, http://www.census.gov/hhes/www/disability/sipp/disable02.html.

[4] Brookhaven Bulletin, Brookhaven National Laboratory, "The First Video Game," March 13, 1981, http://www.bnl.gov/bnlweb/history/higinbotham.asp.

[5] D. Lopez, "A History of Game Controllers," Carnegie Mellon University, Pittsburgh, Pennsylvania, May 17, 2007 http://lombardi.cfa.cmu.edu/infovis/exercises/10-FINAL-PROJECT-Phase-5-The-Archive/sketches/dlopez.

[6] United States Census, "Selected Disability Measures," United States Census Bureau, Washington D.C., 2002, http://www.census.gov/hhes/www/disability/sipp/disable02.html.

[7] D. Flűck, "Colorblind Population," post to Colblindor, April 28, 2006, http://www.colblindor.com/2006/04/28/colorblind-population/.

[8] C. A. Schoenborn and K. Heyman, "Health Disparities Among Adults With Hearing Loss: United States, 2000-2006," Center for Disease Control and Prevention (CDC) National Center for Health Statistics (NCHS), May 2008, http://www.cdc.gov/nchs/products/pubs/pubd/hestats/hearing00-06/hearing00-06.htm#Ref1.

[9] United States Legislation, Americans with Disabilities Act of 1990, July 26, 1990, http://www.ada.gov/pubs/ada.htm.

[10] "What Blind Gamers Want the Industry to Know… ," Ed. R van Tol, S Huiberts, presented at the Game Developers Conference, San Francisco, California, 2006.

[11] R. van Tol, S. Huiberts, List of games accessible to blind gamers, 2008 http://audiogames.net/listgames.php.

[12] GMA Games, "Shades of Doom Version 1.2," (game) 2005, http://www.gmagames.com/sod.html.

[13] T. Westin, "Game accessibility case study: Terraformers – a real-time 3D graphics game," in *Proceedings of the 5th International Conference on Disability, Virtual Reality and Associated Technologies,* Oxford, UK, 2004.

[14] D. Grammenos, A. Savidis, and C. Stephanidis, "UA-Chess: A Universally Accessible Board Game," in *Proceedings of the 3rd international conference on Universal Access in Human-Computer Interaction*, Crete, Greece, 2005.

[15] R. Miller, "Heuristic Evaluation," class notes for 6.831, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, Spring 2008.

[16] J. Nielsen, *Web Usability*. Apogeo Editore, 2000.

[17] D. Norman, *The Design of Everyday Things*. Doubleday, 1990.

[18] P. Hannukainen, "Disabled Persons as Lead Users in Mobile User Interface Design," M.S. thesis, Helsinki University of Technology, Helsinki, Finland, 2005.

[19] E. Folmer, "Gordon's Trigger Finger," (game) 2007, gtf.eelke.com.

[20] D. Grammenos, A. Savidis, and C. Stephanidis, "Unified Design of Universally Accessible Games," in *Universal Access in Human-Computer Interaction. Applications and Services*, Springer Berlin / Heidelberg, 2007, pp 607- 616.

[21] D. Grammenos, "Game Over!," (game) 2006, http://ua-games.gr/game-over/.

[22] E. Walker, S. Stanfield, B. Alfieri, C. McGarry, J. Saucedo, "Strange Attractors," (game) 2005, http://www.ominousdev.com/games.htm.

[23] L. Wyse, "A Sound Modeling and Synthesis System Designed for Maximum Usability," in *International Computer Music Conference*, Sinagpore, 2003, pp 447-451.

[24] E. Gibson, "PS3 has outsold Xbox 360 in Europe," May 6, 2008, http://www.eurogamer.net/article.php?article_id=137142.

[25] J. Lee, "Head Tracking for Desktop VR Displays using the Wii Remote," Carnegie Mellon University, Pittsburgh, Pennsylvania, December, 2007, http://www.cs.cmu.edu/~johnny/projects/wii/.

[26] H. Ishii, and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms," , in *Proceedings of Conference on Human Factors in Computing Systems CHI '97*, Atlanta, March 1997, ACM Press, pp. 234-241.

[27] V. Hayward, and J. M. Cruz-Hernandez, "Tactile Display Device using Distributed Lateral Skin Stretch," in *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, November, 2000.

[28] R. Hunicke, and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in *Challenges in Game Artificial Intelligence AAAI Workshop*, Northwestern University, Evanston, Illinois, 2004.

[29] J. Ocampo, "GDC '06: SiN Episodes: Emergence Hands-on," March 23, 2006, http://www.gamespot.com/pc/action/sinepisodes1/news.html?sid=6146567&mode=previews.

[30] J. Canny, "The Future of Human-Computer Interaction" in *ACM Queue*, Volume 4, Issue 6, pp 24-32, 2006.

[31] C. Steinkuehler, D. Williams, "Where Everybody Knows Your (Screen) Name: Online Games as "Third Places"," in *Journal of Computer-Mediated Communication*, Volume 11, Issue 4, pp 885-909, July, 2006.

[32] D. Roy, "Mastery and the Mobile Future of Massively Multiplayer Games," M.S. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, May, 2007.

Appendix
AudiOdyssey Game Development Timeline

The game was developed over two month period, split into four iterations of two weeks each. The development timeline of each iteration is presented here.

Iteration 1 (weeks 1 and 2)

Goals: The goal of the first iteration was for the team to learn about the fundamental research behind their project by studying and analyzing existing accessible games, and then to come up with both initial game designs and beginning functionality.

Testing: No testing took place at the beginning of the first iteration, though the team itself did test out several different accessible games in an effort to analyze and understand different types of UIs so that particularly meaningful interactions could be emulated.

Progress: The team got to know each other, learned how to work together, and learned about the subject content of the game. During this period the team also learned how DJs (Disc Jockeys) perform by watching documentaries, meeting real DJs, and even trying DJing ourselves. We also began proposing theoretical game designs that would meet our various research goals. Ideas were rapidly discussed, refined, and then either discarded or iterated upon to serve as a foundation for development. Similar rapid idea generation took place for all aspects of game design, including visual and audio style and quality assurance/testing methodology. Meanwhile the programmers created a simple interface for the Nintendo Wii Remote to demonstrate functionality and to familiarize the team with the device's capabilities.

Problems: By the end of the first iteration it was starting to become apparent that working with the Wii Remote without a developer's kit would be challenging. While game design was progressing a finalized version still had not been agreed upon.

Results: At the close of the first iteration the team had created an orange background with bouncing silhouette figures that moved with the beat of the music. The user could overlay simple two second music loops by pressing buttons in time with a separate rhythm track. Each loop featured a different instrument, so overlaying the loops felt like creating a song from separate pieces.

Iteration 2 (weeks 3 and 4)

Goals: Iteration two's goals were to finalize unfinished game play design from iteration one, create new, more engaging artwork, start creating an original song for the game, incorporate Wyse's sound generation software, and begin work on the multiplayer mode.

Testing: Feedback from user testing of the iteration one demo yielded useful suggestions for changing game dynamics, such as increasing intervals between action cues. Kestrell,

our blind consultant, played this version and offered many helpful insights such as how non-sighted users can best learn about the Wii Remote. Namely, she suggested letting users touch it all over thoroughly and attempt to push the all buttons with helpful feedback. She also critiqued our game design.

Progress: Over the iteration most aspects of the game were improved and refined, and only minor changes to game mechanics and flow were implemented. Design continued to swing between several concepts for matching game beats. A silhouette motif was chosen for the visual style, as we believed that sighted players would focus on the audio without feeling as though the visuals were under-polished, and non-sighted players would be less likely to feel they were missing a critical game element. The Wii Remote scheme was successfully integrated, but controller sensitivity turned out to be a large stumbling block. Setting usable thresholds that prevented false positives from triggering took up a large amount of programming manpower. Music synchronization woes forced the team to switch file formats, which increased the size of the game by an order of magnitude and prevented the team from embedding the game in a browser; the user would have to download and install the application.

Problems: The team had fallen behind in iteration one due to overzealous goals. While partially defined, the game design had not been completed and it was unclear exactly how the user would play AudiOdyssey. The music loops were only two seconds long, and no full length tracks had been created. Integrating robust Wii Remote controls was turning out to be much more difficult than expected, and only partially functionality had been achieved.

The most unfortunate result of these problems was the realization that the team had bitten off more than we could chew. While the game had initially called for an online multiplayer element, we realized that we would not have time to implement both it and the single player mode. Despite having spent a good amount of time designing multiplayer we cut the feature to keep the quality of the game as high as possible.

Results: At the end of the iteration the team had a new demo set in a nightclub. The audio files were the same as those used in the first iteration, but they looped for longer, had no synching problems, and spread out the notes over larger intervals. The Wii Remote UI was functional but had serious usability issues concerning learnability and error prevention. Single player design was converging but had several unresolved issues, and testing for the iteration was virtually non-existent due to the testing lead's shift over to assist with design. Much progress had been made, but many tasks remained to be completed.

Iteration 3 (weeks 5 and 6)

Goals: Iteration three had a heavy emphasis on several key points: finalizing game design, adding full songs, and getting the Wii Remote working correctly.

Testing: With design settled and new builds every other day, quality assurance held testing sessions with both visually impaired and sighted users. Feedback indicated the game needed more polish, the controls were difficult to use, and that the game concept was certainly fun. Blind testers had problems using the Wii Remote as the device was completely different from standard UIs, so the team added an accessibility component detailing how to use the Wii Remote effectively.

Progress: The design was settled early in the iteration, with the simplest scheme winning out. The sound lead began working two external audio students on original game scoring. Alternate keyboard controls were incorporated which turned out to be more robust.

The team also worked on several smaller tasks critical to the game's overall look and feel. Interstitial material such as a main menu and the ability to exit and restart within the program were added. Specialized sound generation software created by Wyse was integrated to let users create their own sounds during bonus sections. The programmers also integrated lots of new art and sound assets, leading to a more immersive gaming environment.

Problems: The programmers spent an inordinate amount of time massaging the Wii Remote controls, which only started functioning reasonably by the end of the iteration. Concerns were popping up about the team staying in the office later and later each day (frequently past 8pm) and worries about completing and integrating all of the project components were growing.

Results: The iteration three demo showed drastic improvement over the previous build. The nightclub environment showed a DJ avatar spinning turntables as silhouetted audience members danced in the foreground in time with the music. A semi-complete menu system allowed the user to browse through options to start the game. An original song (which would later be named "Endless") had been added, and though it still needed polish the song was more interactive than the old two second loops. Wyse's sound generation tools had been added, but not tested. All of these loose ends would have to be tied up by the end of the following iteration.

Iteration 4 (weeks 7 and 8)

Goals: Complete and integrate all remaining assets, and produce a stable build of the game for installation.

Testing: Iteration four began with the first thorough evaluation of AudiOdyssey by a large group of subjects. The testers enjoyed the game's new look, but it was discovered that the controls were still too difficult, and that the beat matching mechanic occasionally sounded unimpressive due to the varied song structure. The sound generation component, although confusing, turned out to be a highly enjoyable and silly game section, and was permanently added to the game's set of mechanics.

Progress: The sound team worked diligently refining the sound effects to make cues more learnable, so users wouldn't have to ask the game for hints. Endless was simplified to make the various tracks sound more similar, which counter-intuitively actually resulted in the song sounding better since it sounded complete even with many of the instrumental tracks removed. Work began on "Hyper Act", a second song with a reggae/rock aesthetic. Meanwhile, the two second loops used in previous builds were swapped into a tutorial level.

During this period the final art assets were added to the game, and game bugs were methodically identified and resolved. New DJ and turntable animation gave the user a more distinct impression of controlling an in-game avatar, while a refined audience and club environment made success feedback much clearer. Bug fixes and asset integration occupied the majority of the programmer's time, and the little time left was devoted to adjusting difficulty level and the Wii Remote controls.

Problems: While the team made respectable progress it suffered from problems due to last minute crunch. Team members were staying longer hours each day with only an incremental increase in productivity, and our management methodology was abandoned as it was deemed too time consuming and not useful enough during the final crunch. However, the biggest problem encountered was the eleventh hour addition of Hyper Act as an untested feature. It had to be removed later on, after the project finished, due to undiscovered bugs.

Results: AudiOdyssey was completed! See chapter 3.4 for details on the final version.

Final Development Thoughts

Overall, the team did an exceptional job with an extremely difficult task and a short production timeline. Over the four iterations the rate at which useful code and assets were produced increased dramatically, and the team accomplished much more in the final iteration than they had in the first two combined. Despite long hours together in close quarters the team emerged very friendly with each other, and there were no serious arguments whatsoever. AudiOdyssey was completed as a playable game which satisfied all of the initial research goals, with the exception of multiplayer. Luckily future spiritual sequels with this functionality will likely be created.