Ametista: a mini-toolkit for exploring new window management techniques

Nicolas Roussel
Laboratoire de Recherche en Informatique (LRI) & INRIA Futurs*
Bât 490, Université Paris-Sud
91405 Orsay Cedex, France
+33 1 69 15 66 23
roussel@lri.fr

ABSTRACT
Although the HCI research community has contributed a number of metaphors, interaction techniques and layout algorithms to improve window management tasks, most of these ended as prototypes and only a few were implemented in real window managers. In this paper, we present Ametista, a mini-toolkit designed to facilitate the exploration of new window management techniques using both low-fidelity prototyping and a high-fidelity approach based on X Window application redirection.

Keywords
Window management, graphical interaction, prototyping, application redirection, OpenGL, X Window system, VNC

INTRODUCTION
In [16], Myers defines a window manager as "*a software package that helps the user monitor and control different contexts by separating them physically onto different parts of one or more display screens*". He adds: "*Before window managers, people had to remember their various activities and how to switch back and forth*". Twenty years after the general adoption of the desktop metaphor [12] and overlapping windows [5], the growing range of activities supported by interactive computer applications has brought us back to the point where it is again difficult to remember these activities and organize them.

Over the past few years, a number of novel metaphors, interaction techniques or layout algorithms have been proposed to extend or replace the desktop metaphor such as the *pile* metaphor [14], *elastic windows* [13], *tabbed, rotated and peeled back windows* [2] or constraint-based layout [1]. However, most of these proposals were never implemented in a 'real' window manager. *Piles*, for example, were prototyped with Macromind Director; *elastic windows* were implemented within custom applications and *rotated and peeled back windows* were prototyped in Tcl/Tk.

With the advent of hardware-accelerated graphics, the graphics libraries available to application developers have tremendously improved in recent years. Performance of graphics hardware is increasing faster than Moore's law, supporting more and more advanced graphics functions. The Direct3D and OpenGL libraries, for example, natively support arbitrary 3D transformations, double buffering, Z-buffering, alpha blending, texture mapping and material lighting. As illustrated by [3], all these graphics functions make it possible to efficiently implement advanced graphical interaction techniques such as *toolglasses* and *magic lenses* [4] or *zoomable interfaces* [18].

By contrast, the graphical libraries used for window management have not followed this trend, making it difficult or impossible to use texture mapping, alpha blending or arbitrary geometric transformations at the level of windows. Until recently, the three most popular windowing systems were still based on graphics libraries designed in the 1980s: GDI for Microsoft Windows, QuickDraw for Apple Mac OS and the Xlib for the X Window system. The graphics models associated to these libraries were relatively simple. In particular, the color of each pixel on the screen was determined by a single application through a few logical operations (e.g. *and*, *or*, *xor*) applied on elementary 2D primitives. The main reason why these models were so simple is that the hardware available when they were designed was barely powerful enough for them. It actually took several years before GDI, QuickDraw or X server implementations could take advantage of hardware acceleration provided by consumer-level graphics hardware.

We believe that the large difference between the graphics models available to applications and window managers is one of the reasons why many innovative graphical interaction techniques were never taken to the point where they can be used in a real window management context. To address this problem, we introduce *Ametista*, a mini-

* projet In Situ, Pôle Commun de Recherche en Informatique du plateau de Saclay, CNRS, Ecole Polytechnique, INRIA, Université Paris-Sud

toolkit specifically designed for HCI researchers who want to explore new window management techniques.

RELATED WORK

In this section, we briefly describe the current state of the rendering and windowing systems of Apple Mac OS and Microsoft Windows as well as the X Window system. We also describe several research projects related to the exploration of new window management techniques in a real-use context.

Quartz Compositor

The windowing system of Apple Mac OS X is based on three different libraries: Quartz for 2D graphics, OpenGL for 3D graphics and QuickTime for dynamic media (e.g. animated graphics and video). A fourth component, the Quartz Compositor, is responsible for the composition and display of graphics rendered with these three libraries.

Quartz offers high-quality screen rendering and printing. It is based on the Portable Document Format (PDF) graphics model and features a number of advanced 2D graphics capabilities such as spline-based curves, text rotation, transparency and anti-aliased drawing. The move from the old QuickDraw graphics model to this new one allowed significant visual changes in the user interface of Mac OS. Semi-transparent menus and controls, drop shadows or "fade away" effects that were once limited to a few applications are now available to all through the standard Mac OS graphical user interface toolkit.

The Quartz Compositor is based on the idea that the window system can be considered as a digital image compositor [11]: Quartz, OpenGL and QuickTime graphics are rendered into off-screen buffers that are then used as textures to create the actual on-screen display. As a matter of fact, since Mac OS X v10.2, the Compositor is just another OpenGL application. As such, it can take advantage of hardware-accelerated graphics functions to transform windows in real-time before composing them. Examples of transformations include alpha blending, color fading or geometric transformations, as the *scale* and *genie* effects shown when windows are minimized.

The introduction of Quartz and the Quartz Compositor in the graphics and windowing systems of Mac OS illustrates the potential uses of richer graphical models for supporting new graphical interaction techniques and therefore, new window management techniques. However, the windowing system of Mac OS is tightly coupled with the operating system, which makes it difficult - if not impossible - to access and modify. Although the image compositing approach coupled with hardware-accelerated rendering seems promising, the Quartz Compositor in its current state cannot be used by HCI researchers to explore new window management techniques.

The Task Gallery and Windows Longhorn

Microsoft's Task Gallery [20, 24] uses a redirection mechanism for hosting existing Windows applications in a 3D workspace without changing or recompiling them. By taking advantage of the powerful graphics model of Direct3D, the authors created a 3D window manager that better takes into account the human perception and spatial cognition. This example clearly shows again how a rich graphics model can significantly change the user's interactions with applications and documents.

According to its Web site[1], "the Task Gallery is not a future version of the Windows operating system or user-experience". Yet, recent talks from Microsoft at the Windows Hardware Engineering Conference clearly state that the windowing system of the future versions of Windows will be based on Direct3D and a compositing process [15].

The Task Gallery is a stand-alone application. However, in order to implement their redirection mechanism, the authors had to modify Windows 2000. As they are not allowed to release the patches corresponding to these modifications, the Task Gallery and its redirection mechanism remain out of reach for HCI researchers, like Apple's Quartz Compositor.

The X Window system, Render and RandR

A key feature of the X Window System [22] is that any user-level application can act as a window manager. As a consequence, a large number of window managers have been developed for this system, providing a large range of appearances and behaviors. Yet, all these window managers are based on the original X graphics model and therefore, they differ mostly in minor details such as window decorations or keyboard shortcuts, and not in their operation principle. Most windows remain rectangular and opaque, very little use is made of the advanced features of modern graphics hardware and the interaction techniques and metaphors remain the same.

The mismatch between the original X Window rendering system and modern interactive graphical applications is very well described by K. Packard in [17]: "*The two new open source user interface environments, Gnome and KDE, were hamstrung by the existing X rendering system. KDE accepted the limitations of the environment and made the best of them. Gnome replaced server-side rendering with client-side rendering turning the X protocol into a simple image transport system. The lack of hardware acceleration and the destruction of reasonable remote application performance demonstrated that this direction should be supplanted with something providing a modicum of server-side support.*"

The X Rendering extension (Render) [17] and the Resize and Rotate extension (RandR) [10] were designed to address many of the shortcomings of the original X rendering architecture. These two extensions provide image compositing operators and glyph management and allow applications to resize, rotate, reflect and change the refresh rate of an X screen on the fly. XFree86 4.3.0 partially implements the Render extension, providing anti-aliased text drawing and image composition. Support for the RandR extension has also been partially integrated, providing support for resizing the root window at run-time.

---

[1] http://research.microsoft.com/ui/TaskGallery/

The recent changes in the X Window rendering system coupled with its openness and extensibility makes it more and more usable to explore new graphical interaction techniques. However, basic functions of the Render extension such as affine transformation of images remain to be implemented in existing X servers. Even then, the graphics model of X will still be far simpler than the one of OpenGL, for example. Therefore, OpenGL-based applications will remain graphically richer than any possible X window manager for some time.

VNC-based approaches to new workspace interaction techniques

As we have seen, the graphics and windowing systems of the three most popular platforms make it difficult if not impossible for HCI researchers to take advantage of modern graphics hardware to explore new graphical interaction techniques for window management. In order to overcome these difficulties, a number of researchers are using the VNC remote display system [19] to bring existing desktops and applications into innovative workspaces.

The Three-Dimensional Workspace Manager (3Dwm) [8] includes a VNC viewer implementation that makes it possible to integrate traditional graphical desktops into an immersive 3D environment implemented with OpenGL. In a similar way, Shiozawa et al. use VNC to combine several individual desktops into a perspective layered collaborative workspace [23]. Denoue et al. also used VNC to capture window contents and display them as paper flyers posted on a virtual board [7].

These examples show how VNC can help create innovative workspace interactions without modifying the operating system or the window system. However, in the first two examples, the documents and applications are still displayed and manipulated through the traditional desktop interface, which is simply mapped as a whole inside a new environment. The third example is more interesting regarding window management techniques, although individual windows are captured at regular intervals through a polling mechanism, which, as the authors admit, is not responsive enough to content changes.

GENERAL OVERVIEW OF AMETISTA

Ametista is a mini-toolkit designed to facilitate the exploration of new window management techniques. It supports low-fidelity prototyping, similar to the Director or Tcl/Tk prototypes described in [14] and [2], as well as high-fidelity prototyping using real applications, as in [20].

The current implementation of Ametista supports three types of windows:

- *pseudo-windows* that are randomly-colored rectangles;
- *placeholders* that display a fixed image or a video stream;
- live windows of X Window applications.

Pseudo-windows can be used for low-fidelity prototyping in the early stages of the exploration of a new window management technique. Placeholders can help getting a better idea of the envisioned technique by displaying snapshots or movies of real applications. Finally, live X windows can be used for high-fidelity prototyping and evaluation of the technique. The three kinds of windows can be freely mixed, as shown in Figure 1.

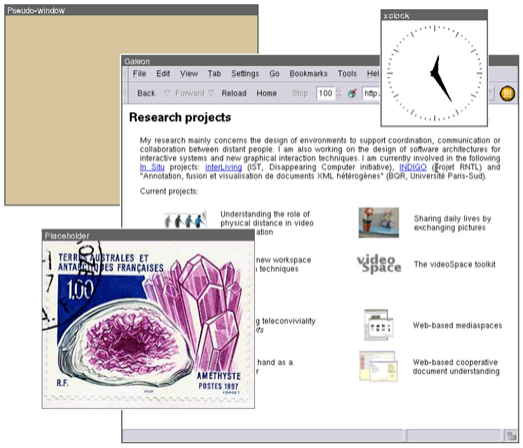

Figure 1: The three window classes of Ametista: a pseudo-window (top-left), a placeholder showing a JPEG image (bottom-left) and two live X Window applications (xclock and the Galeon Web browser).

Ametista uses OpenGL to display windows. As we explained in the previous section, this library offers a rich graphics model well adapted to the exploration of new window management techniques. Alpha blending, for example, makes it easy to create translucent objects and shadows. Scaling, rotation and translation can also be used with a perspective projection to position windows in $2^{1/2}$D or 3D, as illustrated by Figures 2 and 3.

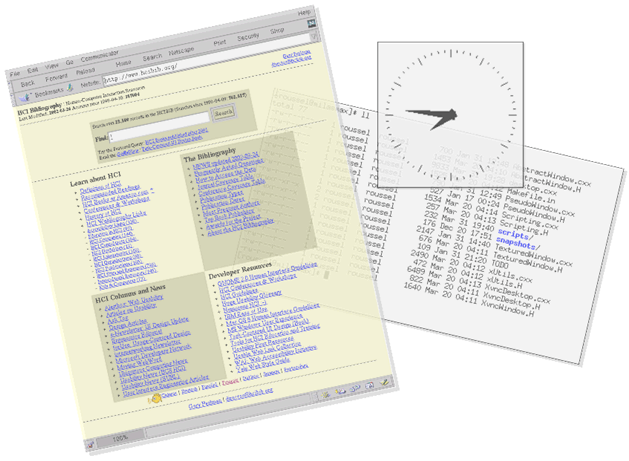

Figure 2: Combining 2D transformations, shadows and transparency.

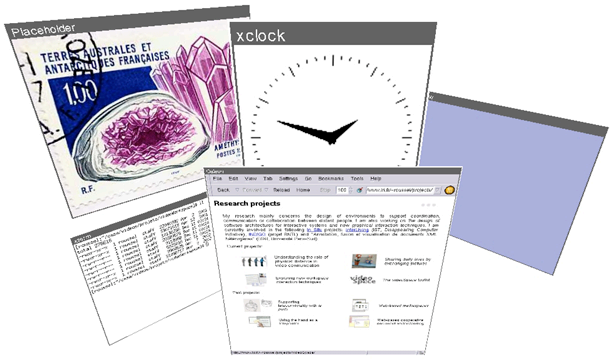

Figure 3: Arranging windows in 3D space.

Ametista makes an extensive use of texture mapping. Textures are used to display fixed images and video streams in placeholders as well as the content of X windows. They also make it possible to transform the window shapes in real-time. Figure 4 shows two examples of such transformations: a peeled back window (Galeon), as described in [2], and two windows cropped to circular shapes (xclock and a placeholder showing a JPEG picture).

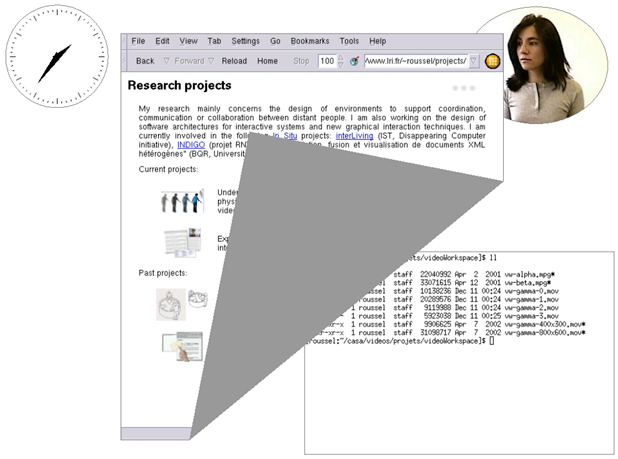

Figure 4: Examples of window shape transformations using texture mapping.

IMPLEMENTATION DETAILS

Ametista is implemented in C++ and uses the videoSpace toolkit [21], OpenGL and VNC. The Ametista software alone consists of about 2500 lines of code. The three window classes described in the previous section (pseudo-window, placeholder and live X window) each correspond to a C++ class that derives from *AbstractWindow*. This class gives access to the content of the window (color or texture) as well as geometry information for mapping screen coordinates to window coordinates.

Each window object has an associated *AbstractWindowRenderer* object. Developers will typically derive this class to redefine methods such as *pointerEvent* or *keyEvent* to manage mouse and keyboard events that occur in the window or *display* to redisplay it when the content has changed. Several renderer classes have been implemented to experiment with transparency, shadows or interactive animations such as the peeling-back effect.

In addition to the implementation of the three window classes and a set of renderers, Ametista also provides the skeleton of a generic window manager that can be customized to implement specific layout and interaction techniques.

X Window application display redirection

We use an approach similar to the redirection mechanism of [24] to make X Window applications available in Ametista. Our approach is based on the X Window version of the VNC remote display system.

VNC consists of two user-level applications: a server that generates the display, and a viewer that draws the display on a screen, receives mouse and keyboard events and forwards them to the server. XVNC, the VNC server implementation for X Window, is a slightly modified but fully functional X server. This server renders applications off-screen, making the desktop image available to VNC viewers (Figure 5), and forwards mouse and keyboard events to the appropriate applications.

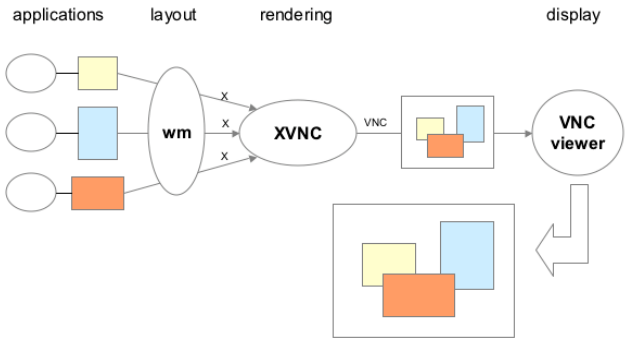

Figure 5: Standard XVNC remote display operation. Note that the window manager (wm) is not part of the VNC system.

The videoSpace toolkit implements the viewer side of VNC as an image source: new desktop images become available from this source whenever display updates are received from the VNC server. This provides Ametista with a real-time stream of images of the X Window desktop[2]. Note that, as opposed to [7], desktop images are pushed by the VNC server to Ametista and not pulled at regular time intervals, which ensures a good response time to application changes.

In order to extract the images of individual applications from desktop images, Ametista also implements the window manager used by the XVNC server. This window

[2] a previous version of Ametista was called VideoWorkspace to reflect the fact that the VNC desktop is seen as a video stream by our compositing process

manager simply tiles the windows next to each other so they do not overlap (Figure 6).

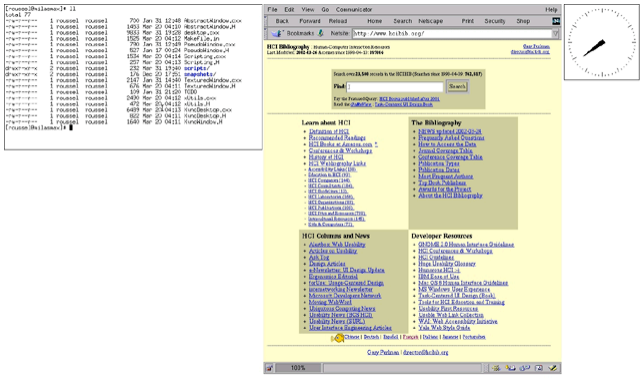

Figure 6: Sample tiled layout of an XVNC desktop managed by Ametista.

Ametista uses the XVNC desktop image as a texture. Whenever it is notified that part of this image has changed, it updates the texture and notifies the corresponding window objects. The *display* method of the renderers associated to these objects uses the size and position communicated by the window manager to set the appropriate texture coordinates. In order to reduce memory usage and achieve better performance, Ametista uses several OpenGL extensions to handle non-power of two textures and to avoid unnecessary memory copies between image data and textures.

Figure 7 summarizes our output redirection mechanism. Note that this approach differs from 3Dwm or the perspective layered workspace from [23], in that Ametista is able to extract images of individual applications and not images of the desktop as a whole. The same effect could be obtained without VNC. For example, we could modify an existing X server, as described in [9], or use a more platform-specific technique.

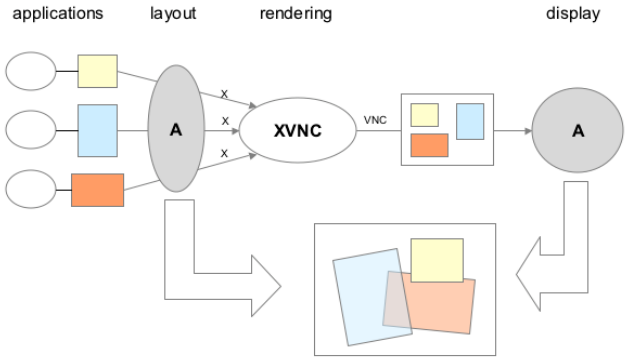

Figure 7: Output redirection of X Window applications using Ametista (VW).

Input handling

Ametista uses OpenGL selection mode and picking to assign the keyboard focus to the window under the mouse. Keyboard and mouse events can be handled locally by the Ametista application, e.g. to implement window management operations such as moving a window. They can also be forwarded to the proper X Window application, in which case the pointer coordinates are transformed into the local window coordinates system.

Window creation and destruction

Ametista can read commands from the standard input to create pseudo-windows and placeholders and to connect to XVNC servers. Each command specifies a rendering class, a window class and some additional parameters such as width, height and title for pseudo-windows or a URL for XVNC servers. As an example, the following commands were executed to set up the windows shown in Figure 1:

```
decorated PseudoWindow 400 300 Pseudo-window
decorated PlaceHolder demo/amethyste.jpg Placeholder
decorated XvncDesktop vnc://127.0.0.1:1
```

When the connection with an XVNC server is first established, all windows existing on this server are automatically added to Ametista's workspace. X applications can then create and destroy windows at will.

A video-enabled application

The videoSpace toolkit provides Ametista with a variety of image sources to be displayed on placeholders. These sources include JPEG and PNG images, QuickTime and MPEG movies but also live video input (e.g. a webcam) as well as networked image sources. As videoSpace image sources are described by URLs, they can be specified at run-time with *PlaceHolder* commands such as the one above.

VideoSpace also provides several image sinks that make it possible to record Ametista's display as a QuickTime or MPEG file or to send it over the network to another application. We already took advantage of this to create short video clips demonstrating Ametista. But most importantly, we anticipate that this feature will be especially interesting for observing users during evaluations and keeping records of these evaluations.

Performance evaluation

We conducted a preliminary evaluation of the performance of Ametista with the images presented in this article. The software ran on a Fujitsu/Siemens PC with a 1.5 GHz Pentium IV and an AGP NVidia GeForce2 MX 400. The operating system was Linux Mandrake 9.0. The screen size was 1280x1024 and the XVNC desktop size was 1280x1024. Ametista achieved full-screen display rates of up to 65 frames per second. Display rates of more than 30 frames per second were also achieved on a 667 MHz Apple PowerBook G4 with an AGP ATI Radeon Mobility.

DISCUSSION

The work presented in this paper relies on the assumption that richer graphics models will allow significant changes in window management techniques in the near future. But what kinds of changes do we expect? In this section, we take several basic features of modern graphics libraries such as OpenGL or Direct3D and explain how we think these features will help us create innovative graphical presentations and interaction techniques for window management.

The third dimension

3D user interfaces are very controversial. On one hand, user studies like [20] show that placing documents and applications in 3D space helps users remember where they are during later retrievals. Yet, other studies like [6] tell us that performance deteriorates as the freedom to locate items in the third dimension increases and that 3D interfaces can be perceived as more cluttered and less efficient than 2D or $2D^{1/2}$. On a less academic perspective, endless discussions about the potential benefits and disadvantages of 3D interfaces (including window managers) are also regularly posted on discussion forums[3].

Most comments in these discussions are related to the frequent navigation problems encountered in 3D interfaces and the need for better input techniques. It is true that a 3D *drag-and-drop* operation on a window might require a lot more concentration and effort than its 2D equivalent. However, specific devices such as isometric joysticks and spaceballs or even better, bi-manual interaction techniques, can help solve these problems.

Many other comments point out that reading, writing and drawing cover a fair amount of our uses of computers and are almost always associated to 2D surfaces. Some people think this makes 3D interfaces inadequate for these tasks. However, when dealing with physical objects, whether 2D or 3D, we perceive them and manipulate them in a 3D world. The same could be true for the digital world. The interesting problem is to find the appropriate interaction techniques and we believe that window management is a good test case for these techniques as it is an unavoidable task.

On a more pragmatic perspective, the third dimension combined with the depth test offers a convenient way to implement multi-layer graphical applications. Each layer can be associated to a particular depth, which can reduce the need for specific data structures. The activation of the depth test allows to render objects in arbitrary order, pixels being updated only if the current object is closer than the one already displayed (if any). Ametista already uses this approach to display overlapped windows, assigning a different depth to each window.

Geometric transformations

Moving windows (translating them) has always been possible since the adoption of the overlapping model. Scale transformations have almost never been possible. Note that the resizing of a window is usually not a scale transformation since it changes the layout of the window instead of just making the content bigger or smaller. The closest things to scale transformations of windows are the icons used in several systems that show a reduced version of the original application display. Rotations of windows have never been possible until the RandR extension of the X Window system that allows to change the orientation of the whole display.

Scale transformations have been used to create zoomable interfaces for a while [18]. While scaling the whole workspace might not be a good idea, we believe that scaling individual windows will be much more interesting. Translations of objects combined with a perspective view allow to *move away* some objects, making them smaller, and *bring closer* some others. We are currently using Ametista to explore this kind of interactions with windows (Figure 8).

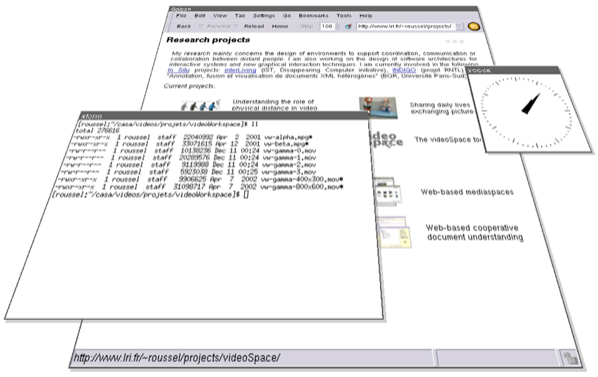

Figure 8: Example of perspective view.

We anticipate that rotations might play an important role in the future. In situations like Figure 8, viewpoint rotations allow to explore the three dimensions in a continuous way. As illustrated by Figure 2, rotations might also be used to better differentiate windows. In that case, similar orientations could be used to indicate that two windows are related in some way (e.g. they belong to the same application or they refer to the same document). Rotations of individual objects also make it possible to create interfaces for horizontal displays, which are particularly interesting for single-display groupware situations [25].

Alpha blending

Alpha blending allows to easily create translucent objects. As illustrated by Figure 2, we have started experimenting with the use of translucency for window contents and decorations. The least we can say after these initial tests is that it is not clear what windows should be made transparent, why and for how long. Obviously, the interesting property of a translucent object is that one can see through it. Thus, translucency should be valuable when one wants to see something behind the current object of interest. This suggests that translucency might be better thought of as a time-limited interaction technique rather than a timeless property of an object. This, in turn, might explain why ten years after the publication of the first paper describing them [4], toolglasses and magic lenses are still the best examples of use of alpha blending in graphical interfaces.

[3] check http://www.useit.com/alertbox/981115.html, http://slashdot.org/article.pl?sid=99/11/03/0917216 or http://nooface.com/search.pl?topic=visualui for some examples of these discussions

Alpha blending also poses a number of pragmatic problems. Primitives drawn using it need to be drawn after all opaque primitives are drawn. Moreover, unless the translucent objects are sorted in back-to-front order, depth buffer updates must be disabled, although depth buffer compares should remain enabled. Maintaining this back-to-front sorted list can be quite expensive if many geometric transformations are applied on the objects.

Texture mapping, lighting and image processing

Figure 4 illustrates how texture mapping can be used to transform window shapes. More complex transformations could be easily implemented in Ametista. As an example, one could create a window renderer that would apply a fisheye deformation on the window's content. One could also implement a renderer that would display only a part of the content that would have been selected interactively (the circular crop shown in Figure 4 is computed automatically).

The current implementation of Ametista does not make any use of lighting. Yet, material lighting and shading could be used, for example, to highlight the window having the keyboard focus. Similarly, image processing techniques could be used to render some windows out of focus to get a sense of depth of field. Full screen antialiasing or motion blur could also be implemented through these techniques.

The recent introduction of programmable shaders in Direct3D and OpenGL has made visual quality of interactive computer graphics take a quantum leap towards realism. We anticipate that these shaders will help us create new rendering transformations and filters for Ametista in the future.

CONCLUSIONS AND FUTURE WORK

We have presented Ametista, a mini-toolkit for exploring new window management techniques. We have described how this toolkit supports both the low-fidelity prototyping of these techniques using pseudo-windows and placeholders as well as a high-fidelity approach based on X Window application redirection. Preliminary results are encouraging: we have been able to use Ametista to experiment with several rendering styles and interaction techniques with excellent performance.

Future work on Ametista includes a better tiling algorithm for the XVNC window manager and more scripting capabilities. We plan to use the toolkit to implement and evaluate some of the layout algorithms, interaction techniques and metaphors contributed by the HCI community. Of course, we also plan to use it to explore some of the directions we mentioned in the previous section.

AVAILABILITY

Ametista and videoSpace are available in source code from

http://www.lri.fr/~roussel/software/

Several short videos of Ametista are also available from

http://www.lri.fr/~roussel/projects/ametista/

ACKNOWLEDGEMENTS

The author thanks Wendy Mackay, Michel Beaudouin-Lafon, Olivier Beaudoux, Renaud Blanch and Stéphane Conversy for providing helpful comments on an earlier version of this document.

REFERENCES

1. G. Badros, J. Nichols, and A. Borning. *Scwm - an intelligent constraint-enabled window manager*. In proceedings of AAAI Spring Symposium on Smart Graphics. IEEE Computer Society Press, March 2000.
2. M. Beaudouin-Lafon. *Novel interaction techniques for overlapping windows*. In Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2001, Orlando (USA), pages 153-154. ACM Press, November 2001.
3. M. Beaudouin-Lafon and H.M. Lassen. *The architecture and implementation of cpn2000, a post-wimp graphical application*. In Proceedings of ACM Symposium on User Interface Software and Technology, UIST 2000, San Diego (USA), pages 181-190. ACM Press, November 2000.
4. E. Bier, M. Stone, K. Pier, W. Buxton, and T. De Rose. *Toolglass and magic lenses: the see-through interface*. In Proceedings of ACM SIGGRAPH 1993, pages 73-80. ACM Press, 1993.
5. S.A. Bly and J.K. Rosenberg. *A comparison of tiled and overlapping windows*. In Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems, pages 101-106. ACM Press, 1986.
6. A. Cockburn and B. McKenzie. *Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments*. *CHI letters*, 4(1):203-210, April 2002. Proceedings of ACM CHI 2002 Conference on Human Factors in Computing Systems, Minneapolis.
7. L. Denoue, L. Nelson, and E. Churchill. *Attractive windows: Dynamic windows for digital bulletin boards*. Conference companion, Proceedings of ACM CHI 2003 Conference on Human Factors in Computing Systems, to be published (2 pages).
8. N. Elmqvist. *3Dwm: Three-Dimensional User Interfaces Using Fast Constructive Solid Geometry*. Master's thesis, Chalmers University of Technology, Göteborg, 2001.
9. S. Feiner, B. MacIntyre, M. Haupt, and E. Solomon. *Windows on the world: 2d windows for 3d augmented reality*. In Proceedings of ACM Symposium on User Interface Software and Technology, UIST '93, Atlanta (USA), pages 145-155. ACM Press, November 1993.
10. J. Gettys and K. Packard. *The X Resize and Rotate Extension - RandR*. In proceedings of USENIX Annual Technical Conference, FREENIX Track, pages 235-243. USENIX association, 2001.
11. P. Graffagnino. *Apple OpenGL and Quartz Extreme*. Presentation at SIGGRAPH 2002, OpenGL BOF.

12. J. Johnson, T.L. Roberts, W. Verplank, D.C. Smith, C. Irby, M. Beard, and K. Mackey. *The Xerox Star: a retrospective*. IEEE Computer, 22(9):11-29, September 1989.

13. E. Kandogan and B. Shneiderman. *Elastic Windows: evaluation of multi-window operations*. In Proceedings of ACM CHI'97 Conference on Human Factors in Computing Systems, Atlanta, pages 250-257. ACM Press, March 1997.

14. R. Mander, G. Salomon, and Y.-Y. Wong. *A pile metaphor for supporting casual organization of information*. In Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems, pages 627-634. ACM Press, 1992.

15. C. McCartney. *Windows Desktop Composition*. Presentation at WinHEC 2002, Windows Graphics Architecture session.

16. B.A. Myers. *A taxonomy of window manager user interfaces*. IEEE Computer Graphics and Applications, 8(5):65-84, sept/oct 1988.

17. K. Packard. *Design and Implementation of the X Rendering Extension*. In Proceedings of USENIX Annual Technical Conference, FREENIX Track, pages 213-224. USENIX association, 2001.

18. K. Perlin and D. Fox. *Pad: An alternative approach to the computer interface*. In Proc. of ACM SIGGRAPH 1993, pages 57-64. ACM Press, 1993.

19. T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. *Virtual Network Computing*. IEEE Internet Computing, 2(1):33-38, Jan-Feb 1998.

20. G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. *The Task Gallery: a 3D window manager*. In Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems, pages 494-501. ACM Press, April 2000.

21. N. Roussel. *Exploring new uses of video with videoSpace*. In R. Little and L Nigay, editors, Proceedings of EHCI'01, the 8th IFIP International Conference on Engineering for Human-Computer Interaction, volume 2254 of Lecture Notes in Computer Science, pages 73-90. Springer, 2001.

22. R.W. Scheifler and J. Gettys. *The X Window system*. ACM Transactions on Graphics, 5(2):79-109, 1986.

23. H. Shiozawa, K. Okada, and Y. Matsushita. *Perspective layered visualization of collaborative workspaces*. In Proceedings of the international ACM SIGGROUP conference on supporting group work, pages 71-80. ACM Press, November 1999.

24. M. van Dantzich, G. Robertson, and V. Ghorokhovsky. *Application Redirection: Hosting Windows Applications in 3D*. In Proceedings of NPIV99, the workshop on New Paradigms on Information Visualization and Manipulation, pages 87-91. ACM Press, 1999.

25. F. Vernier, N. Lesh, and C. Shen. *Visualization techniques for circular tabletop interfaces*. In Proceedings of AVI'2002, Trento, Italy, pages 257-263, May 2002.