Three-Dimensional User Interfaces for Immersive Virtual Reality

Research Grant No. NAG 2-830

FINAL REPORT
(April 1, 1996 to March 31, 1997)

Brown University Computer Graphics Group
Dept. of Computer Science
PO Box 1910, Brown University
Providence, RI 02912

Principal Investigator: Andries van Dam

Abstract

This document is a final report of the research completed by The Brown University Graphics Group under NASA Research Grant NAG 2-830, April 1993 to March 1997. The focus of this grant was to experiment with novel user interfaces for immersive virtual reality (VR) systems, and thus to advance the state of the art of user interface technology for this domain. Our primary test application was a scientific visualization application for viewing computational fluid dynamics (CFD) datasets. This technology has been transferred to NASA via periodic status reports and papers relating to this grant that have been published in conference proceedings. This final report summarizes the research completed over the past year, and extends last year's final report of the first three years of the grant.

1 Project Description

The primary goal of this research project was to develop novel user interface techniques for use in immersive virtual environments (IVEs). We tested these techniques in an application that creates and manages visualizations of three-dimensional computational fluid dynamics (CFD) datasets. We have been developing techniques for use with both 2D desktop and immersive virtual reality (VR) environments, though prior to this past year, we had only tested them on the desktop. We did not attempt to build a complete scientific visualization system. Rather, we addressed the lack of easy-to-use user interfaces commonly found in existing commercial CFD visualization applications including [1][9][16]. The user interfaces for these applications are generally implemented and used within the familiar two-dimensional desktop metaphor and consist, therefore, of interface primitives like menus, buttons, sliders and so forth. These widgets collectively reflect the current state of a visualization and provide a means for controlling its parameters. While these interfaces are often complete (i.e., one is able to change any part of a visualization), they can also be difficult to use. A typical user interface to a commercial application consists of many 2D interface widgets which both consume large amounts of screen real estate and clutter the workspace.

In designing the user interfaces for this grant, we have adopted a different strategy which stresses the so-called "direct manipulation" of the visualization tools themselves. This approach obviates the need for so many 2D widgets by placing the controls for parameters like position and orientation directly into the 3D world of the CFD data itself. In addition, this approach provides a convenient means for combining related parameters into single widgets. Since the CFD datasets and the techniques used to visualize them are inherently three-dimensional, we can apply our knowledge of 3D user interface design to this new domain. The environment we use for this exploration is our own 3D graphics application development system, called UGA [26].

2 Accomplishments

The research accomplishments of the past year have centered on designing and implementing novel user interfaces for Immersive Virtual Environments. Our driving application was a scientific visualization application derived from the Virtual Wind Tunnel system [5]. The application supports visualization and interaction primarily through 3D Widgets [7] and was used as a proof-of-concept environment for our research. The application requires a high degree of user interaction to explore CFD datasets and served as a useful test environment.

This past year we have done the following:

- Designed and implemented a variety of interaction techniques for IVE systems
  - Object selection and manipulation
  - Navigation techniques
- Developed a rapid prototyping system for VR input devices
  - Enables hand-held widgets
  - Facilitates building customized input devices

- Performed pilot studies to evaluate these user interface designs

The remainder of this report will describe each of the above items in more detail; some of these accomplishments have been presented at computer graphics and user interface conferences.

3 Development Environment

3.1 Software

We have developed two applications with which to experiment with user interfaces for scientific visualization tasks. These applications were written using our in-house graphics system, called the Unified Graphics Architecture (UGA) [26]. The first application was implemented in the FLESH programming language, an interpreted, object-oriented language developed by the Graphics Group. This language served as an interface to the underlying functionality of the UGA system. This first application, which we used for the majority of our user interface experiments on the desktop, was ideal for this kind of experimentation because one could rapidly prototype new interaction techniques and widgets. The downside of FLESH, however, was its poor performance when using it for anything but simple designs.

The Graphics Group then implemented another system, called "trim-lite". This newer system is written in C++ and consists of a relatively small set of libraries which include classes for many of the same components of 3D graphics applications that the FLESH programming language supported, including geometric objects, cameras, lights and input devices. We chose C++ for this system because it afforded much higher performance than the interpreted FLESH language did. As our interface designs became more complex, we needed this increased performance to properly evaluate new interface designs.

Applications written using the "trim-lite" libraries are compiled and thus run more efficiently than FLESH applications. As part of this grant, we added a library to "trim-lite" which performs all of the low-level scientific visualization functions of our prior application, and designed a framework in which we could continue our user interface experiments in IVEs. The resulting application, which is still under rapid development, includes many of the user interface components of the earlier FLESH application. However, we have focused more strongly on implementing new techniques specifically tailored for immersive virtual reality.

We have tested our system with relatively simple (less than 500,000 points) steady-flow datasets using both regular and curvilinear computation grids. The majority of our work has been done on a curvilinear dataset of airflow velocity (speed and direction) past the body of the Space Shuttle.

3.2 Hardware

On the desktop, we have used both conventional hardware (CRT and 2D mouse), and "fishtank virtual reality" hardware, including a Logitech 6D mouse and StereoGraphics LCD shutter glasses.

In our VR lab we primarily use a single Ascension Bird tracker for one-handed input in conjunction with a Binocular Omni-Orientation Monitor (BOOM) for head tracking and stereoscopic display. We also used our Virtuality Head Mounted Display (HMD) which we acquired in the last month of the grant period. The tracker is equipped with three buttons for additional input. In its default configuration, the tracker controls the position of a simple 3D crosshair cursor in the virtual world.

We also have a glove input device which can be used to input more complex data to the application such as postures and gestures of the hand. We have not yet used the glove directly within our scientific visualization application due to the significant calibration time and inconsistent data we receive from the device (e.g., we have had significant trouble repeatedly recognizing anything but a few extremely different postures). We have found attaching trackers to points of interest (e.g., a user's fingertips) is an effective alternative to using the dataglove for the interaction techniques we have developed.

4 3D Widgets and Interaction Techniques

In the following sections, we will discuss the various user interface issues and designs that we have worked on over the course of this research. We begin with a few definitions, and present a taxonomy of 3D graphics application tasks, then discuss our user interface design methodology, and finish with descriptions of the specific interaction techniques and 3D widgets that we have implemented.

4.1 Definitions

A *widget* is an entity which possesses both geometry and behavior [7]. We define the geometry of a widget to be its visual appearance when rendered to an output device. The behavior of a widget represents its functional role in an application and defines both how it reacts to user interaction as well as how it affects aspects of the application environment when manipulated by a user. At this fundamental level, 2D and 3D widgets are identical. However, on a practical level, they differ in that 3D widgets exist in a 3D scene whereas 2D widgets exist within a 2D windowing environment. We do not consider 2D widgets that have a 3D "look" (typically achieved with drop shadows) to be true 3D widgets.

According to this definition, a wide range of entities can be called widgets. In practice, we divide this spectrum into specific categories (Figure 1). At one extreme are widgets with geometry but no behavior. These entities, which we call "primitive objects", are the building blocks of virtual environments, and are defined by their geometric attributes (position, size, orientation, color, etc.). When modified by a user, there are no side effects in the surrounding environment.

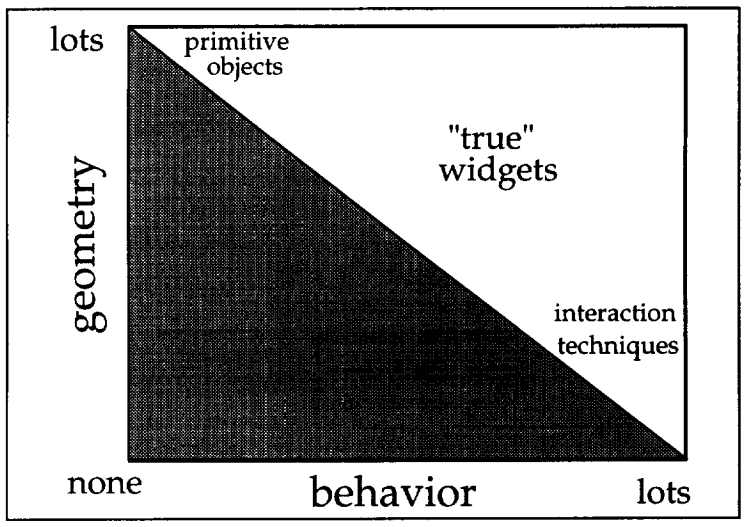

Figure 1: The spectrum of geometry and behavior in widgets. The ratio of geometry to behavior determines whether the object is a primitive, an interaction technique, or a "true" widget with a combination of both geometry and behavior.

At the other end of the spectrum are widgets with behavior but no geometry. The purpose of these entities, or "interaction techniques," is to translate raw user input (e.g., mouse deltas, button presses, etc.) into meaningful actions in a 3D scene. For example, an interaction technique for panning a camera in a desktop application converts 2D mouse deltas into transformations which are applied to the camera viewing a 3D scene. Likewise, an interaction technique for manipulating a 3D object converts mouse deltas into transformations which are applied to the object. Generally, the only feedback that an interaction technique supplies is its actual effect on the scene (e.g., the motion of the camera or object being manipulated).

In the middle of the spectrum lie an array of "true" widgets which contain a more balanced mix of geometry and behavior. Like interaction techniques, the behavioral components of these widgets serve to transform raw user input into values which are meaningful to primitive objects. The geometry of a widget acts as a virtual input device through which we can indirectly modify attributes of an object that may be unnatural or impossible to access directly through a simpler interaction technique.

Both interaction techniques and widgets can be used to modify spatial or non-spatial parameters of primitive objects. Widgets with geometry are often designed to provide feedback to the user, though this is not always necessary. For example, the rake widget, described later, utilizes a one-dimensional slider to set the number of streamlines along its bar. The position of the slider simultaneously determines the distance between the streamlines on the bar and provides feedback to the user about this spatial quantity.

4.2 Taxonomy of Tasks

In interactive 3D graphics applications, tasks can be classified according to the

following three categories:

- selection (of objects)
- manipulation (of objects, parameters, etc.)
- navigation (of viewpoint)

Most of the interaction techniques and widgets that we have implemented fall neatly into one of these categories. Section 6 describes the techniques that we have implemented for our scientific visualization application.

This taxonomy expands on Robinett and Holloway's [23] presentation of the manually-controlled actions that may be implemented in an IVE which involve changing the location, orientation or scale of either an object in the IVE (manipulation) or the user herself (navigation). In Robinett's treatment, the selection task is implicitly included as part of manipulation. We have chosen to make selection a unique category because as with manipulation and navigation tasks, there are a wide variety of unique methods for performing this task.

5 Design Strategy

Our strategy for developing interaction techniques for immersive VR was to facilitate user interaction with 3D widgets. Three dimensional widgets (defined in Section 4.1) are used frequently in direct-manipulation applications and the basis for all visualization tools in our test application. Facilitating interaction with 3D widgets is a more complex task than facilitating interaction with single objects because a 3D widget typically has multiple components with which the user can interact. Often these components are designed to minimize intrusiveness in the scene. Consequently, widget components are close to one another and as small as possible. There is a close relationship between 3D widget design (how small and compact a widget is) and the effectiveness of the interaction techniques that will be used to interact with the widget. An interaction technique for use in an immersive environment with 3D widgets must enable precise object selection, intuitive subsequent manipulation, and effective navigation through the environment. In addition, we have followed the General Design Strategies reported in last year's final report of work supported under this grant.

This strategy led us to the development of improved software techniques for interaction as well as new hardware devices that allow rapid prototyping of "hand-held" widgets and the construction of custom input devices (see Section 6).

6 Implementations

The following sections describe implementations of the user interfaces that we have experimented with over the course of this research. When appropriate, we discuss how particular design decisions were made, and how these designs relate to the design strategy described above. The first three sections address each of the tasks outlined in the taxonomy above. The final section describes our rapid prototyping system for input devices.

6.1 Selection

Selection is one of the fundamental tasks in any interactive graphics application. Through selection, users indicate which objects they are interacting with and specify which parameters they wish to view or modify. The method used to select an object, however, differs greatly depending on the type of input and output devices at hand and the application itself.

6.1.1 Desktop

There are a variety of configurations of input and output devices for desktop systems, ranging from the conventional CRT and 2D mouse combination, to more elaborate stereo displays and 3D input devices. Selection techniques for the latter will be discussed in the next section since many of the issues are the same.

When using conventional desktop hardware (2D mouse input and standard CRT output), the method we usually utilize for selection is *ray intersection*. We construct a ray based at the focal point (viewpoint) of the camera through the point on the image plane which corresponds to the position of the 2D mouse cursor. By testing for intersections between this ray and all of the geometry in the scene, we can determine which object the mouse cursor was "over" in the 2D projection of the scene, and select that object (usually in response to a button press). We have found that this technique is very effective on the desktop because from a perceptual point of view, it emulates the "point and click" behavior of 2D desktop windowing systems. Consequently, this is a general method of selection for desktop applications that have both 2D and 3D components. Also, since the mouse is a virtually noiseless and thus very precise input device, we are able to select very small objects which project to only a few pixels on the screen. As we will see, this is not the case for IVE input devices.

6.1.2 Virtual Reality

Just as on the desktop, selection techniques are used in IVE's to specify which object(s) in the environment the user wishes to interact with. In the real 3D world, of course, people "select" objects by touching them with their hands, or indicate a particular object in the distance by pointing in its general direction. When designing and implementing selection techniques for virtual reality applications, we can look to this real world human behavior for inspiration. However, in VR applications, the selection task is complicated by a number of factors, including limitations imposed by input and output devices as well as by software techniques.

In VR, user actions are mediated by input and output devices which are often imprecise and which either lack or distort perceptual cues that we take for granted in the real world such as haptic feedback, stereopsis, field of view, texture (both visual and tactile), and sound. Display devices like the BOOM and our HMD provide stereoscopic views, but some people are not able to resolve depth from this type of display. Other head-mounted display devices typically provide fairly low-resolution images, making it difficult to resolve small objects. Magnetic 6D trackers introduce problems as well. First, they are not always accurate in many real environments

because they are adversely affected by metallic objects and electronic devices in the physical environment. This can cause significant registration error between the actual and displayed position of the tracker. A number of techniques have been proposed and implemented to correct for static distortions (those that do not change significantly over time) of this type [4][11], including a method developed here at Brown [12].

Secondly, the data reported by magnetic trackers is somewhat noisy, so even when they are held perfectly still, the 3D cursor in the scene appears to randomly jump around. We found the magnitude of this noise in our tracker is approximately 0.1". The combination of hardware and software used to convert magnetic fields into position and orientation values that can be used by our software introduces lag into the system. This results in a delay between the actual tracker movement and the display update in the IVE. As shown in [20], lag contributes significantly to error.

On the software side, the *selection tests* used by most virtual reality applications consist of precise, mathematical tests (e.g., ray intersection or point enclosure). While we have found that these seem to work well for desktop applications, they are not nearly as successful in IVE's. As our pilot studies have shown, this is in part due to the physical limitations of the tracking and display hardware we use. However, there may also be more subtle phenomena at work. For example, since IVE's strive to provide an experience which mimics many of the perceptual qualities of our real-world experience (including stereopsis, wide field of view, etc.), users of IVE applications may thus presume that they can interact with objects in an IVE in the same way that they interact with objects in the real world. Unfortunately, computers are not yet adept at inferring user intentions exclusively from the kind of vague indications which humans are accustomed to using for communicating with one another in the real world (pointing, gesturing and speaking, for example). As a result, designing effective selection techniques for VR applications is a tricky process. Our general design methodology has been to look to the real world for examples of how people select, indicate or manipulate objects, and transfer qualities of these interactions into software techniques in an IVE. Often, the resulting interaction technique in an IVE is very different from its real world source since the software technique must both cope with limitations of the hardware devices, as well as exploit the "magical" properties of an IVE (such as the ability to manipulate objects at great distance, which can not easily be done in the real world).

In the following sections, we will describe some previously existing selection techniques (Section 6.1.2.2 through Section 6.1.2.5) and the selection techniques we have developed (Section 6.1.2.6 through Section 6.1.2.9). We also discuss the advantages and disadvantages of each technique.

6.1.2.1 Virtual Input Devices and Physical Props

A virtual input device in an IVE is analogous to the mouse cursor on the desktop. It is a piece of geometry which represents the state of the input device(s) currently being used. In the case of a single 3D tracker input device, the virtual input device (VID) might be a simple crosshair cursor (Figure 2a). A more complex VID is a glove driven by a glove input device where the geometry drawn reflects the user's current hand posture. Generally, the appearance of a VID depends on a number of factors,

including the type of physical input device, the task it is being used for, and any physical modifications that have been made to the device itself (props). Some of the selection and manipulation techniques that we have implemented were motivated by observations of how people operate with tools in the real world and are implemented with these metaphors in mind. Therefore, when we have felt it appropriate, we have used props to emphasize the metaphor. In other research [13], props have been shown to aid users' understanding of user interfaces for virtual reality applications.

We make use of a number of different props in our lab, including a drumstick and a ski pole handle. We modify the geometry of the VID to suggest the shape of the prop (Figures 2b and 2c). Though this is not strictly necessary, it is often helpful because a user can more easily correlate what she sees in the IVE with the physical sensations she perceives of the actual object in her hand.

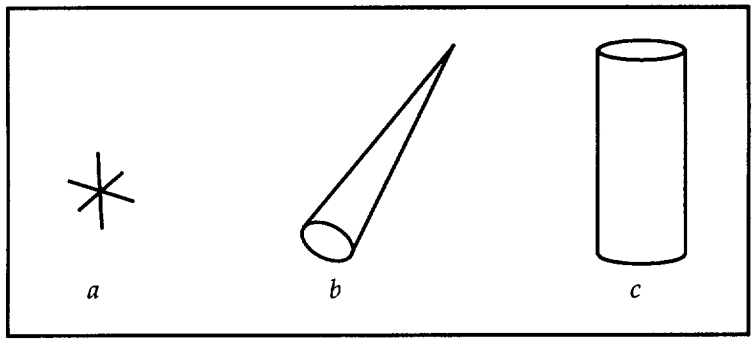

Figure 2: Three types of virtual input devices. *a*) a simple crosshair cursor; *b*) a cone (used with the drumstick); *c*) a cylinder (used with the ski pole handle)

6.1.2.2 Touch

In terms of borrowing ideas from the real world, this might be considered the most straightforward selection technique since humans are very familiar with touching objects first in order to manipulate them. From an implementation point of view, the input to this technique is also fairly simple, requiring only a 3D position (e.g., from a tracker) and a means of signaling to the application when to select an object (e.g., a button press, voice command, etc.). We have implemented two variations of the touch selection technique in our system. In the first, the position of the tracker determines the placement of the 3D cursor used for the selection test. When the 3D cursor is placed inside the geometric boundary of an object, that object can be selected by pressing a button on the tracker. We may use other input methods such as speech recognition to replace the button as a way for the user to signal selection. This technique is similar to touching objects in the real world, except that there is no haptic feedback (i.e., subjects can not feel the object in the virtual world). One drawback of this technique is that it requires that the desired object be within reach. If the object is out of reach, then the subject must first navigate to the vicinity of that object in order to select it, and the selection task thus becomes a two-step process.

The second variation was inspired by observations of glass blowers who manipulate objects at a distance with long sticks so as not to be burnt by the hot glass. In our implementation, we use the drumstick prop, which emulates the glass blower's

tool, and place the 3D cursor at the end of the corresponding VID in the virtual environment. Using the drumstick increases the distance from the user that objects can be selected, but may make it more difficult to select objects at close range due to the awkwardness of holding the stick in these positions. It is still not clear which of these techniques is better overall, but they clearly each have particular strengths and weaknesses. User studies will, of course, help to determine which approaches are better.

Note that due to noise in the position values reported by the tracker, coupled with normal jitter in the user's hand, the VID in this technique (and in others) appears to jump around even when attempting to hold it perfectly still. This phenomenon indicates that objects must be larger than some minimum size in order to be selectable with this technique (or that the technique itself must be modified). In the second variation, since the orientation of the tracker influences the position of the end point of the VID which is used for the selection test, this technique is susceptible to errors from noise in the orientation of the tracker as well as positional noise. In practice, these factors present severe usability problems which we have attempted to alleviate by modifying the technique (see the technique described in the next section).

Testing whether a given input point is within the geometric boundary of an object can be computationally very complex. To reduce this complexity, we can use a simplified geometric representation for objects in the selection test. In our first implementation, we used a sphere scaled to the 3D extent of the object. With this approximation, a simple analytic test could determine roughly when the cursor was in the vicinity of the target object. However, this simplified technique posed problems when the actual geometry of the visible object was very different from a sphere (e.g., if the object is convex). We have also used a polygonal collision detection system called "I-Collide" developed at UNC, Chapel Hill [6], which can accurately detect exact collisions between a VID (represented by a geometric object) and the objects in the scene at interactive rates.

6.1.2.3 Touch Plus Intersection Ray

In our evaluations of the touch method described above, we found that even though it seems like a very natural technique, and seemed to work fine for selecting large objects, it is nearly impossible to use it to select small or narrow objects. We have augmented the touch technique to remedy this shortcoming. In the basic technique, we simply test whether the cursor is inside the geometric boundary of each object (using either the simple spherical test or the true collision detection method). Due to the noise in the tracker and instability in the user's hand, this technique is extremely susceptible to the effects of temporal aliasing. Since there is no haptic feedback in this system to alert the user that she has touched a given object, the VID is allowed to pass through any objects. If the frame rate is not sufficiently high, or if the noise in the tracker and user's hand is significant compared with the size of the target object, there may be cases when the user feels that she has placed the cursor in the correct position, but still can not select the object. This situation most often occurs when at time t, the cursor is either inside the object or just to one side of it (Figure 3). Then, at time $t+1$, the cursor has passed either outside the boundary of the object or to the opposite side. In the touch technique, when the button is pressed at time $t+1$, the selection test obviously fails[1] (it

can still succeed in some cases if the approximate spherical test is used). However, if we consider the line segment between the sample point at time *t* and the position of the tracker at time *t+1*, we can determine which object the cursor passed through by looking for intersections between this line segment and all of the objects in the scene. When the button is pressed at time *t+1*, we select the appropriate object.

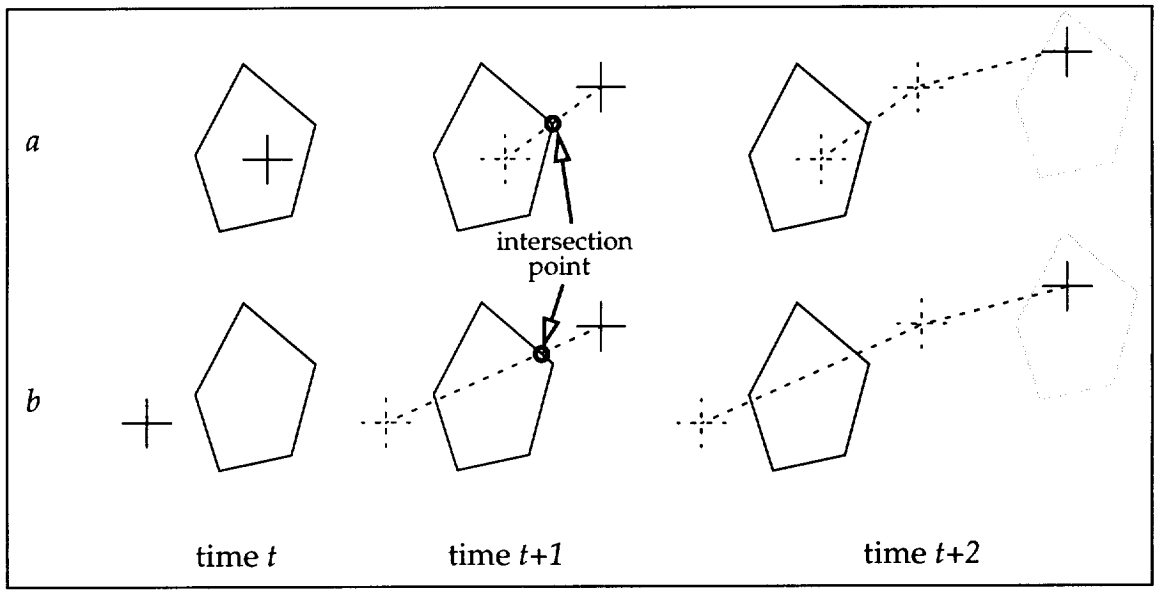

Figure 3: Augmenting the touch selection technique. At time *t*, the cursor is either inside the geometric boundary of an object (*a*), or just to one side (*b*). At time *t+1*, the cursor is outside the object (*a*), or on the opposite side (*b*). Finally, at time *t+2*, the cursor may have entered a second object (*c*). The dotted line segments in *b* and *c* are tested for intersection with the object.

Adding this heuristic does not by itself quite make a successful technique because if the user waits until time *t+2* before pressing the button (which is likely considering each frame is displayed for less than 0.1 second), then the line segment between the two sample points may not intersect any objects. We have thought of two possible solutions to this problem. First is to "remember" the last object that was intersected for some time interval (1/2 to one second) during which any button press will select that object. This method would still fail if the time between frame updates was longer than the memory interval. This will not occur in an IVE because the frame rate must be higher than 10 Hz.

Second, we might use a spatial measure to determine which object to "remember." That is, until the cursor travels more than a certain distance away from the last intersection point, a button press will select the last object that the cursor passed through. Of course, both the length of the time interval and distance travelled by the cursor should be determined by user studies. While we have not yet determined the optimal parameters, our pilot studies do indicate that the addition of these heuristics greatly improves the usability of the touch technique.

---

1. It is also possible that at time *t+1*, the cursor has entered the boundary of an adjacent object, in which case, that object would be selected instead of the intended one.

6.1.2.4 Laser Pointer

This technique gets its name from real laser pointers which are used in darkened rooms to point out distant objects. It uses ray intersection to perform selection similar to the desktop selection technique described earlier. In this technique, however, the base point and direction of the selection ray are determined by the position and orientation of the tracker, respectively. The ray points down one of the principal axes of the tracker's coordinate system (we have chosen the z axis). We generally use this technique with the drumstick prop, holding it as if it were a laser pointer. The physical prop reinforces this metaphor.

In our pilot studies, we found that this technique, though easily learned, presented severe problems when trying to select small objects even at close range. The noise from the device coupled with the inherent instability in one's hand causes the direction of the intersection ray to fluctuate by as much as ±5 degrees. At a distance of three feet, this error amounts to 6.25 inches, suggesting that objects any smaller than this are effectively unselectable at this distance. The techniques we describe below introduce methods to reduce the adverse effects of tracker and hand noise.

6.1.2.5 Target (Laser Pointer from Eye)

The target technique is also based on ray intersection, but borrows even more from the 2D desktop techniques than the laser pointer. In this technique, we cast a ray *from the viewpoint*, controlled by the position of the user's head, *through the 3D cursor* in the IVE, determined by the position of the tracker. This technique was inspired by and is similar to looking at a target through the sight on the barrel of a gun, or to holding up one's thumb to measure the size of a distant object. An important feature of this technique is that the user only need to specify *two* degrees of freedom when selecting objects instead of the usual *three or more* that other techniques (e.g., touch and spotlight selection) require. Although there is still noise in the position of the tracker, and jitter in the user's head and hand, we have observed in our pilot studies that the distance between the head and hand, which determine the basepoint and direction of the intersection ray, respectively, do in fact reduce the overall error in this technique (compared with the laser pointer which is adversely affected by both positional *and* rotational noise in the tracker). Given the absence of haptic feedback in our system, users may also find it more "natural" to select objects from their point of view than from their hand (this may be similar to the decreased accuracy of shooting a firearm "from the hip" compared with using the sight on the barrel).

When using this technique, we must also consider the fact that a stereoscopic image of a scene is produced from two separate eye points (cameras) simultaneously that are positioned side by side to match the physical configuration of the human eyes. Since the base point of the intersection ray in this technique is controlled by an eye point, we must decide which eye point to use. We know that most people have a so-called "dominant" eye which they favor over the other when performing tasks in which a single point of view is required. A simple test can determine which eye is dominant, and we can adjust the technique appropriately.

6.1.2.6 Aperture

The aperture selection technique [10] augments the target technique with an additional feature to help alleviate the adverse effects of error from noise. The visual representation of this technique consists of a circular aperture centered on the cursor point which is marked with a crosshair (Figure 4a). We have experimented with two uses of the aperture. The first (Figure 4c) places the aperture at the location of the 3D cursor, and aligns it with the film plane of the camera viewing the scene. As with the target method described above, a user places the aperture "over" the object(s) she wishes to select, then presses the button on the tracker. This configuration can be used with or without the drumstick prop, and the VID is modified appropriately.

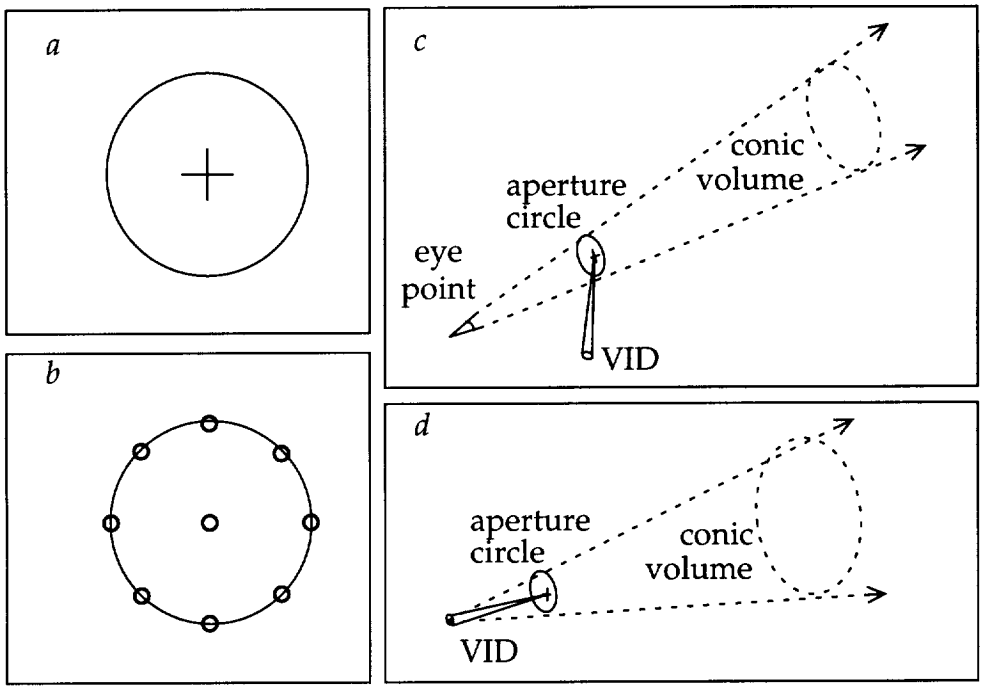

Figure 4: The aperture technique. a) The basic aperture geometry. *b)* Small circles represent the positions of intersection rays cast through the center of the aperture and representative points on the perimeter. *c)* The conic volume described by the viewpoint and aperture. *d)* The conic volume described in the "flashlight" configuration. Both *c* and *d* are shown with the drumstick VID. In our implementation, the conic volume is semi-infinite.

In the second configuration (Figure 4d) we use the drumstick prop and place the aperture geometry at the end of the stick, but this time align it with the plane perpendicular to the axis defined by the stick (similar to the intersection ray in the laser pointer technique).

In either configuration, the combination of a direction vector and the aperture circle defines a conic volume in space. In the first configuration, the apex of this cone is coincident with the eyepoint of the viewer. Any object which appears from the user's point of view to be inside the circular aperture can potentially be selected. In the second configuration, the effect is more like a flashlight sweeping through a region of

space; any object "illuminated" by the flashlight is potentially selectable. A similar technique using the flashlight metaphor was implemented by Liang [19] in the JDCAD system, but the from-eye version was not implemented.

The radius of the cone can be easily modified in the first configuration simply by drawing the VID closer to or further from one's eye. In the "flashlight" configuration, however, an external control is needed to change the radius of the aperture circle. Currently, we adjust this parameter from a command line interface, but we have envisioned a number of possibly more attractive possibilities including using voice recognition, a hardware dial on the physical input device (see the Lego Interface Toolkit described in section Section 6.4), or twisting the stick around its long axis (the central axis of the conic volume). We did try this last technique, but quickly discarded it after we found it very difficult to continue pointing the stick in the same direction while twisting it.

Intuitively, the objects that the aperture-based techniques should select are those that fall within the conic volume. This can be expressed as a test for intersections between the conic volume and all of the objects in the scene. We have tried the following techniques:

- Ray intersection from the base point through center of aperture (by itself, this is a degenerate case equivalent to the laser pointer or target techniques).
- Ray intersection from the base point through representative points on the aperture circle (Figure 4b).
- Project vertices and edges of objects in the scene onto the plane defined by the aperture circle.
- Using I-Collide to detect interpenetration of the conic volume with the objects in the scene.

I-Collide obviously is the technique of choice because it returns an exact solution, though in some complex scenes (e.g., with isosurfaces), it may be too slow because of the large number of polygons. In most of the simple scenes we have used it in, however, I-Collide seems to work very well and at interactive rates.

The first two tests above use the same ray intersection tests that the laser pointer and target techniques use. Computationally, ray intersection is inexpensive compared with projecting vertices and edges of objects in the scene onto a plane, but can easily miss objects that do lie within the boundary of the aperture circle and thus should be selected. The vertex and edge projection tests reduce the 3D intersection test to a 2D problem. Once the vertices and edges are projected onto a plane, we need only determine whether they fall within the circle. However, this technique is computationally very expensive and currently can not be used in complex scenes at interactive rates without hardware acceleration.

Since the aperture technique defines a volume of space, it is possible that we may have multiple candidate objects for selection. In the event that more than one object lies inside the conic volume, we may either select all of the objects, or identify a single object for selection based on some criteria. In practice, there are a variety of mathematical tests to determine which of a number of objects to select. One possibility is to choose the object whose center point is closest to the base point of the intersection

ray (either the viewpoint or location of the tracker, depending on the particular configuration in use). This method presents a problem, however, when a user attempts to select an object near the center of the aperture, but a closer object is partially inside the edge of the circle. In this case, both objects are candidates, but the second, closer one will be selected because it is closer to the viewpoint.

Another possibility for choosing a single object among many potentially selectable objects, and one that appears to be the most intuitive, is to select the object closest to the ray passing through the center of the aperture circle. Note that this object may not be the closest object to the viewer. This method recognizes the user's intuition (backed up by anecdotal evidence in our user studies) that the more "centered" an object is in the aperture, the more likely that it will be selected.

A third option is to select an object based on its apparent size (e.g. select the largest object in the aperture). In practice, this is not such a good option, since it may be difficult to select a small object next to a larger one. Also, if two objects of roughly the same size are at different distances from the viewpoint, their apparent size will, in a perspective projection, be very different as well. Using the size test to determine which object to select may work in this case, but a test based on distance would probably work just as well. As mentioned, in practice, we have found that the distance test is more appropriate than an apparent size test.

Note that this technique is also subject to modification based on each user's dominant eye.

6.1.2.7 Glove-Based Interface for Aperture

Though we have not implemented it yet, we have designed a glove-based interface to the aperture technique which we feel is more natural than the two configurations described above. This technique utilizes the posture recognition software in our system to identify when the user has shaped her hand in a pinching posture. When this posture is recognized, the aperture geometry is drawn between the index finger and thumb. As with the first configuration of the basic aperture technique, the aperture is aligned with the film plane. Two advantages to this technique are that the user does not have to hold a prop, and also that the size of the aperture can be adjusted simply by moving one's thumb and index finger further apart or closer together.

6.1.2.8 Orientation

The orientation selection technique [10] selects objects in an IVE by comparing the orientation of the tracker with the orientations of objects in the scene. Any objects which approximately match the orientation of the tracker are candidates for selection. The inspiration for this technique was the observation that in the real world, when we attempt to grab an object with our hand and fingers, we first must configure our hand so that it conforms to the part of the object we reach for (e.g., the handle of a cup, a telephone, or the middle of a bar, etc.). At a very gross level, the task we perform is matching the orientation of our hand with that of the target object. We can approximate this with a simple mathematical test.

According to this technique, the shape of an object plays a direct role in determining how it can be selected. In the UGA system, primitive objects initially have a canonical uniform scale. Given this property, the scale components of an object's current transformation matrix (CTM) can reveal information about it's shape. Long thin objects and flat objects can be easily identified by significant differences in their x, y and z scale components. Of course, objects which are long and thin but which are not aligned with a principal axis will not be so easily identified. In general, determining the shape of an object may be a harder, more subjective problem that involves at least an analysis of the object's geometry, and perhaps even some higher-level semantic knowledge about important features of the object (such as the handle of a cup or knob on a door). However, for some simple cases, we can get reasonable behavior under the current scheme.

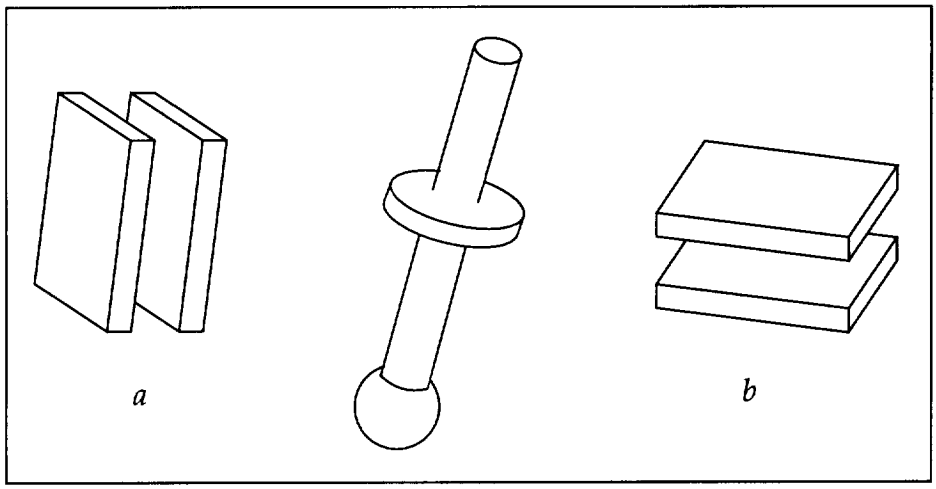

Figure 5: Orientation selection technique. For this technique the cursor geometry is a pair of parallel plates which indicates the current orientation of the tracker. This cursor may be used by itself or in conjunction with a VID such as the drumstick. a) shows the cursor orientation that would select long, skinny objects like the bar of the object in the middle of the figure. b) shows the cursor orientation that would select short, wide objects, like the disc on the bar. The ball at the end of the bar presents something of a problem for this selection technique. This can be remedied by adding a heuristic which identifies uniformly-scaled objects and compares distance to the cursor rather than orientation.

In our application, we tested this technique on the rake widget, which consists of a bar (a long thin cylinder), a slider on the bar (a flattened cylinder), and a ball at one end of the bar (a small sphere). We modified the geometry of the VID for this technique so that the orientation of the tracker was clearly represented (two parallel flat blocks placed side by side). As the user rotates the tracker, so does the cursor rotate and thus indicate the kinds of objects that can be selected (Figures 5a and 5b).

Note that if this selection technique uses only the relative orientations of the tracker and objects in the scene to determine which objects it will select, and not their respective positions, it can select objects that are a significant distance from the tracker (or even outside the view). In our initial tests, we found this behavior unnatural. To cope with this drawback, we used orientation as a means to disambiguate which single

object should be selected when there were multiple candidates for selection (as is the case with the aperture or flashlight techniques where all objects that fall in a conic volume are candidates for selection).

The orientation of objects which are uniformly scaled, like the sphere at the end of the bar, is not readily apparent. In these cases, this selection technique uses only the distance measure to determine selectability. In this particular instance, a glove-based posture recognition interface might be more effective – the user would simply shape her hand to fit the desired object. Such an interface might compare the convex hulls of the user's hand and nearby objects and pick the one with the closest match.

6.1.2.9 Image Plane Techniques

We have developed and implemented a variety of novel selection techniques for IVEs. We have named these techniques "image plane" selection techniques [22]. When using an image plane selection technique, a user selects an object by interacting with the *2D projection* of an object as opposed to the actual *3D geometry* of an object. The desktop analog is the use of a mouse to interact with objects in a 3D scene based on their projections on the monitor screen.

Consider the task of selecting a streamline widget that is beyond a user's reach (see Figure 6). Using a film-plane selection technique, the user selects the widget by positioning her hand in the 3D scene so that the projection of her index finger on her image plane is positioned on top of the projected image of the widget. The user does not need to navigate to the object or need information about the actual size or distance of the object to interact with it. In addition, the user need only specify two degrees of freedom instead of three or more as required by other techniques.

We have implemented several examples of image plane selection techniques (see Figure 7). In the *Head Crusher* technique, the user positions her thumb and forefinger around the desired object in the 2D image. We determine which object is between the user's fingers in the image plane by casting a pick ray into the scene from the user's eye-point through the point midway between the user's forefinger and thumb.

The *Sticky Finger* technique provides an easier gesture when picking very large or close objects by using a single outstretched finger to select objects. The object underneath the user's finger in the 2D image is the object that is selected. To determine which object the user has selected, we cast a ray into the scene from the user's eye-point through the location of the tip of the user's index finger in the scene. Objects intersecting this ray are beneath the user's fingertip in her image plane.

We extend the notion of using the image plane for selection with the *Lifting Palm* technique. We borrowed the idea for the Lifting Palm technique from the famous optical illusion of a man in the background of a photo apparently standing on the palm of a man in the foreground. The user selects an object by flattening his outstretched hand and positioning his palm so that it appears to lie below the desired object on his image plane. We determine which object the user has selected by finding the current location of her palm in the scene and imposing a slight offset on it. The ensures that we check for the object that lies above the user's palm in the image plane. We then cast a

pick ray into the scene from the eye-point through that position.

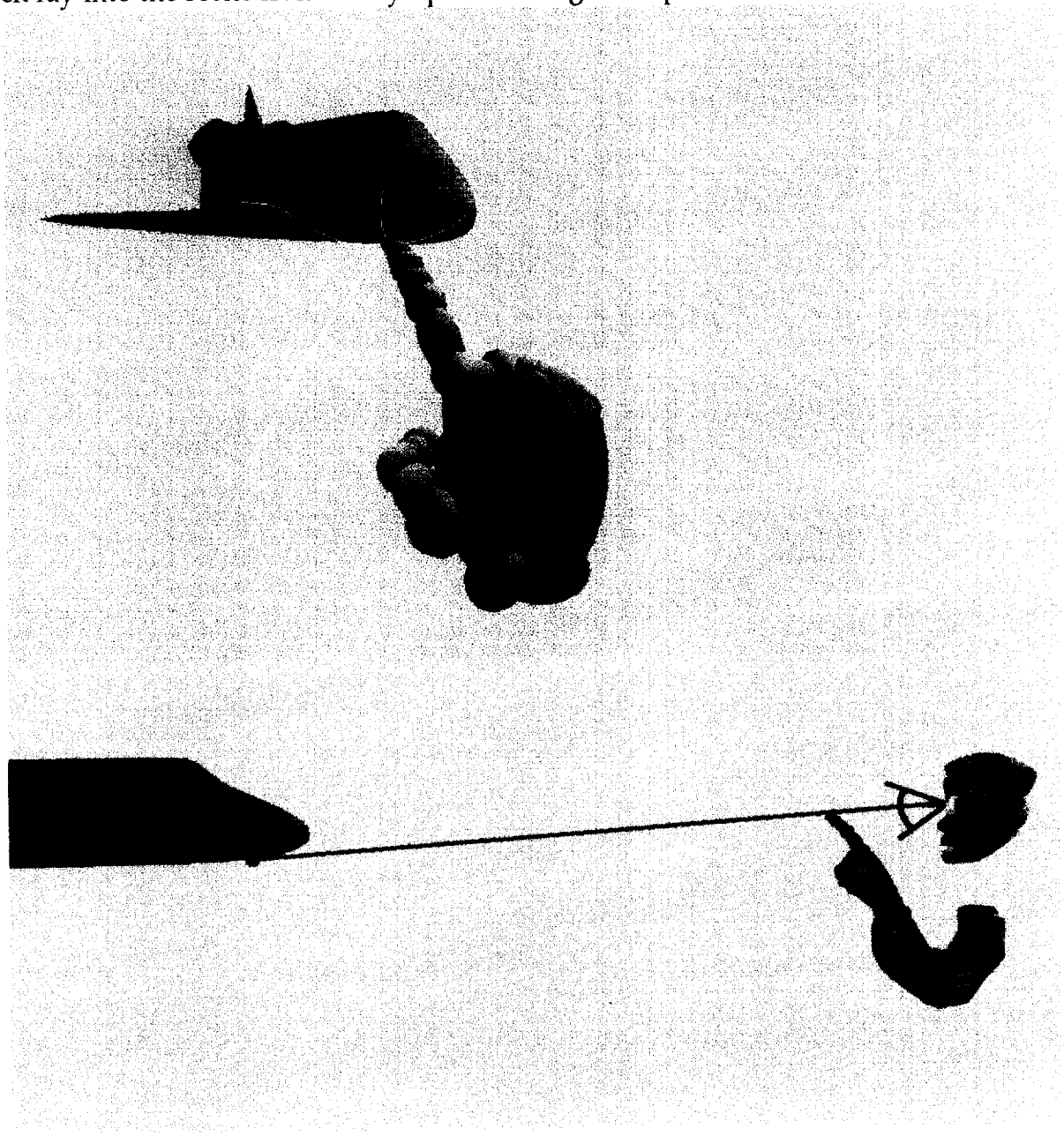

Figure 6. Selecting an object using the sticky finger selection technique. The upper left image shows the first person view of the user selecting the red streamline widget. Below, a third person view of the user, their hand, and objects in the world are shown. The closest object on the ray from the user's eye through their index finger is selected.

The *Framing Hands* technique uses both hands to select objects. Using this technique the user positions her hands to form the two corners of a frame in the 2D image. The user then positions this frame to surround the object to be selected. The implementation for this selection is similar to the Head Crusher's implementation. We determine the location of the user's hands, and then calculate the midpoint between the two in the scene's coordinate system. We again cast a ray into the scene from the user's

eye-point through that midpoint to select an object.

The user can also use the Framing Hands technique to select a group of objects by selecting all of the objects that lie within the frame formed by the user's hands. This is similar to the 2D rectangle selection technique used in many desktop applications, except that this technique allows the user to use both hands simultaneously to specify the desired selection area. The user can quickly and arbitrarily rotate or resize the frame formed by her hands with a motion of her hands. We can draw the frame explicitly on the image plane to provide additional feedback to the user for which objects will be selected.

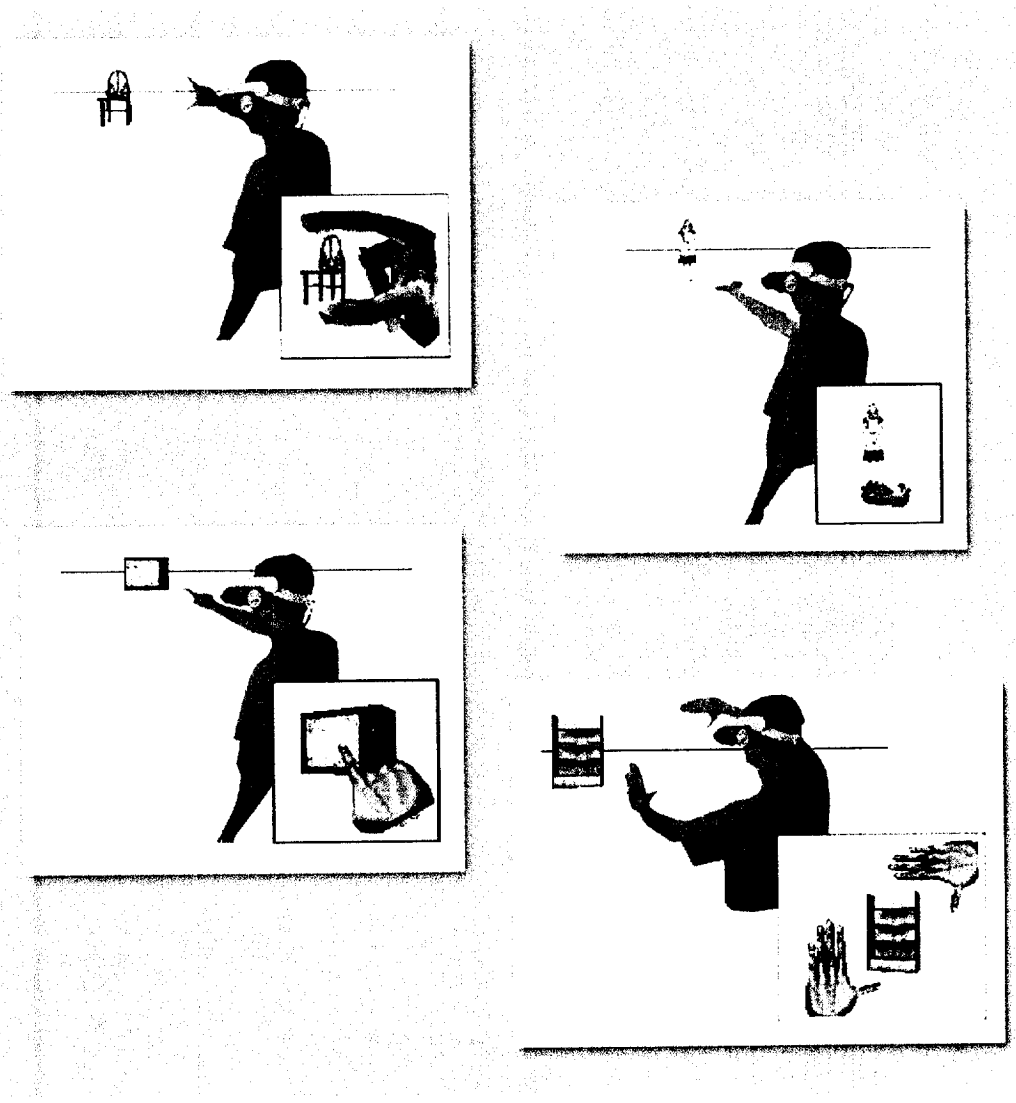

Figure 7: Image plane selection. The user selects an object by positioning their hand over or around the target geometry and signaling to the computer to select. The *Head Crusher*, *Lifting Palm*, *Sticky Finger*, and *Framing Hands* techniques are illustrated from top to bottom, respectively.

There are a few general notes about these techniques. First, the system should

provide explicit feedback to the user about what object will be selected when these techniques are used. The system can provide this feedback by highlighting or showing the bounding box of the object that is the current candidate for selection. The user can use this feedback to confirm that he has positioned his hand correctly for a desired object before issuing a selection command. An important feature of image plane techniques is that a minimal amount of visual feedback is necessary since the relationship between the projection of target object(s) and the user's body usually indicate exactly which object will be selected. This contrasts the amount of feedback required by the laserpointer or touch techniques with which the user heavily relies on some type of feedback to know which object they are currently selecting.

These technique also provide an orientation that can be used to disambiguate the user's selection when there are a number of candidate objects with distinguishable orientations (see Section 6.1.2.8). The user's finger(s) provide this orientation for the Head Crusher, Sticky Finger, and Framing Hands techniques. The normal to the user's palm is the disambiguating orientation for the Lifting Palm technique.

6.2 Manipulation

Manipulation is a generic term which describes any of a number of ways to interactively modify the state of objects in a computer application. Manipulation in a 3D graphics context includes applying affine transformations to objects, discrete actions such as pressing buttons or complex actions like gestures or speech acts which are interpreted by a user interface as modifications of primitive objects. The key concept is that manipulation implies interactivity, and that therefore a user interface can be characterized by the types of manipulations it requires one to perform.

6.2.1 Direct vs. Indirect Manipulation

Most types of manipulation in user interfaces can be categorized as either direct or indirect. In his classic article on the subject [24], Shneiderman explains that a direct manipulation user interface is one in which the human user is presented with a visual model of a problem domain, and that the interaction dialog includes "continuous display of the object of interest" and "rapid, incremental, reversible operations whose impact on the object of interest is immediately visible." This definition was proffered in 1983, when the art of graphical user interface design was still in its infancy. It stands in stark contrast to indirect manipulation found in batch, menu or command-line interfaces which generally require users to maintain an abstract mental model of a problem that conforms to a specific specification language (e.g., the command keywords and syntax). Since this first definition of direct-manipulation, many others have presented their own versions. For example, Laurel [18] stresses that direct-manipulation interfaces present the "continuous representation of the potential for action."

Direct-manipulation interfaces, when implemented well, give users a sense of being in control of the application, and reduce the *cognitive distance* between a user's intentions and the resulting physical actions she must take. By relying more heavily on visual perception and cognition (through the use of icons and other graphical elements)

than on abstract thought processes required by text-based command interfaces, direct-manipulation interfaces can help users be more productive.

As we have experimented with different interaction techniques and widgets for 3D graphics applications, subtle variations of direct-manipulation interfaces have emerged. In its basic form, users directly control the object of interest and there are no side-effects. This type of direct-manipulation occurs in a 2D or 3D graphics application, for instance, when a user drags a shape or geometric object across the canvas or from one point in space to another. If manipulating this object has additional effects on other objects in the environment, then it is itself a component of the user interface. In this case, though the user has directly manipulated the widget, she has also *indirectly* manipulated some other part of the environment. Thus, direct-manipulation interfaces often incorporate and rely on indirect manipulation.

The widgets we use in our test application have significant direct and indirect manipulation elements. Some of the interaction techniques described earlier, however, do not provide a visual representation of themselves beside their effect on the scene. The use of predictive feedback, such as highlighting the object(s) that would be selected if the user pressed a button, for example, do provide a sense that the user is wielding a tool which can somehow modify the environment.

Designing good direct-manipulation interfaces is a tricky business, and requires deep insight into the exact nature of the tasks for which they are developed. A well-designed direct-manipulation interface can greatly help task execution, but a poorly-designed one can actually be more difficult to use than a non-graphical, indirect-manipulation interface. Thus, direct-manipulation does not necessarily equate with ease of use [15].

6.2.2 Types of Manipulation in 3D Applications

The most common types of manipulation tasks in 3D graphics applications are inherently geometric. That is, they involve changing the current transformation matrix (CTM) of 3D objects. Modeling, animation and scientific visualization applications all provide techniques for modifying the position, orientation and scale of objects, but the exact interaction techniques and widgets that one uses differ from application to application. Other attributes may also be manipulated, such as the color or transparency of an object, or higher-level attributes like the spacing of a gridded floor plane, or the number of streamlines on a rake in a scientific visualization application. However, whatever the parameter, a 3D user interface for modifying it almost always involves some kind of geometric manipulation. In the following sections, we discuss some of the widgets and techniques that we have developed for modifying parameters of 3D objects.

6.2.3 Position and Orientation Techniques

Positioning and orienting objects in 3D are two forms of manipulation that are widely used in 3D graphics applications. Many positioning and selection techniques designed for use with conventional desktop hardware (a 2D mouse and CRT display) aim to overcome many of the difficulties which result from using 2D devices for 3D

interaction tasks. We have used these same techniques as a starting point for VR interaction and found that while some are still useful, others must be (and have been) abandoned or at least significantly redesigned in order to be usable in VR. The selection techniques described above each suggest their own unique form of manipulation once an object has been selected. We next describes the manipulation techniques we have designed and implemented.

Image plane selection techniques readily transition to post-selection manipulation of objects. We identify two classes of objects a user may interact with: objects within reaching distance and objects beyond the user's reach. Virtual objects within the user's reach can naturally be directly manipulated through the touch selection technique (although the lack of haptic feedback is currently an unavoidable drawback). One of the primary motivations for using image plane techniques is to interact with objects beyond a user's arm reach. If a selected object is beyond a user's reach, there are two options for how we might implement object manipulation: it can be moved at-a-distance or moved in the user's hand after somehow producing the illusion that the object has moved to the user's hand. To move an object at a distance, we constrain the selected object to lie somewhere along a line (defined in the same way the original pick ray was; e.g., from the user's eye point through the index finger for the Sticky Finger selection technique). Moving an object at a distance typically feels like an indirect operation and is less natural than a technique in which an object seems to be in one's hand. There are several alternatives that "bring" the selected object into the user's hand. We found the most successful technique was to instantaneously scale the world around the user's eyepoint such that the selected object is moved to the user's hand. This approach has the advantage of bringing the world within reach so that the user can translate the object by directly manipulating the object's position in the scaled world. When the object is released, the world returns to original scale and position. This scaling technique is similar to the Worlds In Miniature [25] interaction techniques.

6.2.4 Adaptive Positioning Accuracy of Widgets

During a talk at the Institute for Computer Applications in Science and Engineering (ICASE) at NASA Langley, Professor John Hughes of Brown University received strong interest in a proposed novel user interface for manipulating 3D widgets. The translation of a widget is separated into two components: movement along the axis normal to a reference geometry (e.g., the space shuttle) and translation in the plane perpendicular to that same axis. The hypothesis is that users require fine control over the position of a widget when it is near the surface of a reference geometry where the most dynamic interactions of fluid flow occur. As a widget is moved further from the surface of reference geometry, less control is needed as the flow information becomes less dynamic. Thus, translation of a 3D widget is *linear* in the plane perpendicular to the normal of a reference surface and *logarithmic* along the axis normal to the reference surface. We have implemented a version of this technique and found it to be an improvement over static accuracy positioning; especially when exploring data near the shuttle's surface. A difficultly is identifying proper parameters for how much and exactly when to change the resolution with which objects are moved.

6.3 Navigation

In our virtual reality application, we exploit the built-in degrees of freedom of the BOOM or HMD to provide most of the navigation and viewpoint specification. In most cases, this suffices because the majority of the objects that we interact with in our test applications are at close range (5-10 feet away). In case we need to move beyond the somewhat limited range of the BOOM, however, we use the two buttons to "fly" forward and backward along the viewing axis. Generally, we "fly" at some constant velocity, but we have experimented with using an acceleration constant as well so that we can travel larger distances more quickly.

Image plane techniques can also be used for navigation relative to a selected object. We place a constraint on the positions of the user's hand and selected object in the 2D image so that they remain in the same position relative to each other on the image plane. For example, if the user selects an object with the sticky finger technique, the constraint will keep the object under the user's index finger in the 2D image. Because the position of the selected object is held constant in the 3D scene, we maintain this constraint by translating the user to some position along the vector from the selected object through the user's index finger. This technique is similar to Orbital viewing [17]. However, in their implementation, the user is required to move his head to orbit the selected object. In our implementation, the user orbits an object by moving their head and hand at the same time. Unlike orbital viewing, our implementation permits the user to freely look around while orbiting an object. The distance of the user from the selected object can remain the same as it was when the object was selected or can dynamically be changed. We have used a linear function that moves the user closer or further to the selected object when the user moves her hand closer or further, respectively, from their viewpoint. In this implementation, if the user moves her hand half the distance to her eye-point, she moves to half the original distance to the selected object.

6.4 Rapid Prototyping of Input Devices

In addition to developing new software techniques for interacting in immersive virtual environments, we have developed a rapid prototyping system for physical interaction devices. Because of the increased complexity of 3D interactive environments and the lack of standard interactive tools, designers are unable to use traditional 2D hardware in 3D virtual environments. As a result, designers must create entirely new interaction devices, a slow and expensive process. Designers must choose between non-functional device mock-ups, and functional prototypes which are not easily modified. To fully understand the usefulness of a given design and advance the technology of virtual reality interfaces, designers must be able to quickly prototype malleable, functional input devices. The Lego Interface Toolkit [2] allows designers to experiment with the construction of new 3D interaction devices both quickly and inexpensively.

We tested our toolkit in our implementation of the virtual windtunnel. The virtual visualization tool used was a rake. Traditionally, the user controls the position and orientation of the rake with a hand-held 6-DOF cursor and a push-button. With the other hand, the user controls her position with a BOOM viewing device. The user varies the parameters of the rake widget (length, number of streamlines, position,

orientation) using the 6 DOF tracker and an interaction technique like those described in Section 6.1 and Section 6.2. The user grabs the rake, positions and orients it, releases it, adjusts the size of the rake and density of the streamlines using geometry widgets and then re-grabs the rake to continue exploration. Because of the noise involved in 3D devices, this can be inconvenient and the rake can be difficult to reacquire once it is released.

One alternative to this traditional approach is procedure is to add physical dials or sliders that are dedicated to controlling the various aspects of the rake widget to another physical hand held object representing the rake frame. This allows modification of the rake's parameters while maintaining control of the rake's position and orientation. Given this hardware, the designer must decide how many controls are appropriate and where to place them. Although some of the ergonomics of various arrangements can be user-tested without being hooked into the system, functional models provide the designer and user with a more complete picture of the strengths and weaknesses of a design. To illustrate the control dilemma, we prototyped several devices which allow the user to manipulate the parameters of the rake without releasing it (see Figure 8).

We have found that although the prototype widgets allow quick assembly of interaction hardware, they are not suitable for prolonged use. The devices are sturdy, but not as sturdy as machined or molded parts. In addition, the input is not precise enough for most applications. The toolkit did, however, allow us to get a general idea of which sorts of devices are usable while assisting in the rapid prototyping process.

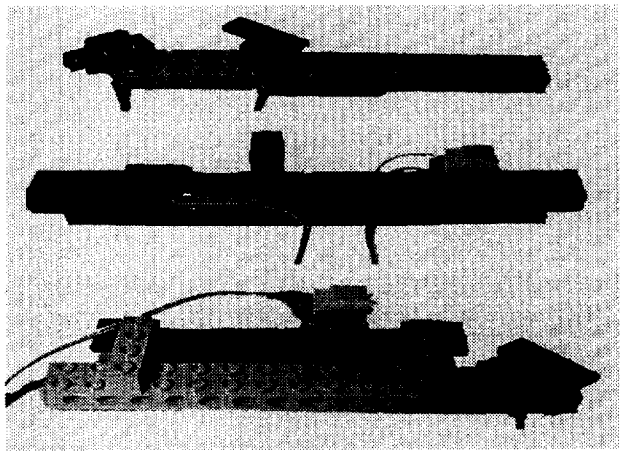

Figure 8. A hand held rake widget. Various devices prototyped using the Lego Interface Toolkit. The top device uses the two dials to control rake length and streamline density. The middle uses a single slider for both functions and a button to chose between them. The bottom widget uses a slider to control length, a dial for density and a button to retain the 3D cursor functionality of the original widget.

7 Evaluation

Numerous guidelines for user interface design have been proposed and implemented for *windows, icons, menus* and *pointers* (WIMP) interfaces on desktop

computers, including those for Macintosh, Windows, X and others. This style of interface is the accepted standard for most desktop operating systems and applications today. In contrast, the discipline of user interface design for interactive 3D graphics applications is still young compared to 2D UI design. Although many interaction techniques have been implemented and used in interactive 3D graphics applications, little is known about which techniques are successful and why. This is partly because very few formal studies have been performed to test these interaction techniques [13][14].

Since our system facilitates rapid prototyping of user interfaces, the interaction techniques and widgets that we have implemented for this project were produced over multiple iterations of design, implementation and evaluation. Most often, the evaluation phase consisted merely of reviews by members of the research team of incremental changes to the interface. We did perform a set of informal pilot tests and user studies, however, over the course of this grant which focused on evaluating specific interfaces. In these studies, we attempted to determine which of a variety of possible designs was qualitatively best for the users as well as how difficult it was for users to learn to use a technique.

7.1 Image Plane Techniques

We have performed informal tests with ten users to determine whether or not people have problems using the image plane interaction techniques. No user has had any trouble understanding how the techniques work. Every user has been able to select and manipulate objects and to navigate around the scene. Although our informal tests have been positive, we feel the need for more definitive user studies to determine how well these new techniques work in relation to previous techniques like laser pointing and spotlight selection.

Bowman et al. have presented a user study comparing pointing and gaze directed navigation techniques for navigating in immersive virtual environments [3]. They studied how effective each technique was for translating to a target object (absolute navigation) and around a target object (relative navigation). They found gaze directed navigation was most effective for absolute navigation and pointing navigation was best for relative navigation. We note that our image plane navigation techniques combine the best aspects of both techniques. For instance, to perform absolute navigation to a target object using the sticky finger selection technique, a user would move their index finger over the target object, signal to the computer to select, move their finger to their eye (thereby navigating all the way to the object), and then signal to the computer to end the navigation. An example of a relative navigation task is to move in front of a bookshelf. Using image plane techniques, a user can select the bookshelf, then in combinations of head and hand movements quickly move to a position some distance in front of the bookshelf.

8 Future Work

We intend to continue our investigation of user interfaces for IVE's by continuing to refine our existing interfaces through user testing and by experimenting with new

interface techniques that push the state of the art. There are a number of key areas that we will focus on:

- Composite Interaction Techniques
- User Studies and Pilot Studies
- Shared Environments
- Dynamic Transitions between Desktop and Immersive VR Systems
- Interaction in Larger Environments

We discuss each of these items further below.

8.1 Composite Interaction Techniques

We now have a suite of basic interaction techniques. The next step towards more effective systems is to explore how these techniques work together. We are currently examining all the techniques we have developed in isolation; there is no means to dynamically switch between techniques. Therefore, there is room to experiment with how well these techniques work together, and how they can be incorporated into a system. Mark Mine at UNC and Carlo Sequin at Berkeley have been working on understanding how immersive VR techniques work together for modeling task, and how the users can dynamically indicate which interaction technique they want to use [21].

8.2 User Studies and Pilot Studies

The studies we have done to date have proven very useful in determining the success or failure of specific interface designs. We believe that formal usability testing for the image plane techniques is required. Although we have performed an informal evaluation, we need more rigorous testing to determine how the speed and accuracy of these techniques compare to more established techniques like laser pointing and spotlight selection.

8.3 Shared Environments

The appearance of interaction techniques to other users in a shared virtual environment is another open question for investigation. If the user selects an object, we must decide how this appears to an observer standing nearby or at a distance. Possibilities include showing the object floating to the user's hand. We have published one paper on this topic [8] and are investigating it further.

8.4 Dynamic Transitions between Desktop and Immersive VR Systems

We envision a system that allows users to freely switch between desktop and immersive VR depending on which type of environment is most appropriate for the task the user wishes to perform. We are currently exploring this type of system using an Active Desk, LCD shutter glasses for stereo viewing, and six degree-of-freedom tracking devices in conjunction with a workstation desktop environment for a

modeling task. Using the Sketch system [27] a user creates geometry at the desktop, but, when appropriate, can transition into an immersive environment consisting of the same model.

8.5 Interaction in Larger Environments

The techniques we have been working on are best suited for relatively small virtual environments; that is, environments in which objects are within eye sight and, in particular, environments where objects are distinguishable without navigating. There is room for exploring interaction techniques for much larger environments.

9 Publications and Reports Related to this Grant

- Ayers, M.R., and Zeleznik R.C., The Lego Interface Toolkit. Computer Graphics (Proceedings of the ACM Symposium on User Interface and Software Technology (UIST)), '96, pp. 97-98.
- Forsberg, A., Herndon, K.P. and Zeleznik, R. Effective Techniques for Selecting and Direct-Manipulation of Local Objects in Immersive Virtual Environments. Computer Graphics (Proceedings of the ACM Symposium on User Interface and Software Technology (UIST)), '96, pp. 95-96.
- Pierce, J.S., Forsberg, A., Conway, M.J., Hong, S. P., Zeleznik, R.C., and Mine, M. Image Plane Interaction Techniques in 3D Immersive Environments. Proceedings of 1997 Symposium on Interactive 3D Graphics, (Providence, Rhode Island, April 27-30, 1997).

10 Brown Personnel

The Brown Graphics Group, directed by Professors Andries van Dam and John F. Hughes, is a team of Ph.D., Masters, and undergraduate students and full-time staff. Professor van Dam, the principal investigator of this research, is also currently the Director of the NSF Science and Technology Center for Computer Graphics and Scientific Visualization. He and John Hughes are co-authors of the standard computer graphics textbook, *Computer Graphics, Principles and Practice*, along with James Foley and Brown Ph.D. Steven Feiner. Van Dam is a co-founder of ACM SIGGRAPH and co-founder and first chairman of Brown University's Computer Science Department. The full-time staff of the Graphics Group includes the Director of Research (Bob Zeleznik), a Research Scientist (Timothy Miller), a User Interface Developer (Andrew Forsberg), an Educational Outreach Director (Anne Morgan Spalter), and a Software Engineer/ Researcher (Loring Holden). A number of graduate and undergraduate students complement the group by assisting on various research projects. The Media Coordinator (Mark Oribello) and three part-time students support computers, video-teleconferencing, and the group's other AV equipment. Andrew Forsberg was the principle researcher being funded by this grant.

11 Facilities and Equipment at Brown

The facilities at Brown include a variety of workstations from HP, DEC, Sun and SGI. Our Virtual Reality Lab contains a Fakespace Labs BOOM, a Virtuality Visette Pro Head-Mounted Display, a Virtual Technologies CyberGlove, and an Ascension extended-range Bird tracker. We also use a StereoGraphics VR setup (LCD-shutter glasses and two Logitech 3D mice). An Active Desk built by Input Technologies, Inc. (ITI) was recently donated to our lab by Alias/Wavefront. We have two Phantom haptic feedback devices made by Sensable Technologies. We also have a teleconference system which uses a dedicated T1 line to connect us to the four other sites of the NSF STC Center for Computer Graphics and Scientific Visualization. A full audio/video non-linear editing system is used to record footage directly from workstation screens and to edit videotapes.

We also maintain a World Wide Web site which contains general information about our group and research projects:

`(http://www.cs.brown.edu/research/graphics/)`

12 Acknowledgments

The groups is sponsored by the following; the NSF Science and Technology Center for Computer Graphics and Scientific Visualization, NASA, Sun Microsystems, Taco, Microsoft, and SGI.

13 References

[1] AVS Software application.

[2] Ayers, M.R., and Zeleznik R.C., The Lego Interface Toolkit. Computer Graphics (Proceedings of the ACM Symposium on User Interface and Software Technology (UIST)), '96, pp. 97-98.

[3] Bowman, D. A., Koller, D., Hodges, L.F., Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques. In proceedings of Virtual Reality Annual International Symposium (VRAIS) '97, pp. 45-52, Albuquerque, NM, March 1-5, 1997.

[4] Bryson, S. Measurement and calibration of static distortion of position data from 3D trackers. NASA/Ames Technical Report RNR-92-011, March 1992.

[5] Bryson, S. and Levit, C. The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows. In *Proceedings of Visualization'91*, pages 17–24.

[6] Cohen, M., Lin, D.M. and Ponamgi, K. I-COLLIDE: An interactive and exact collision detection system for large-scaled environments. In *Proceedings of ACM Int. 3D Graphics Conference*, pages 189–196, 1995.

[7] Conner, D.B., Snibbe, S.S., Herndon, K.P., Robbins, D.C., Zeleznik, R.C. and van Dam, A., Three-dimensional widgets. *Computer Graphics (Proceedings of the 1992 Symposium on Interactive 3D Graphics)*, Vol. 25, No. 2, ACM SIGGRAPH, March,

1992, pages 183–188.
[8] Conner, D.B. and Holden, L.S., Providing a Low-Latency User Experience in a High-Latency Application. In *Proceedings of 1997 Symposium on Interactive 3D Graphics* (Providence, Rhode Island, April 27-30, 1997).
[9] FAST! Software application.
[10] Forsberg, A., Herndon, K.P. and Zeleznik, R. Effective Techniques for Selecting and Direct-Manipulation of Local Objects in Immersive Virtual Environments. Computer Graphics (Proceedings of the ACM Symposium on User Interface and Software Technology (UIST)), '96, pp. 95-96.
[11] Ghazisaedy, M., Adamczyk, D., Sandin, D., Kenyon, R. and DeFanti, T. Ultrasonic calibration of a magnetic tracker in a virtual reality space. In *Proceedings of the IEEE Annual Virtual Reality International Symposium (VRAIS)*, pages 179–188, 1995.
[12] Herndon, K.P., Hughes, J. and Forsberg, A. Compensating for static distortions in electromagnetic tracking systems. To be submitted for publication in *The Journal of Graphics Tools*.
[13] Hinckley, K., Pausch, R, Goble, J. and Kassell, N. Passive Real-World Interface Props for Neurosurgical Visualization. *ACM CHI'94 Conference on Human Factors in Computing Systems*, pages 452–458., 1994.
[14] Houde, S. Iterative design of an interface for easy 3D direct manipulation. In *Proceedings of CHI'92 Human Factors in Computing Systems*, pages 135–142, May, 1992.
[15] Hutchins, E.L., Hollan, J.D. and Norman, D.A. Direct manipulation interfaces. In: User Centered System Design (Norman, D.A. and Draper, S.W., Eds). Lawrence Erlbaum Associates, Inc., 1986.
[16] IBM Explorer, Software application.
[17] Koller, D., Mine, M., and Hudson, S. Head-Tracked Orbital Viewing: An Interaction Technique for Immersive Virtual Environments. *Proceedings of UIST '96*, pp. 81-82, November 1996.
[18] Laurel, B. Computers as Theatre. Addison-Wesley Publishing Company, 1993.
[19] Liang, J. and Green, M. JDCAD: A highly interactive 3D modeling system. *Computers & Graphics* (Pergamon), Vol. 18, No. 4, pages 499–506, July/Aug. 1994.
[20] Liu, A., Tharp, G., French, L., Lai, S. and Lawrence Stark. Some of What One Needs to Know About Using Head-Mounted Displays to Improve Teleoperator Performance. In *IEEE Transactions on Robotics and Automation*, Vol. 9, No. 5, pages 638–648, 1993.
[21] Mine, M., Brooks, F. P., and Sequin, C. Moving Objects in Space: Exploiting Proprioception in Virtual-Environment Interaction. To appear in proceedings of SIGGRAPH 97, Los Angeles, CA, 1997.
[22] Pierce, J.S., Forsberg, A., Conway, M.J., Hong, S. P., Zeleznik, R.C., and Mine, M. Image Plane Interaction Techniques in 3D Immersive Environments. Proceedings of 1997 Symposium on Interactive 3D Graphics, (Providence, Rhode Island, April 27-30, 1997).
[23] Robinett, W. and Holloway, R. Implementation of flying, scaling, and grabbing in virtual worlds. *Computer Graphics (Proceedings of the 1992 Symposium on Interactive 3D Graphics)*, Vol. 25, No. 2, March, 1992, pages 189–192.
[24] Shneiderman, B. Direct Manipulation: A Step Beyond Programming Languages.

IEEE Computer 16, 8 (Aug. 1983) 57-69.

[25] Stoakley, R., Conway, M.J. and Pausch, R. Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings of ACM SIGCHI'95*, 1995.

[26] Zeleznik, R.C., Conner D.B., Wloka, M.M., Aliaga, D.G., Huang, N.T., Hubbard, P.M., Knep, B., Kaufman, H., Hughes, J.F. and van Dam, A. An object-oriented framework for the integration of interactive animation techniques. In *Computer Graphics (Proceedings of SIGGRAPH'91)*, Vol. 25, No. 4, pages 105-112, July 1991.

[27] Zeleznik, R.C., Herndon, K.P. and Hughes, J.F. Sketch: A system for rapidly constructing 3D models. To be published in *Computer Graphics (Proceedings of SIGGRAPH'96)*, July 1996.