Elements of a Three-dimensional Graphical User Interface

Geoff Leach, Ghassan Al-Qaimari, Mark Grieve, Noel Jinks, Cameron McKay

Department of Computer Science, RMIT, Melbourne, Vic., 3000, Australia
{gl|ghassan}@cs.rmit.edu.au

ABSTRACT The graphical user interface (GUI) is now firmly established as the preferred user interface for end users in most situations. Just as decreasing hardware prices and increasing hardware capabilities made two-and-a-half dimensional ($2\frac{1}{2}$D) GUIs affordable in the early eighties and widespread in the ninetees, we believe declining hardware prices and increasing hardware capabilities will make three-dimensional (3D) GUIs possible and affordable in the near future. Three-dimensional GUIs raise many issues of design, metaphor and usability. In this paper we discuss elements of a prototype 3D GUI we are developing.

KEYWORDS 3D graphical user interface, 3D window manager, 3D cursor, prototyping, usability.

1. Introduction

The GUI was developed at Xerox PARC in the late seventies for the Star system (Shneiderman, 1992, Smith et al., 1982). It was first successfully commercialised by Apple with the Macintosh computer in the early eighties and has since become an integral part of every modern operating system for personal computers and graphics workstations. One reason for this growth is productivity: a number of studies have shown GUIs, with their direct manipulation style of interaction, enhance productivity (Rauterberg, 1992, Margono and Shneiderman, 1987). Another reason is subjective preference: people express a preference for GUI interfaces. Coupled with the rise of the GUI has been a general elevation of the importance of the user interface, which is now recognised as a key, and sometimes central, component of an interactive product or environment.

Today's GUI depended on declining hardware prices to make it affordable at the low-end of the computer market — personal computers. The same trend, a doubling of performance approximately every 18 months to two years (widely known as Moore's Law), now means personal computers are capable of performing interactive 3D graphics. In tandem with improving hardware performance, graphics libraries such as OpenGL (Neider et al., 1993) which were previously available only on workstations have now become available for personal computers.

We believe that **3D GUIs** — graphical user interfaces which utilise 3D graphics — offer significant potential for improvement over today's $2\frac{1}{2}$D GUIs, and furthermore, that it is now, or very shortly will be, possible to run such interfaces on personal computers. In this paper, we discuss elements of a prototype 3D GUI we are developing and discuss some early feedback from usability tests.

2. Background

The researchers and developers at Xerox PARC who produced the Star system were determined to produce a system far easier and more intuitive to use than the command-line interfaces which predominated at the time (Smith et al., 1982). A key part of achieving this was providing a metaphor to integrate the visual elements of a graphical user interface into a recognisable and comprehensible framework. In the **desktop metaphor** used for the Star, and for most commercial GUIs since, the interface is conceptualised and presented as a graphical version of a typical office. Its key elements are:

- **A desktop** or space on which **icons** or small pictographs which correspond to commonly found equipment in offices (bins, printers, documents and folders) are placed.

*A copy of the paper with colour figures is available at http://www.cs.rmit.edu.au/~gl/research/HCC/interact97.html

- Overlapping rectangular **windows** which house applications and documents which correspond to working sheets of paper.
- **Folders** for storing other documents and folders.
- **Pull-down menus** to allow the user to make choices and provide instructions (which have no direct office analogue).

An example of the desktop metaphor GUI is shown in figure 1.

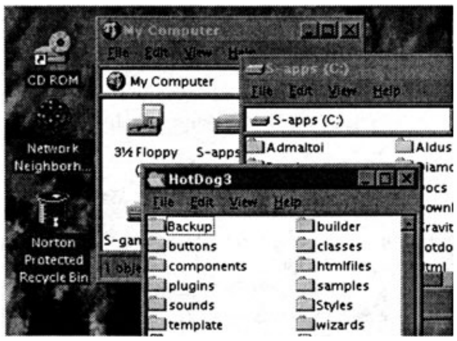


Figure 1: Desktop metaphor (Windows 95)

Another key aspect of the GUI, although not strictly part of the desktop metaphor, is the use of a pointing device, typically a mouse (Johnson et al., 1989, Shneiderman, 1992), to control the position of a 2D **cursor** on the screen. Operations may be invoked with the mouse in a variety of ways, with the user being provided with continuous visual feedback. For example, a **copy** operation may be invoked by **selecting** an object's icon with the cursor and then **dragging** it to another location, with the exact semantics of the copy operation depending on the selected object and the source and destination positions. For example, if the selected object is a document, the source position is a folder on one disk and the destination position is a second disk then a copy of the document is made on the second disk. The user is provided with continuous feedback through smooth animation of the movement of the document icon whilst it is being dragged from source to destination. In some cases, further feedback on the progress of the copy operation is provided by another visual element constantly displaying what proportion of the document has been copied.

The desktop metaphor GUI is described as a **direct manipulation** interface. Direct manipulation is a term introduced by Ben Shneiderman in 1982 (Shneiderman, 1982, Shneiderman, 1983) to describe interfaces exhibiting the following characteristics:

- Continuous visibility of objects of interest. For example, an object stays where it is until the operator moves it.
- Rapid, incremental reversible operations whose impact is immediately visible.
- Pointing, selecting and dragging techniques replacing the need to type complex syntax, such as command lines like MS-DOS or shell interfaces.

The copy operation described above, for example, exhibits these characteristics.

Advocates of direct manipulation maintain that systems designed with these principles in mind are easier to use than command-line oriented systems, because they allow the creation of a prefered mental model and because they reduce the mental workload by allowing recognition rather than recall (Hix and Hartson, 1993). For a person to complete a task, they must transform their mental goals ("I want to write a document") into physical ones ("Move the mouse to the word processor icon, and double-click to open it"). The process of constructing a task-action mental model (Young, 1983), visualising the desired outcome, and mapping this to physical actions can be measured by two gulfs: the gulf of execution and the gulf of evaluation (Hutchins et al., 1986). The gulf of execution refers to the distance between the goals of the user and the means of achieving them through the system, and the gulf of evaluation refers to the distance between the system's behaviour and the user's goals. These gulfs are reduced in GUIs where by directly manipulating objects and data visually users are given a sense of direct engagement and of control, both of which facilitate understanding of the nature of tasks (Johnson et al., 1989, Shneiderman, 1992, Carroll, 1987).

3. The 3D GUI

The desktop metaphor GUI is **$2\frac{1}{2}$D**. It is **2D** because its visual elements are two-dimensional: they lie in the xy plane, are defined in 2D coordinates, are flat and contain only planar regions (areas). It is **$2\frac{1}{2}$D** because where visual elements overlap they obscure each other according to their **priority**. In a 3D GUI the visual elements are genuinely three-dimensional: they are situated in xyz space, are defined in terms of 3D coordinates, need not be flat and may contain spatial regions (volumes).

The design considerations for a 3D GUI appear more complex than for a $2\frac{1}{2}$D GUI. To begin with, the issues of metaphor and elements arise afresh. The desktop metaphor with its windows, icons, menus and pointing device elements is firmly established for $2\frac{1}{2}$D GUIs. In contrast no clearly defined metaphor and set of elements for 3D GUIs are manifest — yet. 3D GUIs offer considerably more scope for metaphors than $2\frac{1}{2}$D GUIs; there are many metaphors which could be based on our physical 3D environment, including the obvious extension of the desktop metaphor into a 3D office metaphor. On the other hand, much more abstract metaphors are possible, such as one based "starmaps" where objects are simply placed somewhere in "cyberspace". Likewise the elements of a 3D GUI may resemble, or differ substantially from, the elements of the $2\frac{1}{2}$D GUI.

In our prototype we identify essentially the same elements in the 3D GUI as in the $2\frac{1}{2}$D desktop GUI: windows, icons, menus, a general space in which to arrange the visual elements, a cursor and an input device to manipulate the cursor. At this stage we are not proposing radical change to the GUI, in so far as we regard provision of an appropriate metaphor and the use of familiar elements as important design criteria — our 3D GUI is clearly recognisable as a GUI. It will be seen, however, that some of these elements are quite different from their $2\frac{1}{2}$D counterparts.

There are two main thrusts to our prototyping work. The first is a 3D window manager, which we call MaW^3. Currently it is a simulator rather than a working window system in that windows do not contain editable documents. The second thrust is the use of a 3D cursor, which we call the Magic Wand, controlled by a 3D (six degrees of freedom) input device (a Spacetec spaceball (Spacetec, 1996)).

3.1 Related Work

Today's desktop metaphor GUIs are largely unchanged from those of twenty years ago, which is testament to the strength of the original design. There have been, of course, efforts to improve it. Of specific relevance to our work are efforts utilising 3D graphics. Work in this area tends to be focussed on experimental systems directed at a particular application or application area — as is our system. The Information Visualizer developed at Xerox PARC is an experimental system which explores a 3D user interface paradigm suitable for applications to manipulate large amounts of information (Robertson et al., 1993). At Apple, as part of the meta content format (MCF) project (Apple, 1996), a 3D MCF browser called HotSauce has been developed. It allows the user to explore a hierarchical (branching) structure in 3D space. Another example is the WebForager for exploring the world wide web (Card et al., 1996). In the WebForager 3D WebBooks containing collections of web pages, and other objects, are placed in a 3D workspace.

An innovative 2D approach to the user interface is proposed in the Pad system (Perlin and Fox, 1993). There the workspace is an infinite 2D plane on which information can be placed at arbitrary places with arbitrary scale. The user can repeatedly zoom in on particular regions through **portals** which act as magnifying glasses. Our system shares some of these characteristics, although it has only finite scaling, rather than infinite scaling.

3.2 MaW^3: A 3D Window Manager

A task which occurs in the $2\frac{1}{2}$D desktop GUI is window management. Windows house applications and documents, and may overlap and obscure each other. A natural mode of human operation is **multitasking**, that is, to have a number of tasks or jobs underway concurrently. A consequence of this, and the small screen size (up to 21" maximum typically) dictated by today's CRT and LCD display technologies, is the need to manage the position and layout of windows. The **window manager** performs this task in response to user input.

A situation which arises in the use of windowing environments is the need to constantly arrange and rearrange windows to get access to the particular window which houses the task or information one needs at a given point in time — which has been described as **window thrashing** (Henderson and Card, 1986). A number of techniques have been proposed to facilitate efficient display usage and application switching to reduce or overcome window thrasing, including tiled layout, "virtual" desktops (Hines, 1996, Harch, 1996) and fisheye views (Furness, 1986). However, most of these approaches adhere to the $2\frac{1}{2}$D desktop model and whilst helping considerably still suffer residual problems: multiple overlapping windows are hard to identify, iconified windows more so, and one quickly runs out of space trying to group related applications. Alleviating the problem of window thrashing is one of the main design goals of our 3D window manager.

The metaphor we have used for the 3D space in which we arrange windows is that of a **tunnel**. The user is positioned in the middle of the mouth of the tunnel looking toward the other end. The tunnel, and windows in it, are displayed with a perspective projection, as shown in figure 2. Windows are essentially 2D, in that their "work area" is 2D, although they have 3D frames, decoration and buttons. At this stage we do not plan on changing the structure of documents, which are essentially 2D constructs, although there may be contexts in which some kind of 3D presentation may be advantageous — outlining for example.

Windows may be positioned at arbitrary depths in the

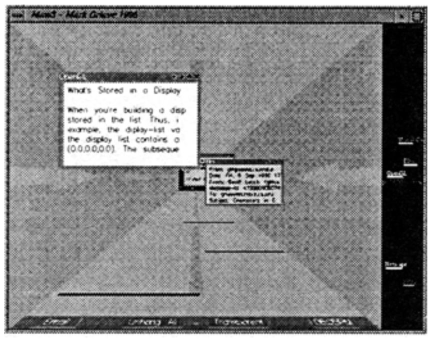


Figure 2: Tunnel

tunnel but are restricted in their orientation. The normal orientation is orthogonal to the longitudinal axis of the tunnel, or more simply, "front-on". As a user pushes a window further into the tunnel its size is diminished in the normal inverse size to distance relationship of perspective projection. Window positions are constrained so that at least a portion of a window always remains inside the tunnel, although, for example, a window can be moved downwards so that only its top part, or status bar, is visible.

In addition to the front-on window display mode there is a "hanging" mode where the windows are hung on the left or right walls of the tunnel, as shown in figure 3. Individual windows may be hung or all the windows may be hung at once. Hanging all the windows allows the user to quickly gain an overall idea of where the windows are in the tunnel.

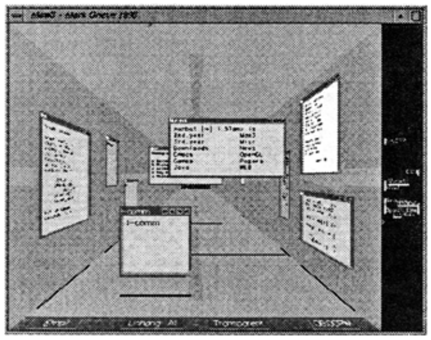


Figure 3: Window hanging

There are two further major components of our 3D window manager. The first is an **overview** area to the right of the tunnel which may be seen in figures 2 and 3. The overview area provides a plan or top view of the tunnel and is provided in addition to the main "down the tunnel" view. Windows may be selected and then moved up and down in the overview area, corresponding to back and forward in the tunnel. Windows' names are displayed in the overview area to aid identification of specific windows, although, due to the small font the names are hard to see in the diagrams.

The last major component of our window manager is a console, positioned at the bottom of the screen as shown in figures 2 and 3. Controls are provided on the console for changing global settings affecting windows. There is a button to hang all the windows at once, rather than hanging invididual windows. There is another button to toggle transparency: windows can be rendered partially see-through, so that windows may be seen through other windows. There are also mechanisms to control lighting, which provides enhanced visual cues about the position of windows in the tunnel, and options to set different rendering modes — solid, wireframe and invisible — for the walls to allow the user to customise their appearance. The console is attached to the user's viewpoint so that it moves through the tunnel with the user. Finally, a clock is provided on the console as an example of the more general functionality we envisage a working model to incorporate.

We now return to the issue of window thrashing. Our 3D window manager addresses it in four main ways:

1. Through window hanging. Hanging some or all of the windows allows the user to obtain a global view of window locations in a natural and easily invoked manner. It also allows better allocation of the available screen real estate.

2. Through the scaling of window size in inverse proportion to distance down the tunnel. Windows which are not in use may be pushed back down the tunnel where they will be small but visible.

3. By reducing mouse movement required to access windows. The centralising of windows in the screen via the tunnel, combined with the inverse size versus distance relationship, appears to reduce mouse movement required to access windows compared to the approaches to window thrashing adopted for $2\frac{1}{2}$D desktop GUIs (see section 5).

4. Through the use of transparency. If windows are made transparent, obscured windows are able to be seen (although selection becomes difficult).

3.3 The Magic Wand: A 3D Cursor

The second thrust of our 3D GUI prototyping work is exploring the design and usability of a 3D cursor controlled by a 3D input device. One of our 3D cursors is shown (at close range) in figure 4 where it is being used to select a chess piece.

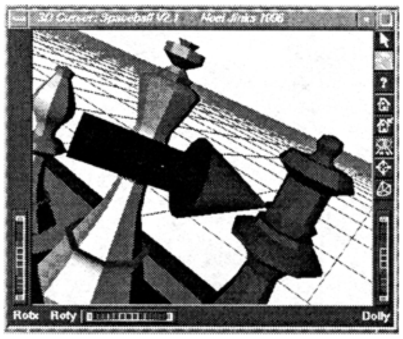
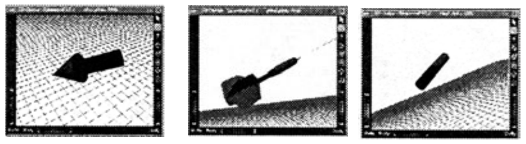


Figure 4: 3D cursor

Most interactive 3D graphics applications employ a 2D cursor and 2D pointing device for interaction. Given that the user is manipulating 3D objects or worlds in these applications, this gives rise to a fundamental mismatch in dimensionality. In the context of a 3D application the cursor "floats" over the top of the objects rather than being part of the scene. We believe the use of a 3D cursor, introduced into the scene as an object in its own right, controlled by a 3D (six-degree-of-freedom input) device is a better approach for interactive 3D applications, including a 3D user interface, than the traditional approach. Recently, work in the area of 3D interaction has intensified (Bier, 1990, Herndon et al., 1994, Houde, 1992). We believe this to be a significant area for research in its own right, with the potential for many important practical outcomes.

In the $2\frac{1}{2}$D GUI there are different cursors — arrow, hourglass, cross-hair, etc. — for different contexts. Likewise we believe a 3D cursor should take on different characteristics and behaviours, to allow a better relationship to the particular type of interaction task being performed. However, we believe these cursor changes are better applied if the user is given more control over them. The user can be made aware, or reminded, of the particular characteristics and behaviours of a given cursor by the visual characteristics of the cursor — adhering to the principles of direct manipulation. By putting more control of the cursor in the user's hands the 3D cursor becomes a far more sophisticated tool than its 2D counterpart — a sort of "magic wand".

To provide a framework for comprehension of the different characteristics and behaviours of our 3D cursor we use different metaphors. These are shown in figure 5. The user may switch between the type of cursor displayed by pressing a button on the spaceball. The cursors are intended for different types of interaction:



(a) Arrow (b) Laser (c) Bar magnet

Figure 5: Cursor metaphors

- The arrow cursor is our general purpose cursor and is the 3D equivalent of the 2D arrow. It must be in contact with objects to select them. Once selected objects may be dragged.
- The laser cursor is used for precision selection of objects. It sends out a "light ray" which can be swept over objects. As objects fall in the path of the light ray they are highlighted to give the user continuous feedback consistent with the principles of direct manipulation. Unlike the general arrow cursor the laser cursor may select objects from a distance. When a particular object of interest is highlighted it may be selected by clicking a button. Once selected, objects may be dragged or **tractored**, that is, moved along the light ray's path towards (or away from) the cursor. At this stage the light ray does not penetrate objects, and thus only the first object intersected is highlighted. An example of the laser cursor being used to select an object is given in figure 7.
- The bar magnet cursor is used to select groups of objects within a certain distance of the cursor, that is, within its magnetic field. It is intended for interaction tasks where less accurate but faster gross selection is required rather than precise selection and tasks where multiple objects are to be selected and operated on at once.

3.4 The Spaceball: A 3D Input Device

In order to effectively control the 3D cursor we use a spaceball (Spacetec, 1996), as shown in figure 6. The

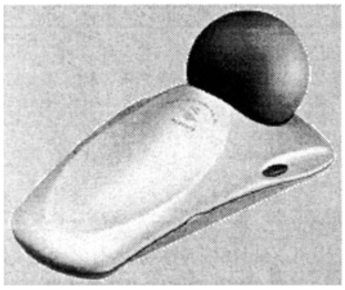


Figure 6: Spaceball

spaceball is a 3D input device with six-degrees-of-freedom to match those of the cursor. It consists of a **powersensor**, which is a ball slightly smaller than a tennis ball, mounted on a platform that rests on the top of the desk. Pushing, pulling and twisting actions on the powerball are mapped onto translations and rotations of the 3D cursor. For instance, by pushing forward on the powersensor whilst similtaneously twisting it around the vertical axis, the user is able to direct the cursor into the scene, (increase its z value) whilst similtaneously rotating it about the y axis.

4. Implementation

For our prototyping work we have used two 3D graphics libraries: OpenGL and Open Inventor. OpenGL (Neider et al., 1993) is a low level graphics library originally developed by Silicon Graphics but which is now available for most UNIX graphics workstations and personal computer operating systems. Open Inventor (Wernecke, 1994) is a higher level, object-oriented, 3D graphics, C++ class library from Silicon Graphics.

The 3D window manager is implemented in C and uses OpenGL. The 3D cursor is implemented in C++ and uses Open Inventor. We have used both the OpenGL and Open Inventor libraries to explore their suitability for our 3D GUI project. OpenGL, with its lower level facilities tends to give the programmer greater control and better performance, and its wide availability is obviously important. On the other hand, the object oriented approach of Open Inventor offers advantages for prototyping work.

Implementation of a working 3D GUI is a considerably larger and more complicated task than the building of a prototype. For instance, implementing "live" windows in a 3D GUI requires an inversion of the current GUI architecture in which 3D graphics are displayed in 2D windows in a $2\frac{1}{2}$D environment — and it appears that considerable low level programming is required to achieve this. Another major issue is performance. We have found that our 3D window manager and 3D cursor perform quite well on an Indy workstation and that the 3D window managers performs satisfactorily on a pentium based personal computer running Windows 95 (without a 3D graphics accelerator board). There are undoubtedly specialised approaches to improve the performance of both the 3D window manager and the 3D cursor, rather than using the traditional brute force redraw approach of 3D graphics (which we use), however, the performance of today's low-end architectures already appears to be adequate, or nearly adequate, to support at least simple 3D GUIs.

5. Usability Testing

The star life cycle approach to system development (Hix and Hartson, 1993) stresses rapid prototyping and incremental development of interactive systems. The star model is highly iterative and flexible enough to allow development to commence from the prototyping stage when the design specifications are not clear in the mind of the developer. This can be advantageous when novel design ideas are considered as it allows developers to better understand the limitations of technology and how these ideas should be implemented in order to match users needs. Advances in technology make possible imaginative and innovate approaches to both existing and new problems. Rapid prototyping offers designers an opportunity to test their novel ideas with representative users, and to set realistic requirements before committing time and effort to implement them.

In the case of our 3D cursor and 3D window manager prototypes, we conducted exploratory evaluations (Nielsen, 1993, Rubin, 1994) to help us test our high-level concepts, and to solicit users' ideas about how to improve confusing areas. Our emphasis during the exploratory test was on understanding users' mental models and why they perform as they do. Later, after implementing the prototypes, we conducted another assessment evaluation for the purpose of determining how effectively the high-level concepts were implemented. Rather than just exploring the intuitiveness of our 3D cursor and 3D window manager, we were interested in testing how well users can actually perform realistic tasks, and in identifying usability deficiencies. Some of the issues we aimed to investigate in our exploratory evaluation with the help of our representative users are:

- Subjective satisfaction with both the 3D window manager and the 3D cursor.
- Ease of navigation in the tunnel and how efficiently

windows could be managed.

- Whether users are cognizant that the 3D cursor is an object in its own right in the environment.
- The intuitiveness of the idea of associating different behaviours (metaphors) to the 3D cursor.
- Whether a 3D cursor controlled by the spaceball is a good approach for 3D interaction.

The results of our evaluation were very encouraging. A key result is users' subjective satisfaction where we found the attitude toward using the 3D window manager, the 3D cursor and the spaceball to be enthusiastic, to an extent which was surprising. We also sought quantative results, although space constraints preclude their detailed reporting here. An example of one of the tasks users performed as part of our usability testing is shown in figure 7. Here

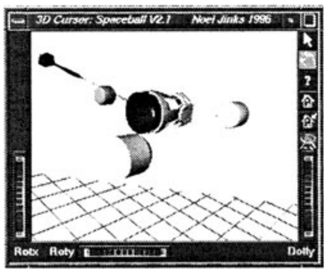


Figure 7: 3D interaction task

users were required to select the components of the object (an exhaust system for a jet engine) one-by-one and pull the object apart using the laser cursor. Users' performance in this task, and others, clearly indicated that: they were cognizant of the fact that the 3D cursor is an object in its own right in the scene, that assigning effective metaphors to the cursor improved its usability when performing selection, highlighting and movement operations, and that using a 3D pointing device reinforces the feeling of a direct engagement between the user and the environment. We also measured the speed of performance and the number of errors made: this task was one the users were able to complete more effectively using the 3D (laser) cursor and spaceball than using the 2D cursor and a mouse.

One of our aims with the 3D window manager was to reduce the amount of mouse movement required to find windows — as part of the goal of addressing the problem of window thrashing. The evaluations indicate that the 3D tunnel may facilitate better management of screen space. We have implemented a routine which tracks mouse movements in an effort to try and quantify the mouse movements performed. An example trace is shown in figure 8, which shows that most of the mouse movement occurs in the centre of the screen with occasional moves to the bottom and right of the screen. The trace was produced by performing operations a user would typically engage in: moving, grouping and hanging windows. As yet we have not performed comparisons based on this metric between our 3D window manager and any $2\frac{1}{2}$D window manager.

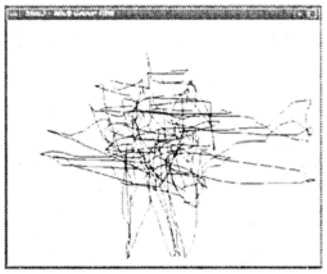


Figure 8: Mouse trace

We intend to conduct more rigorous evaluations to obtain statistically significant results following completion of a working 3D window manager. Further experiments are also needed to examine the issue of alternating between 2D and 3D cursors and 2D and 3D input devices.

6. Conclusions and Future Work

The elements of the 3D GUI prototype we have discussed have been well received by a sample group of users. Obvious further work is to integrate the 3D window manager and the 3D cursor in a working windowing system, and to provide other elements of a 3D GUI. There is also the issue of providing both 2D and 3D cursors and using both 2D and 3D input devices. For tasks which are inherently 2D, a 2D cursor controlled by a 2D input device is superior, whilst, we believe, for tasks which are inherently 3D, a 3D cursor controlled by a 3D input device is superior. Beyond these aspects there is the potential for incorporating intelligence into the interface and the cursor.

References

Apple (1996). Hotsauce. http://mcf.research.apple.com.

Bier, E. (1990). Snap-dragging in three dimensions. In

The Proceedings of the Symposium on Interactive 3D Graphics.

Card, S. K., Robertson, G. G., and York, W. (1996). The webbook and the web forager: An information workspace for the world-wide web. In *CHI 96*, pages 111–117.

Carroll, J. (1987). *Interfacing Thought.* MIT Press.

Furness, G. (1986). Generalized fisheye views. In *CHI '86 Conference on Human Factors in Computing Systems*, pages 16–23.

Harch, G. H. (1996). Impvwm 32bit virtual window manager for windows 95. http://www.kent.edu/ gharth.

Henderson, D. A. and Card, S. K. (1986). Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5(3):211–243.

Herndon, K., van Dam, A., and Gleicher, M. (1994). The challenges of 3d interaction. *SIGCHI Bulletin*, 26(4):36–39.

Hines, C. (1996). Fvwm window manager. http://www3.hmc.edu/ tkelly/docs/proj/fvwm.html.

Hix, D. and Hartson, H. R. (1993). *Developing User Interfaces.* John Wiley and Sons.

Houde, S. (1992). Iterative design of an interface for easy 3d direct manipulation. In *The Proceedings of CHI '92 Workshop.*

Hutchins, E., Hollan, J. D., and Norman, D. A. (1986). Direct manipulation interfaces. In Norman, D. A. and Draper, S. W., editors, *User centered system design:New perspectives on human-computer interaction.* Erlbaum Associates.

Johnson, J., Roberts, T., Verplank, W., Smith, D., Irby, C., Beard, M., and Mackey, K. (1989). The xerox star: A retrospective. *Computer*, pages 11–26.

Margono, S. and Shneiderman, B. (1987). A study of file manipulation by novices using commands vs. direct manipulation. In *Proc. ACM D.C. Chapter 6th Annual Technical Symposium.*

Neider, J., Davis, T., and Woo, M. (1993). *OpenlGL Programming Guide.* Addison-Wesley.

Nielsen, J. (1993). *Usability Engineering.* Academic Press.

Perlin, K. and Fox, D. (1993). Pad: An alternative approach to the computer interface. In *SIGGRAPH Computer Graphics*, pages 57–64.

Rauterberg, M. (1992). An empirical comparison of menu-selection (cui) and desktop (gui) computer programs carried out by beginners and experts. *Behaviour and Information Technology*, 11(4):227–236.

Robertson, G. R., Card, S. K., and Mackinlay, J. D. (1993). Information visualisation using 3d interactive animation. *Communications of the ACM*, 36(4):57–71.

Rubin, J. (1994). *Handbook of Usability Testing.* Wiley.

Shneiderman, B. (1982). The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology*, 1(3):237–256.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69.

Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction, 2nd ed.* Addison-Wesley.

Smith, D., Irby, C., Kimball, R., and Verplank, B. (1982). Designing the star user interface. *Byte*, pages 242–282.

Spacetec (1996). Spacetec imc web site. http://www.spacetec.com/.

Wernecke, J. (1994). *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Addison Wesley.

Young, R. (1983). Surrogates and mappings: two kinds of conceptual models for interactive devices. In Gentner, D. and Stevens, A., editors, *Mental Models.* Lawrence Erlbaum Associates.