Toolspaces And Glances: Storing, Accessing, And Retrieving Objects In 3D Desktop Applications

Jeffrey S. Pierce[1], Matthew Conway, Maarten van Dantzich, George Robertson
Microsoft Research
jpierce@cs.cmu.edu
{mconway, maartenv, ggr}@microsoft.com

Abstract

Users of 3D desktop applications perform tasks that require accessing data storage, moving objects, and navigation. These operations are typically performed using 2D GUI elements or 3D widgets. We wish to focus on interaction with 3D widgets directly in the 3D world, rather than forcing our users to repeatedly switch contexts between 2D and 3D. However, the use of 3D widgets requires a mechanism for storing, accessing, and retrieving these widgets. In this paper we present *toolspaces* and *glances* to provide this capability for 3D widgets and other objects in interactive 3D worlds. Toolspaces are storage spaces attached to the user's virtual body; objects placed in these spaces are always accessible yet out of the user's view until needed. Users access these toolspaces to store and retrieve objects through a type of lightweight and ephemeral navigation we call glances.

CR Categories and Subject Descriptors: I.3.6 [Computer Graphics] Methodology and Techniques - Interaction Techniques.

Additional keywords: desktop virtual reality, 3D graphics, virtual worlds, two handed interaction

1 INTRODUCTION

Three operations are common across many 3D desktop applications:

- **Accessing storage**. This includes storing and retrieving data sets, tools, models, and worlds.
- **Moving objects** when the destination is not currently visible. Example operations include moving a rake widget for scientific visualization [16] and arranging objects in a scene.
- **Navigation**. Examples include architectural walkthroughs, navigation between rooms in the Information Visualizer [24], and moving around a 3D MUD or chatroom.

Users typically perform these operations by clicking on 2D GUI elements or by interacting directly in the 3D world with 3D widgets. These two approaches each have their drawbacks. 2D GUIs consume valuable screen real estate and require the user to switch back and forth between the 2D desktop and the 3D world. 3D widgets, on the other hand, beg the question of access: how do users get at the widgets, and how are they stored when not in use? Widgets just dropped into the 3D scene may clutter up the world and may not be accessible when needed.

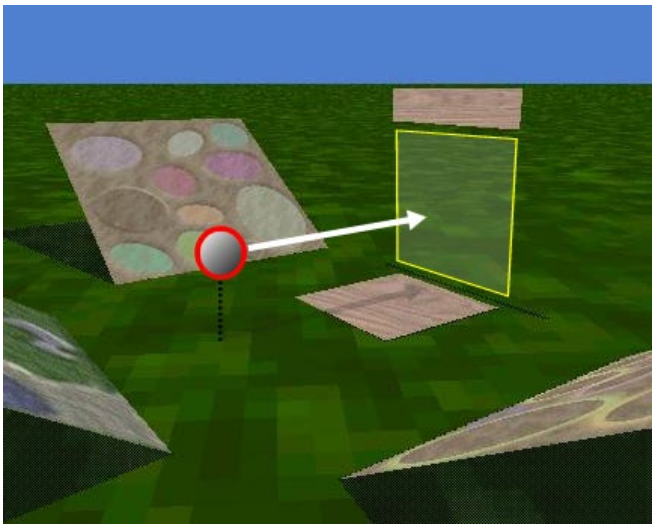


Figure 1: The user (represented by the circle) is surrounded by five toolspaces that do not enter his view (represented by the translucent square) unless explicitly accessed.

One common approach to answering the question of access in immersive virtual environments is to leverage human proprioception through body-centric storage [20]. In this approach, the user attaches objects to his body in a manner similar to hanging tools on a traditional toolbelt. Attached objects travel with the user and are always in the same place relative to the user's body, which keeps these objects within convenient reach, out of view until needed, and accessible without a great deal of cognitive attention.

The toolbelt approach encounters difficulties in desktop-based virtual environments. Even if we give the user a virtual body on the desktop, anything attached to the virtual body outside of the view frustum is inaccessible if the user has no mechanism for manipulating the viewpoint relative to the body. To address this problem we introduce glances.

1. Jeff Pierce is currently at Carnegie Mellon University.

Glances are a lightweight and ephemeral type of 3D navigation. Glances are lightweight because the user does not manually steer the camera into position; the system restricts the new viewpoint to a set of canonical views, allowing the user to initiate a glance with a simple command. The system determines the camera path by interpolating between the two viewpoints using an ease in / ease out animation style. Glances are activated and maintained through muscle tension, like holding down a key or touching the surface of a touchpad [27]. Glances are ephemeral because they do not permanently change the viewpoint; when the user ends the glance by releasing muscle tension the system restores the previous viewpoint. By combining glances and more traditional 3D navigation, we create a new metaphor for viewpoint control that includes both a virtual head and a virtual body. Navigation moves and turns the virtual head and body, while glancing turns the virtual head relative to the virtual body.

We introduce *toolspaces* to give users a mechanism for attaching objects to their virtual body. A toolspace is like a backpack or a fanny pack: a container that moves with the user's body in the 3D world. Our implementation provides the user with five toolspaces arranged around his virtual body and outside his view frustum.

These toolspaces can store any objects that are useful for the task at hand. We combine glances with toolspaces by using glances to access the toolspaces: each glance points the user's gaze into a toolspace.

2 USING TOOLSPACES AND GLANCES

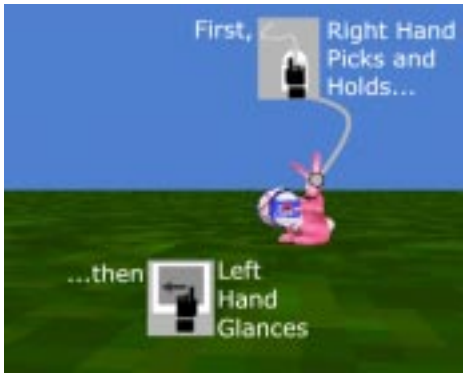
Toolspaces are a general "container" mechanism useful for facilitating the object transportation, navigation, and data storage access operations common to 3D applications. In this section we discuss several examples.

Toolspaces can be used like a backpack or toolbelt in the real world: to store and transport objects (see Figure 2). To store an object in a toolspace, the user clicks on the object to start a drag operation. With the mouse button down, the user glances in the direction of the desired toolspace (by swiping left on a touchpad, for example). The dragged object rotates with the user's head, remaining in the same relative position in his view. The user releases the object in the toolspace by releasing the mouse button.

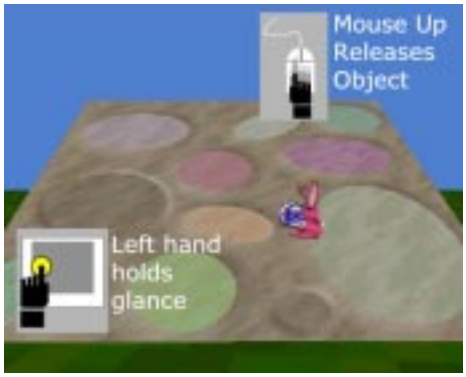
Similarly, the user retrieves objects by glancing into the toolspace, starting a drag operation, and ending the glance to rotate the object back into the world.

We can use these mechanisms to implement a form of cut-and-paste or drag-and-drop with some significant improvements: the ultimate destination of transported objects does not have to be immediately visible and multiple objects can be transported simultaneously. Unlike Fix and Float [25], which had different design goals, the transported objects do not consume any screen real estate, making it easy for the user to engage in intermediate tasks while objects are kept in storage in the toolspaces.

Toolspaces' ability to store objects has more uses than moving objects from one location to another. Users can also manipulate widgets in toolspaces without first removing the widgets from the toolspace. Toolspaces can store a widget like a World in Miniature [29], and users can work with it directly in the toolspace. The WIM occupies no screen real estate when not in use, and placing it in a toolspace slightly below the normal view frustum allows the user to glance down to access the WIM and still see part of the scene. As another example, toolspaces can store palettes containing miniature



This swings the view into the left toolspace.



... which returns the view forward.

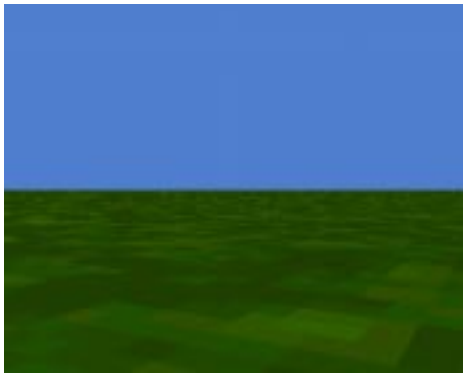


Figure 2: Dragging An Object Into a Toolspace

objects that can be added to the scene [1] [4]. Users can “tear off” copies and move them from the toolspace to the 3D world; unlike selecting the file name of an object to add, users can visually determine the object they need. A toolspace could also store a widget like a color palette. Users could paint objects in the 3D environment, and glance into the toolspace and manipulate the color palette to adjust the current paint color.

Toolspaces can facilitate navigation when used in combination with 3D thumbnails [11]. Users can create a 3D thumbnail representing their current location, place it in a toolspace, and retrieve it later to return to that location.

Toolspaces are also useful in any situation where a file, directory, or other information store might make sense. They can function as a trash can in the 3D world: the system places deleted objects in a toolspace to make them accessible if they are later needed. A toolspace can also represent persistent storage; users can place copies of objects or environments in the toolspace to save them to disk, or retrieve a copy to load it into the application.

3 IMPLEMENTATION

We implemented a system with five toolspaces to experiment with these applications of toolspaces and glances. We placed toolspaces to the left and right of the user as well as above and below the user's forward view because these are the directions people typically glance in the real world. The toolspaces to the user's left and right encompass everything to that side of the user. The toolspaces above and below the user's forward view are smaller, encompassing the volumes in front of the user that are above and below the user's normal view frustum. We placed the final toolspace behind the user. This toolspace is as large as the toolspaces to the left and right but is more difficult to access, making it useful for deep storage; the user can place infrequently needed objects and tools in this space, and move them to a different toolspace if they are needed more often.

We chose to provide the glance commands through a touchpad that users operate with the non-dominant hand. Users glance with the touchpad by sliding a finger in the direction of the desired toolspace for the left, right, top, and bottom toolspaces; to access the rear toolspace users first glance into the left or right toolspaces, and then slide a finger left or right again. Glances are spring-loaded: the new viewpoint lasts only as long as users keep a finger on the touchpad.

We could also have implemented glances using the keyboard: the arrow keys provide the most obvious mapping for accessing toolspaces. However, we found that the touchpad had advantages over the arrow keys. The touchpad is easier to acquire because it is a larger target than the arrow keys. For right-handed users the touchpad is also closer to the resting position of their left hand than the arrow keys, and doesn’t require them to continually bring their left hand across their body. The touchpad also works for more than four toolspaces.

Toolspaces can constrain object manipulation within them to make it easier to position objects in a 3D space using a 2D input device. Our left, right, and back toolspaces each use a Data Mountain [26] to constrain the layout of objects: the Data Mountain constrains objects to move in the plane of the mountain, and implements a simple “bumping” behavior for handling collisions between objects on the mountain. The bottom toolspace constrains objects to move parallel to the ground, while the top toolspace constrains objects to move in a single dimension along a shelf.

Toolspaces can also constrain the maximum size of objects within them by resizing objects; this increases their storage capacity and allows users to carry around very large objects. The left, right, and top toolspaces we built resize objects as they are dragged into the toolspace so that their longest dimension is one fifth of a meter, while the back and bottom toolspaces resize objects so that their longest dimension is one tenth of a meter. Because we want objects to be usable within toolspaces, objects override these size constraints if they would be too small to use at that size. Note that these decisions are ad-hoc; the nature of the user's task will dictate how toolspaces should behave in a given context.

4 EVALUATION

We conducted an exploratory user study with six users to observe how people use toolspaces and glances when given a scene assembly task. We placed three object palettes in the scene (each containing miniatures representing objects that could be added to the scene), as well as two markers located at different positions away from the palettes. We instructed our users to construct a scene near the first marker, using at least one object from each palette. When they finished, we told them to move their scene to the second marker and add additional objects.

Our users made heavy use of toolspaces when moving objects around the scene and when they need to access widgets from different locations. Although in general users had no problems with the interface, based on this study we believe that a discrete touch in a specific region of the touchpad is probably preferable to a swiping gesture for initiating glances: some users had never used a touchpad before and had some initial difficulty swiping. A simple raised grid could be placed over the touchpad to provide haptic feedback for these regions.

We learned that people do represent the positions of the toolspaces egocentrically; users physically pointed left and right when asked about the location of objects in toolspaces, one user spoke about “putting objects to my left”, and another referred to an object that was “behind me”. However, we found that users were not too tied to the metaphor of turning their heads to access toolspaces: a few users wanted to be able to turn their view in a full circle to glance into the left, back, and right toolspaces and then return their view forward.

5 DESIGN CONSIDERATIONS

We have presented one possible implementation of toolspaces and glances. This is not the only possible implementation; designers have a number of choices available when adding toolspaces and glances to a 3D world. These choices include:

How many toolspaces exist? Adding more toolspaces increases the storage space available to the user, but also increases the difficulty of remembering which toolspace an object is in. Adding more toolspaces also requires adding more glances, which can make it more difficult to access a toolspace.

How are the toolspaces arranged? One of the key advantages of combining toolspaces and glances is that designers can place the toolspaces outside the user's normal view frustum. This puts a large collection of objects and tools at the user's fingertips, but keeps them out of sight until needed. Since these toolspaces are normally out of view, we believe that users have an easier time remembering the locations of objects in toolspaces if they can draw on real world metaphors (e.g. backpacks) and relate the position of toolspaces to their body. We note, however, that the layout of toolspaces does not have to be constrained by the real world. For instance, multiple

toolspaces could occupy the same space. Consider two toolspaces attached to the user's back: glancing behind to the left may access one, and glancing behind to the right the other. Toolspaces could also be arranged so that glances translate as well as rotate the user's view. For instance, accessing toolspaces might resemble ascending the ramp of a parking garage. Naturally more exotic layouts will make it more difficult for users to keep track of objects in toolspaces.

Another possibility for toolspace placement (suggested by one of our test users) is to place a toolspace in the view frustum but make it invisible. This toolspace could fade in when needed, and fade out when not being accessed. This idea is similar to the fade-in menus Deering implemented for HoloSketch [10].

How does the user access toolspaces? Glances that rotate the user's view to look into a toolspace are one possibility. This approach mimics the user turning his head in the real world. Alternately, the user might rotate a toolspace around his body to bring it sight: instead of looking over his shoulder to find an object in his backpack, the user removes the backpack and swings it around in front to find the object.

Are glances spring-loaded? Glances may be implemented so that the viewpoint resets automatically when the user releases a key or removes his finger from a touchpad, or the user may need to explicitly restore the previous viewpoint. There are advantages to each approach. Spring-loaded glances save the user a step, and maintaining a key press or touch is a physical reminder that the user is engaged in a glance. By contrast, requiring the user to explicitly restore the viewpoint allows the user to use both hands and perform longer tasks while engaged in a glance. Two users in our study did express the desire to be able to temporarily override the spring-loaded glances and lock their view during extended interaction in a toolspace so that they didn't have to actively maintain the glance.

How are the contents of toolspaces constrained? All toolspaces constrain their contents to maintain a fixed position relative to the user's body. Toolspaces may impose additional constraints to make them easier to use. They may resize objects to make them occupy less space, allowing more objects to be stored in each toolspace. They may also impose layout constraints (such as collision detection, constraining objects to move in a plane, or pigeon holing) that make it easier to store and find objects.

6 RELATED WORK

Both Conner and Herndon presented 3D widgets for use in desktop virtual environments [8] [16]. However, they did not discuss how to manage and transport the widgets in the 3D world. Mine proposed taking advantage of proprioception as one solution to this problem and coined the term physical mnemonics to refer to the storage of virtual controls and objects relative to the user's body in immersive virtual environments [20], but did not attempt to carry this work to the desktop.

A number of systems have attempted to go beyond the limitations of drag and drop in desktop 3D worlds. Herndon's work on interactive shadows [16] addressed difficulties of drag and drop with occlusion and depth ambiguity in 3D, but does not help the user move objects to distant locations.

Robertson's Fix and Float technique [25] does allow the user to move multiple objects simultaneously to distant locations, but objects being transported consume valuable screen real estate. In addition, toolspaces and glances are more general than Fix and Float because they can be used for more than drag and drop operations. One advantage that the Fix and Float technique has over toolspaces is that the relative positions of transported objects can be easily preserved; adding a group selection mechanism to our implementation would provide this functionality.

Local Tools [2] in KidPad, a 2D plus zoom (constrained 3D) system, allows users to drag tools onto their work surface and leave them in different locations, recalling them when needed by clicking on a toolbox in a corner of the screen. Although this allows the user to effectively transport the tools anywhere in the scene while only consuming screen real estate for the toolbox, the toolbox has a very limited size and can't transport anything but the tools.

More researchers are starting to look at the use of multiple 2D devices for interaction on the desktop. Kurtenbach [19] and Zeleznik [31] explored the use of two-handed tablet based interfaces for 2D and 3D drawing programs. Hinckley [17] explored both a two-handed tablet based interface and an interface using physically separate devices (a touchpad and mouse) for panning and zooming maps. Buxton [7] explored the use of physically separate devices for scrolling an application with the non-dominant hand while interacting with it using the dominant hand. Bier's Toolglass system [3] used a trackball in the non-dominant hand to position a toolglass and a mouse in the dominant hand to interact with it and the application.

Several VR authoring tools and interaction techniques use 3D tool palettes. Butterworth's 3DM [6] provided a tool palette in an immersive VR system that could be attached to the user. SmartScene [28] and Mine's ISAAC [21] are both immersive VR systems that provide tool palettes that attach to the user's hand and allow the user to add or modify objects. Polyshop [1] attaches tool palettes for object creation in an immersive environment to a physical desk in front of the user. Billinghurst explored the use of a tool palette attached to a Wacom tablet for creating, coloring, and texturing objects [4] in a fishtank VR system [30].

Stoakley, Pausch, and Elvins each explored the use of 3D miniatures for world navigation. Stoakley gave users a hand-held World in Miniature [29] in an HMD-based virtual world. Users navigated by changing the position of the doll that represented them in the miniature. Pausch later extended this work with the idea of flying into the WIM [22]; users moved from place to place in the world by flying into the hand-held miniature. Elvins presented an idea for navigation in VRML worlds using Worldlets [11]. By selecting one of these 3D thumbnails the user moved his view in the VRML world to the viewpoint the Worldlet represented.

Different games have explored the use of mechanisms similar to glances. Quake [23] players can rotate the camera independent of their motion vector to look in arbitrary directions, and if they did not actively maintain that view the system would automatically re-center their view. Most flight and racing simulator games provide a fixed set of “in cockpit” views, although users generally explicitly toggle between these views and don’t have any additional useful interactions available when pointed in the new direction.

A few games have used a method similar to Toolspaces for one handed 2D interaction. The Goosebumps games [14] are a recent example: users could move their mouse to the bottom of the screen to scroll the screen up and reveal a fanny pack storing the items they were carrying. Clicking on an item caused their character to carry that item in their hand. Moving the mouse back up to the top of the screen caused the screen to scroll down and reveal the world again.

7 CONCLUSIONS AND FUTURE WORK

Toolspaces and glances provide a mechanism for storing, accessing, and retrieving objects in interactive 3D worlds. Stored objects are always available to the user but do not consume screen real estate, require the user to switch contexts between the 2D desktop and 3D world, or clutter the 3D world when not in use. This allows toolspaces to support tasks like accessing data storage, object manipulation and transportation, navigating the 3D world, and others where a variety of 3D widgets are needed.

We believe that there are several potential avenues for future work. One is to explore other layouts and constraints for placing objects in toolspaces, especially if large numbers of objects need to be stored in a small number of toolspaces. We believe that different tasks will require different layouts and constraints, and determining the correct design choices will require careful experimentation.

Another possible avenue is to pursue alternative mechanisms of accessing toolspaces. For example, the user's non-dominant hand may already be occupied, requiring a method of accessing toolspaces without a separate gesture. One possibility is to access a toolspace by dragging the mouse cursor to the edge of the current view to rotate the camera to look into the toolspace on that side. The Windows 95 system tray and the NT 4.0 auto-hide toolbars implement similar functionality, where the tools are hidden until the user drags the mouse cursor to the edge of the window.

Finally, we believe that the notion of toolspaces and glances is applicable to 2D user interfaces. Accessing storage (documents, the web), moving objects (drag and drop), and navigation (virtual desktops) are all common operations in 2D systems. Toolspaces and glances can provide another method of accessing objects that are outside of the current context or view in these systems.

8 ACKNOWLEDGEMENTS

We owe thanks to all the members of the User Interface research group who offered suggestions, comments, and feedback. In addition, David Thiel performed above and beyond the call of duty in helping us to create the video that accompanies this paper.

References

[1] K. C. Abel, M. Alic, J. M. Moshell. The PolyShop Final Report. Institute for Simulation and Training, 1995. Available at http://www.vsl.ist.ucf.edu/polyshop.

[2] Benjamin B. Bederson, James D. Hollan, Allison Druin, Jason Stewart, David Rogers, and David Proft. Local Tools: An Alternative to Tool Palettes. User Interface Software and Technology, 1996, pages 169-170.

[3] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. ACM SIGGRAPH Conference Proceedings, 1993, pages 73-80.

[4] Mark Billinghurst, Sisinio Baldis, Lydia Matheson, and Mark Philips. 3D Palette: A Virtual Reality Content Creation Tool. Virtual Reality Software and Technology , 1997, pages 155 - 156.

[5] Frederick P. Brooks. Grasping Reality Through Illusion - Interactive Graphics Serving Science. Human Factors in Computing Systems, 1988, pages 1-11.

[6] Jeff Butterworth, Andrew Davidson, Stephen Hench, and T. Marc Olano. 3DM: A Three Dimensional Modeler Using a Head-Mounted Display. Symposium on Interactive 3D Graphics, 1992, pages 135-138.

[7] William Buxton and Brad Myers. A Study in Two-Handed Input. Conference on Human Factors in Computing Systems, 1986, pages 321-326.

[8] Brookshire D. Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-Dimensional Widgets. Symposium on Interactive 3D Graphics, 1992, pages 183-188.

[9] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. Computer Graphics, 1993, pages 135-142.

[10] Michael F. Deering. HoloSketch: A Virtual Reality Sketching/ Animation Tool. ACM Transactions on Computer-Human Interaction, vol. 2 no. 3, Sep. 1995, pages 220-238.

[11] T. Todd Elvins, David R. Nadeau, and David Kirsh. Worldlets - 3D Thumbnails for Wayfinding in Virtual Environments. User Interface Software and Technology, 1997, pages 21 - 30.

[12] T. Todd Elvins, David R. Nadeau, Rina Schul, and David Kirsh. Worldlets: 3D Thumbnails for 3D Browsing. Conference on Human Factors in Computing Systems, 1998, pages 163 - 170.

[13] Scott Fisher, M. McGreevy, J. Humphries, and W. Robinett. Virtual Environment Display System. 1986 Workshop on Interactive 3D Graphics, 1986, pages 77-87

[14] Goosebumps: Escape from Horrorland is copyright 1996 by Dreamworks Interactive. Visit http:// www.dreamworksgames.com for more information.

[15] Kenneth P. Herndon, Robert C. Zeleznik, Daniel C. Robbins, D. Brookshire Conner, Scott S. Snibbe, and Andries van Dam. Interactive Shadows. User Interface Software and Technology, 1992, pages 1-6.

[16] Kenneth P. Herndon , and Tom Meyer. 3D Widgets for Exploratory Scientific Visualization. User Interface Software and Technology, 1994, pages 69-70.

[17] Ken Hinckley, Randy Pausch, John Goble, Neil Kassell. Passive Real-World Interface Props for Neurosurgical Visualization. Conference on Human Factors in Computing Systems, 1994, pages 452-458.

[18] Ken Hinckley, Mary Czerwinski, and Mike Sinclair. Interaction and Modeling Techniques for Desktop Two-Handed Input. User Interface Software and Technology, 1998 (to appear).

[19] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The Design of a GUI Paradigm Based on Tablets, Two-hands, and Transparency. Conference on Human Factors in Computing Systems, 1997, pages 35-42.

[20] Mark R. Mine, Frederick P. Brooks, and Carlo H. Sequin. Moving Objects in Space: Exploiting Proprioception in Virtual Environment Interaction. ACM SIGGRAPH 1997 Conference Proceedings, 1997, pages 19-26.

[21] Mark Mine. "ISAAC: A Meta-CAD System for Virtual Environments. Computer-Aided Design: to appear.

[22] Randy Pausch, Tommy Burnette, Dan Brockway, and Michael E. Weiblen. Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures. ACM SIGGRAPH Conference Proceedings, 1995, pages 399-400.

[23] Quake is a product of id Software. More information is available at http://www.idsoftware.com.

[24] G. G. Robertson, S. K. Card, and J. D. Mackinlay. Information Visualization Using 3D Interactive Animation. Communications of the ACM, 36(4), 1993.

[25] George G. Robertson, and Stuart K. Card. Fix and Float Object Movement by Egocentric Navigation. User Interface Software and Technology, 1997, pages 149 - 150.

[26] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data Mountain: Using Spatial Memory for Document Management. User Technology , 1998 (to appear).

[27] Abigail J. Sellen, Gordon P. Kurtenbach, and William A. S. Buxton. The Role of Visual and Kinesthetic Feedback in the Prevention of Mode Errors. INTERACT '90, 1990, pages 667-673.

[28] SmartScene™ is a product of MultiGen Inc. More information on SmartScene™ is available from MultiGen's website at http://www.multigen.com/prodist/mgproducts/sstrain/strain.htm.

[29] Richard Stoakley, Matthew Conway, and Randy Pausch. Virtual Reality on a WIM: Interactive Worlds in Miniature. Conference on Human Factors in Computing Systems, 1995, pages 265-272.

[30] Colin Ware, Kevin Arthur, and Kellogg S. Booth. Fish Tank Virtual Reality. Conference on Human Factors in Computing Systems, 1993, pages 37-42.

[31] Robert C. Zeleznik, Andrew S. Forsberg and Paul S. Strauss. Two Pointer Input for 3D Interaction. Symposium on Interactive 3D Graphics, 1997, pages 115-120.