

BROWN UNIVERSITY
Department of Computer Science
Master's Project
CS-96-M9

“An Implementation of 6-DOF-Based
Direct-Manipulation Techniques for
Immersive Virtual Environments”

by

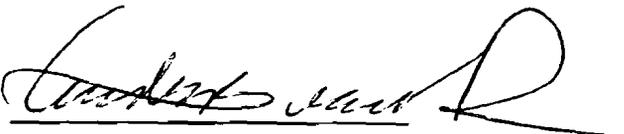
Andrew Stephen Forsberg

An Implementation of 6-DOF-Based Direct-Manipulation Techniques for Immersive Virtual Environments

Andrew Stephen Forsberg

Department of Computer Science
Brown University

Submitted in partial fulfillment of the requirements for the Degree of Master
of Science in the Department of Computer Science at Brown University.

5/17/96 

Professor Andries van Dam

Advisor

May 17, 1996

An Implementation of 6-DOF-Based Direct-Manipulation Techniques for Immersive Virtual Environments

Andrew Stephen Forsberg

Department of Computer Science
Brown University

Contents

1	Introduction	1
1.1	NASA Scientific Visualization	1
1.1.1	Background	1
1.1.2	Why move to immersive VR?	1
1.1.3	Why focus on direct-manipulation?	2
1.1.4	Why focus on 6-DOF-based input devices?	3
1.2	Description of VR lab.	4
1.3	UGA and VR	4
1.3.1	What UGA provides	4
1.3.2	What UGA lacks	5
2	Previous Work	6
2.1	Introduction	6
2.2	Packages	7
2.2.1	WIM	8
2.2.2	ISAAC	8
2.2.3	Conceptual Design Space	8
2.2.4	Silk Cursors	8
2.2.5	JDCAD	9
3	Novel Direct-Manipulation Techniques	10
3.1	Introduction	10
3.2	Aperture selection	10
3.3	Aperture with orientation	13
4	VR User Interface Package for UGA	16
4.1	Overview	16
4.2	Definitions	18
4.3	Implementation notes	18
4.3.1	Visual feedback	19
4.3.1.1	Pre-select feedback	19
4.3.1.2	Guidelines	19
4.3.1.3	Transparency	19
4.3.1.4	Shadows and depth cues	19
4.3.2	Touch	20
4.3.3	Approaches to volume intersection tests	20
4.3.3.1	Sampling the selection cone	21
4.3.3.2	Vertex inclusion tests	22
4.3.3.3	Projected edges	23
4.3.3.4	I-COLLIDE	23
4.4	C++ class descriptions	24
4.4.1	Classes related to input devices and events	24

4.4.1.1	VRtracked_device	24
4.4.1.2	VRhandler	24
4.4.2	Interaction Techniques	25
4.4.2.1	VRinteraction_technique	25
4.4.2.3	VRselection_technique	25
4.4.2.4	VRtransform_technique	25
4.4.2.2	VRdirect_manipulation_technique	25
4.4.3	Select test classes	25
4.4.3.1	VRselect_test.....	25
4.4.4	Examples of direct-manipulation techniques	26
4.4.4.1	VRtouch_select	26
4.4.4.2	VRray_laserbeam	26
4.4.4.3	VRray_from_eye	27
4.4.4.4	VRaperture_flashlight	27
4.4.4.5	VRaperture_from_eye.....	27
4.4.4.6	VRorientation_select.....	27
4.4.4.7	VRaperture_orientation	28
5	Future Work	29
6	Project Statistics	30
7	Acknowledgments	31
8	Bibliography	32

List of Figures

1	Two rake widgets emitting streamlines over a shuttle model are shown. Translation of the light blue disk along the rake frame (the long grey cylinder) changes the number of streamlines emitted from the rake frame. Translation of the red sphere changes the length of the rake frame.	2
2	The stick prop's cursor representation is shown.	4
3	a) The flashlight selection cone emitted from a point in space. Objects A and C are candidates for selection, object B is not. If single object selection is desired, the disambiguation metric is used to determine whether object A or C is selected. b) The flashlight selection technique and the stick prop cursor (the cone in the foreground) are shown. Note the transparent selection cone and bounding-box visual feedback indicating the parallelepiped has been selected.	7
4	a) The conic volume of the aperture selection technique is defined by the eye point, aperture cursor geometry, and distance d_1 between the eye point and aperture cursor. b) A larger volume can be defined by translating the aperture cursor closer to the eye point, thereby reducing the distance d_1 to d_2	11
5	Using the drumstick prop in an immersive virtual environment.	11
6	The sphere is selected with the aperture selection technique when the aperture circle at the tip of the virtual stick is positioned over it. The cone-shaped object entering the scene from the right of the image is the cursor representation of the stick prop in the IVE.	12
7	The orientations of object A and the plane P are tested by comparing the angle between P's normal and A's longitudinal axis	13
8	The aperture technique. a) The conic volume described by the viewpoint and aperture. b) The conic volume described in the "flashlight" configuration. Both a and b are shown with the drumstick VID. In our implementation, the conic volume is semi-infinite.	14
9	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.	14
10	The UGA model for translating input device data to scene modifications.	16
11	The VRUI model for translating input device events to scene modifications through an interaction technique.	17
12	The VRUI model for translating input device events to scene modifications through a direct-manipulation interaction technique.	18
13	A cross-section of an aperture cone is shown. a) Sample points through which rays are "shot." b) A situation in which this approach fails: the shaded triangle is not intersected by any of the sample rays and is not selected, although it lies inside the aperture cone.	21

14	Rays are shot through the thirteen sample points in a cross-section of the aperture cone.	22
15	a) The vertex inclusion test compares whether , the angle between a vertex and the centerline of the selection cone, is less than , the maximum angle subtended by the selection cone. b) A situation in which the vertex test fails— all vertices are outside the selection cone but the triangle face intersects the cone.	22
16	a) One edge of the shaded triangle intersects the selection cone cross-section. b) None of the edges of the shaded pentagon intersect the selection cone cross-section. However, the pentagon does intersect the selection cone and therefore should be selected.	23

Abstract

Over the past four years, the Brown Graphics Group has been researching novel user-interface techniques for scientific visualization. Recent work has focused on user-interface issues for immersive virtual environments (IVEs). Prior to this work, the Brown Graphics Group had facilities for viewing and basic navigation in IVEs, but did not have any means for user interaction.

A framework for the design of interaction techniques and an implementation of several direct-manipulation techniques for use in IVEs is presented. The implementation has been designed so that the techniques may be used with many classes of 3D input devices. Two novel techniques for selecting objects in IVEs that overcome several shortcomings of existing work are introduced. Collision detection, its utility in IVEs, and its use in this work are also discussed.

1 Introduction

1.1 NASA Scientific Visualization

1.1.1 Background

NASA engineers must evaluate their space shuttle designs. One means for testing a design is to place a model of a space shuttle in a wind tunnel and study the flow of streamlines over it. Drawbacks of this technique are that engineers have limited control over the experiment and are constrained in the ways the data produced can be analyzed, and building the model and running the experiment can also be expensive and time-consuming.

Virtual reality (VR) is a tool available to NASA for simulation and analysis. Steve Bryson has been a pioneer in the use of virtual reality for scientific exploration [1], and the application for which the techniques presented in this paper were implemented is derived from his Virtual Wind Tunnel project.

When studying shuttle design in VR, the first step is to create a computer model of the shuttle. The model is input into an application that has been programmed to simulate the dynamics of air flow over an object. The output from this application is a data set describing the airflow around the shuttle. An approximation of the airflow information obtained in the real-world wind tunnel is now available for random access by the computer. VR is a tool for exploring the airflow information due to the 3D nature of the task.

Immersive VR (IVR) is a subset of virtual reality in which the user is more intimately connected to the virtual world. Ideally, the user feels she is *in* the virtual environment viewing and interacting with objects around herself. Displays that encompass a user's view and trackers that sense the position and orientation of a user's viewpoint and hands are tools for implementing IVR. The term *immersive virtual environment* (IVE) is used in this document to refer to the virtual world the user is immersed in when using IVR. *Fishtank VR* is a term for a VR system with IVR-style input devices but without a display that encompasses a user's field of view.

Before IVR becomes an accepted tool for scientific visualization, however, several problems must be resolved. These include improving the user interface to IVR applications. The purpose of The Brown University Graphics Group's research in this area is to provide better user interfaces for scientific visualization applications.

1.1.2 Why move to immersive VR?

Previous work in the Graphics Group [9][13][14][15] has been developed on the desktop. The scientific visualization application that motivated the work

presented in this paper has been ported from the desktop to IVR to explore how the user interface to a scientific visualization application might benefit from *immersive stereo display* and *six-degree-of-freedom* (6-DOF) spatial input devices, which provide 3D position and 3D orientation. An immersive stereo display is a display device which provides each eye with a full computer generated view. This work provides a means to begin exploring how richer input and display devices can improve the graphical user interface of applications.

1.1.3 Why focus on direct-manipulation?

Direct-manipulation techniques provide a way to interact with components of a 3D widget. The 3D widget [3], a concept the Brown Graphics Group has been pursuing for use in scientific visualization applications, is a mixture of geometry and behavior. Widgets often have parameters that are accessed by manipulating their components. For example, the rake widget in the virtual windtunnel application has a slider attached to its frame that controls the number of streamlines emitted from the frame (see Figure 1).

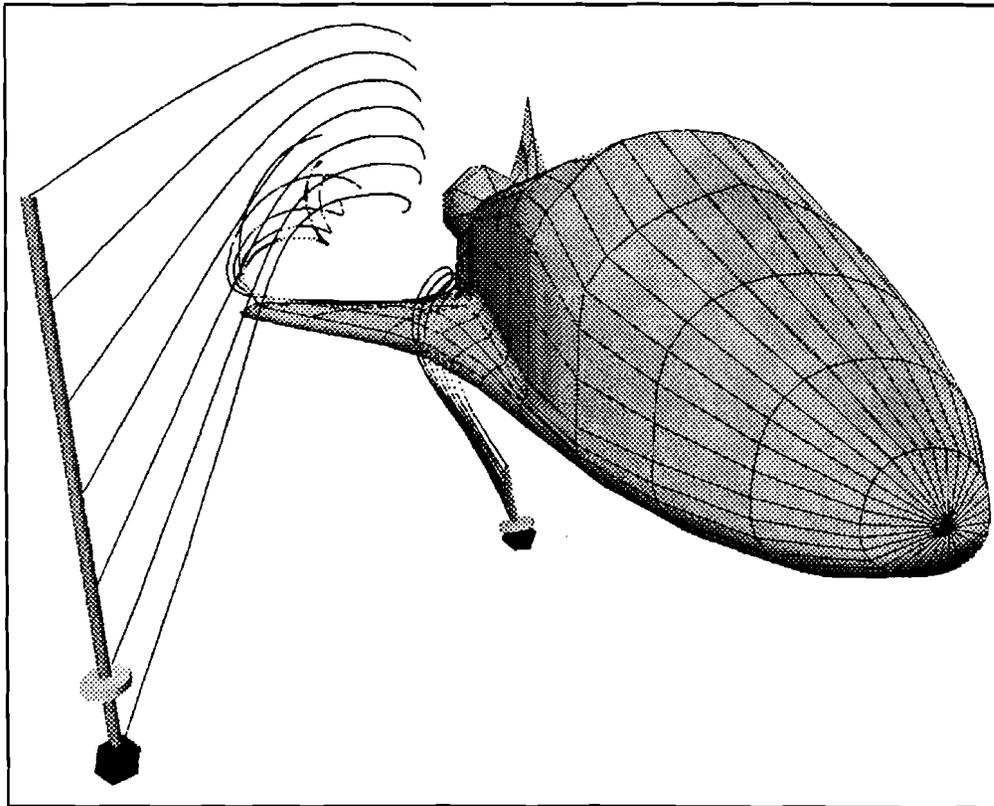


FIGURE 1. Two rake widgets emitting streamlines over a shuttle model are shown. Translation of the light blue disk along the rake frame (the long grey cylinder) changes the number of streamlines emitted from the rake frame. Translation of the red sphere changes the length of the rake frame.

Shneiderman [17] defines a direct-manipulation user interface as one in which there is a “continuous display of the object of interest” and “rapid, incremen-

tal, reversible operations whose impact on the object of interest is immediately visible.” Direct-manipulation techniques mimic the way humans interact with objects in the real world. We select objects and then manipulate them. Throughout the process we are in contact with the object, either directly or through some extension of our bodies such as a tool. An important attribute of direct-manipulation is that it takes place in real time: we immediately sense the state of the world and the effects of our actions.

My implementation provides techniques for translation and rotation of objects only, though the framework presented can be used for other types of direct-manipulation. Translation and rotation are a starting point for interaction in VR— they are elementary transformations and can be used to manipulate components of more complex widgets to perform more complex transformations [19].

1.1.4 Why focus on 6-DOF-based input devices?

6-DOF input devices report the three-dimensional position and orientation of some point or object in space. The minimal information required to specify completely the spatial state of a rigid object in space is position and orientation. A 6-DOF input device is therefore the “base case” for spatial input— it is general and can be obtained from all 6D spatial input devices regardless of their primary function.

Each interaction technique presented in this document requires input from a 6-DOF device— a 3D position and orientation. This spatial information is used in each technique in conjunction with a signal (e.g., a button press) to select and manipulate objects in an IVE.

Our *stick prop* and Virtual Technologies Cyber Glove are examples of devices that can provide 6-DOF position and orientation information. The stick prop is constructed from an Ascension 6-DOF tracker attached to a drumstick. The position and orientation of the tip of the stick prop or parts of the glove can be queried. A direct-manipulation technique such as the flashlight technique (Section 2) may be used with any input device that can supply 3D position and orientation information and a signal. When used with a stick prop, the flashlight selection cone is emitted from the tip of the stick; when used with a glove, the flashlight selection cone is emitted from one of the user’s fingertips. In both cases, a graphical representation of the input device is drawn in the IVE. This graphical representation is called a *cursor*. A cone the same length

as the physical drum stick is used as the cursor for the stick prop in this implementation (see Figure 2). This 1:1 size match between the real-

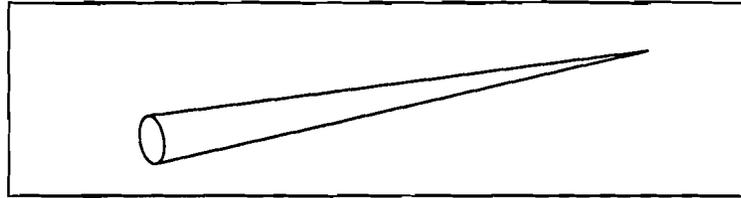


FIGURE 2. The stick prop's cursor representation is shown.

world drum stick and the stick prop cursor was chosen to increase the level of immersion for the user.

1.2 Description of VR lab

This work was designed for use with the following tools:

- A 6-DOF input device is required.
- A method for signaling to the computer such as a button press, glove gesture, or voice command is required.
- A display device. The techniques are intended for use in IVEs, but could be used at the desktop in a fishtank VR setup. If an IVE display device is used, the 3D position and orientation of the user's viewpoint must be tracked to render displays from the correct point of view.

An SGI Onyx Reality Engine 2 was used as the CPU for this work. The stick prop described above provided 6-DOF position and orientation input and a button was used for signaling. A Fakespace BOOM provided an immersive, stereo display and tracked the user's head movements.

1.3 UGA and VR

1.3.1 What UGA provides

Brown's Unified Graphics Architecture (UGA) [23] is intended to provide a flexible framework for the design of interactive, dynamic 3D environments. The system is constantly evolving as new packages are implemented. The primary functions the system supports are:

- object database management
- a main event loop
- rendering hardware

- low-level devices (e.g., for mice, tablets, and dial boxes)
- snap-together math (STM) [6] package for symbolic expressions
- basic utilities such as ray intersection within the environment
- primitive objects (cube, sphere, cone, cylinder)
- animation utilities
- CSG operations
- direct-manipulation techniques for use with a mouse or tablet

The standard input devices used with UGA are the keyboard and mouse, and the standard output device is a high-resolution workstation monitor.

The implementation of UGA includes a set of standard interaction techniques used in conjunction with a mouse for translation and rotation of objects in a 3D environment. These techniques are direct-manipulation techniques: when the user positions the cursor over an object and depresses the middle button, the object is highlighted and subsequent mouse movements result in translation of the object such that the original point selected remains under the cursor. Releasing the mouse button deselects the object. The virtual sphere technique [18] is provided for rotating objects.

1.3.2 What UGA lacks

Until recently, UGA has been used primarily for desktop applications. An exception is Wloka and Greenfield's *Virtual Tricorder* [22], which used a Logitech Flymouse (TM) for fishtank and immersive VR.

Much of the functionality provided by UGA extends readily to use in an IVE. The direct-manipulation techniques intended for use with a mouse or tablet, however, do not because they expect 2D input and most IVE input devices provide at least 6-DOF. It is, of course, possible to obtain 2D information from a 6D device, but doing so in a meaningful way may be difficult and discards the unique information a 6D device offers.

2 Previous Work

2.1 Introduction

A variety of 6-DOF-based direct-manipulation techniques have been developed for selecting objects in virtual reality applications. The most common are the metaphorical *touch* and *laser pointer* [10]. The touch technique allows the participant to place a 3D cursor (representing the tracked hand) on or inside a target object in the VE. The object is typically highlighted when successfully touched. A gesture (e.g., with a glove input device) or button click signals the application to select the highlighted object. The laser-pointer technique utilizes standard ray intersection to determine which object(s) to select, and is attractive due to its low computational cost and ease of implementation.

In practice, however, the effectiveness of these techniques is limited by imprecision and noise in the tracking system as well as instability of the participant's hand. Evaluation studies of the laser pointer technique find there is typically ± 5 degrees of rotational noise when a participant tries to hold the tracker steady and aim at a target object. Consequently, small objects even a short distance away are difficult to select reliably or consistently when using the laser pointer technique. Similarly, positional noise (± 1.5 inches) adversely affects the usability of the touch technique. Also, the touch technique can be used only to select objects that are within reach.

The smaller an object, the more difficult it is to select on the basis of its appearance. The difficulty of selecting a small object is directly related to that of selecting a distant object in an IVE, since under a perspective viewing projection an object's size is inversely proportional to its distance from the viewer.

The *flashlight* (also called *spotlight*) technique [11] is a variation of the laser pointer technique that reduces the effects of noise and object-size problems by using a conic selection volume. The flashlight technique requires sufficient visual feedback for effective use; in particular, feedback to show the volume of the flashlight cone is necessary. Because all the objects within the conic selection volume may be selected, a *disambiguation metric* for choosing a single object from the set of candidates may be required. Two choices for this metric are the distance between an object and the selection cone's apex or the shortest distance between the object and the centerline of the conic selection volume.

Figure 3 illustrates the flashlight selection technique. Objects *A* and *C* are candidates for selection, object *B* is not. When the disambiguation metric is defined to be the shortest distance between an object and the centerline of the selection cone, object *A* is chosen. When the distance between an object and the apex of the selection cone is used as the disambiguation metric, object *C* is

chosen. The nearest-to-centerline disambiguation metric has been most preferred by most users.

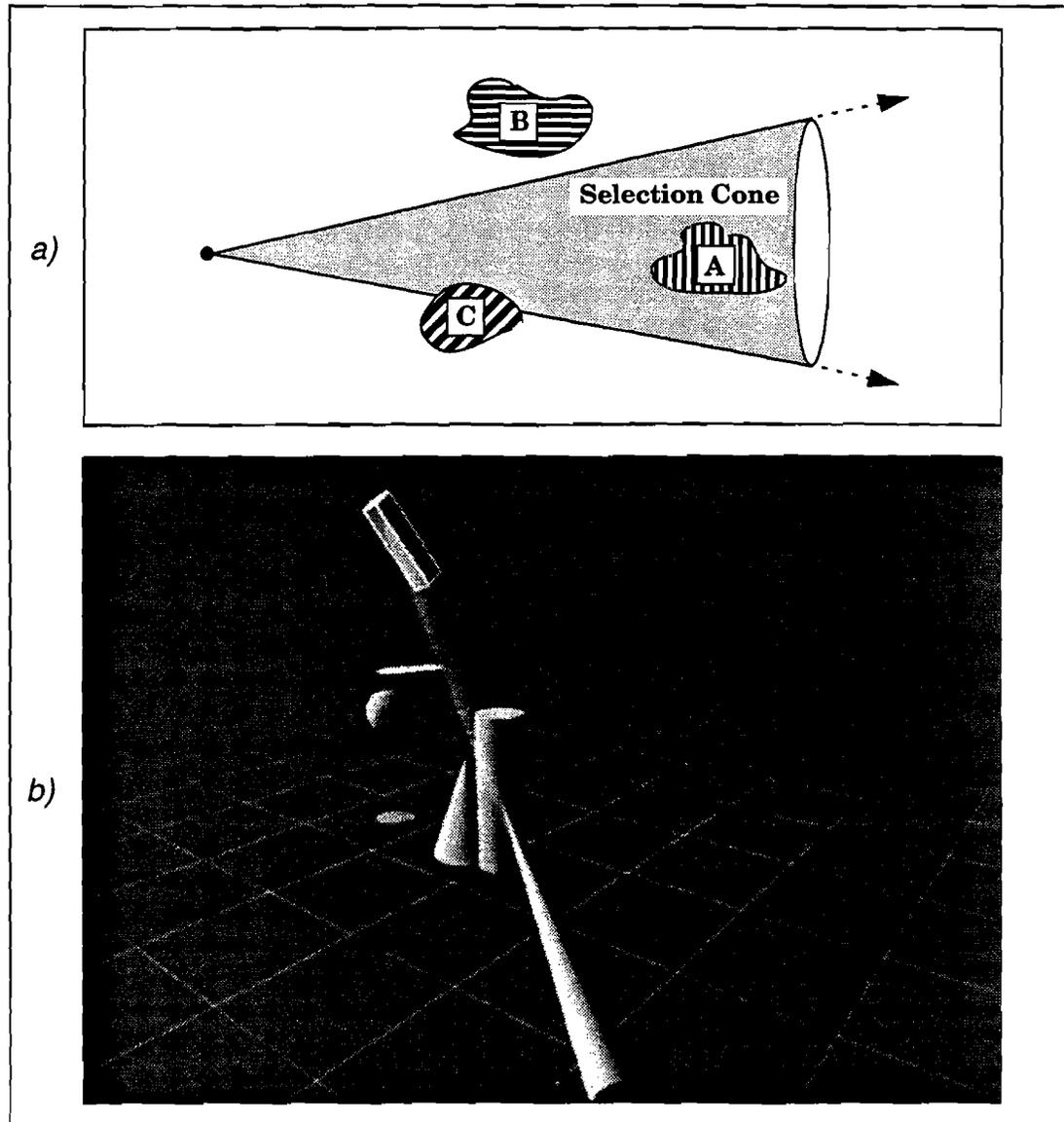


FIGURE 3. a) The flashlight selection cone emitted from a point in space. Objects A and C are candidates for selection, object B is not. If single object selection is desired, the *disambiguation metric* is used to determine whether object A or C is selected. b) The flashlight selection technique and the stick prop cursor (the cone in the foreground) are shown. Note the transparent selection cone and bounding-box visual feedback indicating the parallelepiped has been selected.

2.2 Packages

The following sections describe the strengths and weaknesses of existing implementations of 6-DOF-based direct-manipulation techniques.

2.2.1 WIM

Worlds In Miniature (WIM) [20] is a user interface technique that augments an IVE with a computer-generated miniature copy of the virtual world. A tracked object the user holds in the real world is represented in the IVE by the WIM. Objects in the WIM can be manipulated, so that a second means is available for interacting with each object in the world.

Stoakley's implementation of the WIM used the touch technique for direct-manipulation of objects. WIM's major advantages are that objects out of reach can be manipulated through their miniature counterparts and that occlusion and line of sight problems are reduced. This technique does not facilitate manipulation of small objects, however, because they are scaled down even further in the WIM.

2.2.2 ISAAC

The Immersive Simulation Animation And Construction (ISAAC) [16] system is a testbed for virtual environment interaction techniques. ISAAC incorporates the following interaction techniques: action at a distance, worlds in miniature, constrained motion widgets, rotary tool chooser, and two-dimensional menu systems. The laser pointer technique is used for direct-manipulation of distant objects, of objects in the WIM, and for interaction with menu widgets.

While the laser pointer technique allows for interaction at a distance, it may be difficult to select objects that are small or far away. Widget components and menu buttons must also be large enough or close enough to the viewer for the laser pointer technique to work effectively.

2.2.3 Conceptual Design Space

The Conceptual Design Space (CDS) system developed at Georgia Institute of Technology is an interactive virtual environment application that addresses 3D immersive design. CDS supports 3D menus, 3D widgets for object manipulation, dialog boxes, slider widgets, and tool palettes. The laser pointer technique is used to select and interact with objects and widget components. This system shares the same weaknesses as the ISAAC system.

2.2.4 Silk Cursors

Zhai et al. found that a transparent "area" cursor facilitated the selection of small objects and points in 2D applications and speculated that a volume cursor would result in similar improvements in 3D applications [24]. The word "silk" describes the way the cursor's transparency provides depth cues when it is in front of, inside, or behind other objects. The use of volume selection and transparency to provide depth information are similar to the ideas from JDCAD below.

2.2.5 JDCAD

JDCAD [11] is an experimental CAD application developed for fishtank VR use. A 6-DOF tracker with buttons, the keyboard, and a mouse are used to interact with the application.

JDCAD extends the laser pointer selection technique to the flashlight technique by testing for object inclusion in a volume instead of intersection with a ray. This extension was motivated by the need for a “softer” selection technique. The disadvantages of this technique are the complexity and cost of performing volume intersections and the need for a disambiguation metric if multiple objects fall inside the selection cone. The significant advantage of this technique is the user is permitted larger error when targeting distant or small objects.

3 Novel Direct-Manipulation Techniques

3.1 Introduction

This section presents two novel techniques for selecting objects in immersive virtual environments using a single 6-DOF magnetic tracker. These techniques advance the state of the art by exploiting the participant's visual frame of reference and fully utilizing the position and orientation data from the tracker to improve accuracy when selecting objects. Preliminary results from pilot usability studies validate our designs. Finally, the two techniques can be combined to reduce each other's weaknesses.

The two selection techniques, *aperture* and *orientation*, are used in the test application, an immersive VE for visualizing a computational fluid-dynamics dataset in which users select and directly manipulate components of 3D widgets to control visualization tools [9].

3.2 Aperture selection

The aperture selection technique is a modification of the flashlight selection technique in which the apex of the conic selection volume, the *from point*, is set to the location of the participant's dominant eye and the *direction vector* of the cone is the vector from that eye through the tracker's location (represented in the VE by a cursor). This "aperture cursor" is a circle of fixed radius and a crosshair aligned with the film plane.

The size of the selection volume is determined by the distance between the eye point and the aperture cursor. A user can adjust the scope of the selection conic volume by moving her hand nearer or further from their eye point to change this distance (see Figure 4).

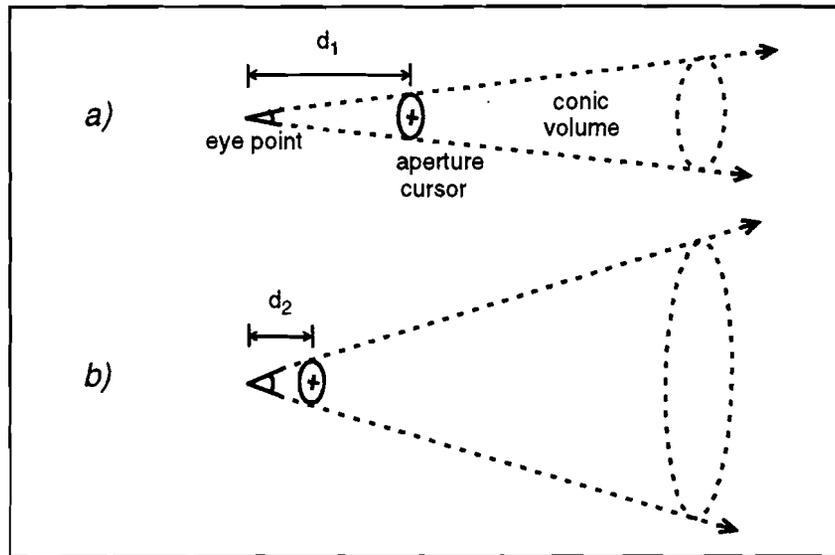


FIGURE 4. a) The conic volume of the aperture selection technique is defined by the eye point, aperture cursor geometry, and distance d_1 between the eye point and aperture cursor. b) A larger volume can be defined by translating the aperture cursor closer to the eye point, thereby reducing the distance d_1 to d_2 .

The aperture requires the user to indicate a position along her line of sight. In preliminary tests, users held the tracker in one hand and the aperture geometry was drawn at the corresponding position in the VE, but this induced considerable arm fatigue. To alleviate this problem, the tracker was mounted on a drumstick prop so as to place the aperture geometry at the end of a virtual representation of the stick in the VE. Anecdotal evidence from further tests indicated that this was less fatiguing because users could keep their arm in a less uncomfortable position by their side (see Figure 5).



FIGURE 5. Using the drumstick prop in an immersive virtual environment.

The advantages of the aperture selection technique are: 1) it closely mimics both how we point at distant objects in the real world and the familiar desktop metaphor of positioning a cursor visually over a target object; 2) it incorporates volume selection as used in the flashlight technique to reduce the effects of tracker and user-induced noise; and 3) the screen is less cluttered because no visual feedback is required other than the aperture cursor itself (though selectable objects should be highlighted to help users determine what they are selecting). In contrast, the flashlight technique requires a transparent outline for the selection cone to help the user compare the current direction of the selection cone with the position of the target object.

An important consideration in this technique is the choice of a from point for the selection volume. Logical possibilities are the positions of either of the user's two eyes or an average of these points. However, since all test subjects exhibited ocular dominance [4], the only option that avoids perceptual confusion is the user's dominant eye. When the cursor is visible only from the non-dominant eye, we dynamically switch the from point to that eye. Our subjects generally liked this selection technique, but some complained that when they attempted to select distant objects, they would focus on that object and could perceive two distinct images of the aperture in the foreground. This phenomenon, due to parallax in the stereo view, may be significant in some applications, but can be resolved by closing the non-dominant eye. A consequence of this solution, however, is that the user's level of immersion is reduced due to the loss of stereo perception.

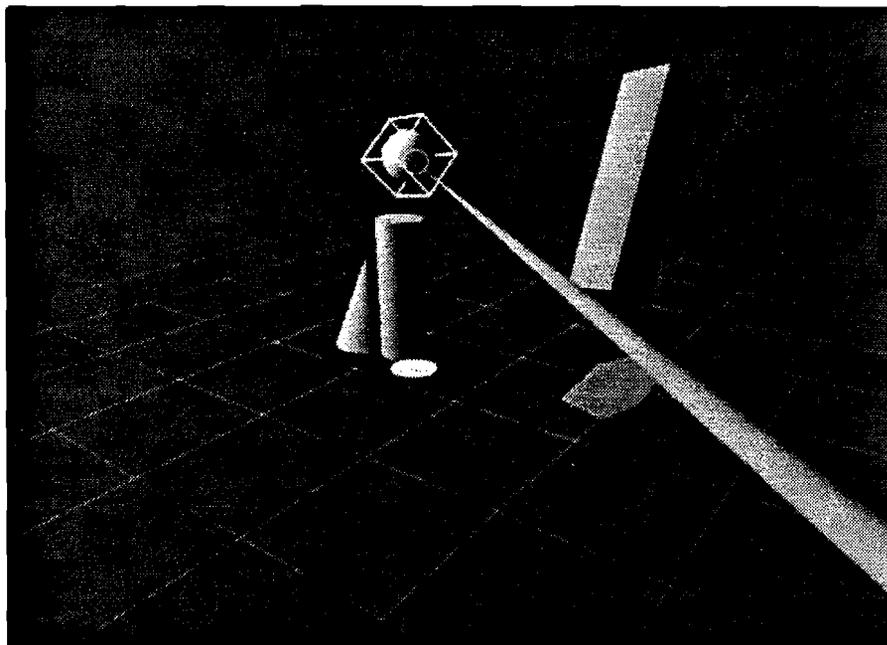


FIGURE 6. The sphere is selected with the aperture selection technique when the aperture circle at the tip of the virtual stick is positioned over it. The cone-shaped object entering the scene from the right of the image is the cursor representation of the stick prop in the IVE.

As in the flashlight technique, we use a disambiguation metric to choose among multiple candidate objects when single object selection is desired. It can be difficult to select one of a set of closely spaced objects. Also, because the cone has infinite extent, it is not clear what actions are appropriate when very distant objects are selected.

Figure 6 illustrates the use of the aperture selection technique.

3.3 Aperture with orientation

In the real world, we orient our hands to match the orientation of a target object before we manipulate the object (e.g., grabbing a book or coffee mug). We can similarly use the 3D orientation information provided by the tracker to augment the aperture technique for selecting objects in a VE. If multiple objects fall within the conic volume of the aperture, we select the object whose orientation most closely matches that of the tracker. Orientation information thus provides the primary disambiguation metric. If all candidate objects have similar orientations, the basic aperture technique disambiguation metric is used.

Let the *longitudinal axis* of an object be the axis along which it is scaled the most. Figure 7 illustrates how the orientation of an object can be compared to the orientation of a plane defined by a tracker's orientation information.

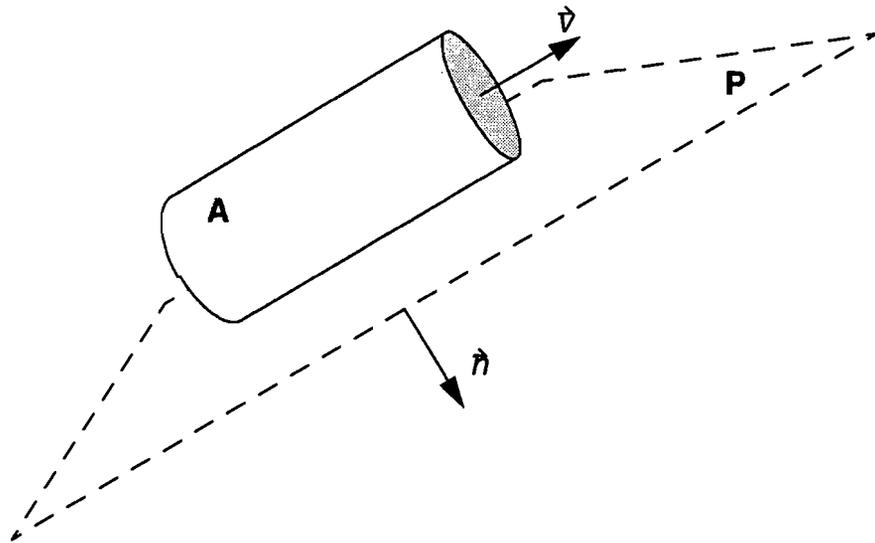


FIGURE 7. The orientations of object A and the plane P are tested by comparing the angle between P 's normal \hat{n} and A 's longitudinal axis \hat{d} .

In the implementation, visual feedback is provided to aid in matching orientation by drawing two transparent parallel "plates" with the aperture geometry

(see Figure 8 and Figure 9). These plates serve as the user's virtual "hands" and are positioned and oriented around an object to select it.

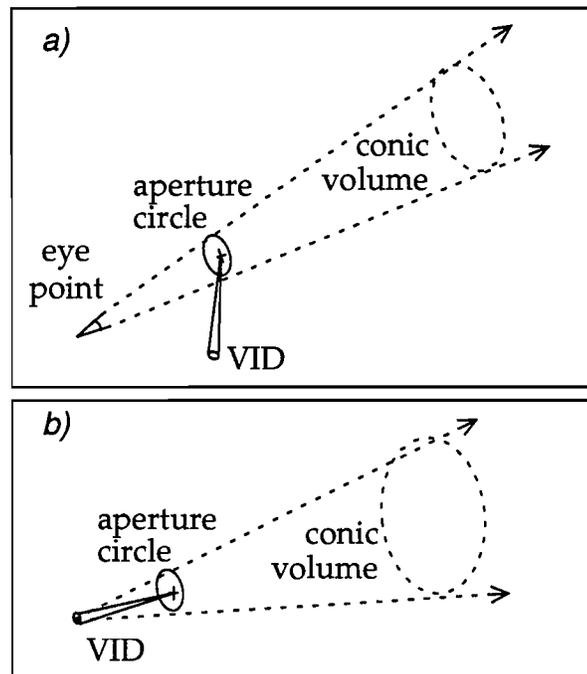


FIGURE 8. The aperture technique. a) The conic volume described by the viewpoint and aperture. b) The conic volume described in the "flashlight" configuration. Both *a* and *b* are shown with the drumstick VID. In our implementation, the conic volume is semi-infinite.

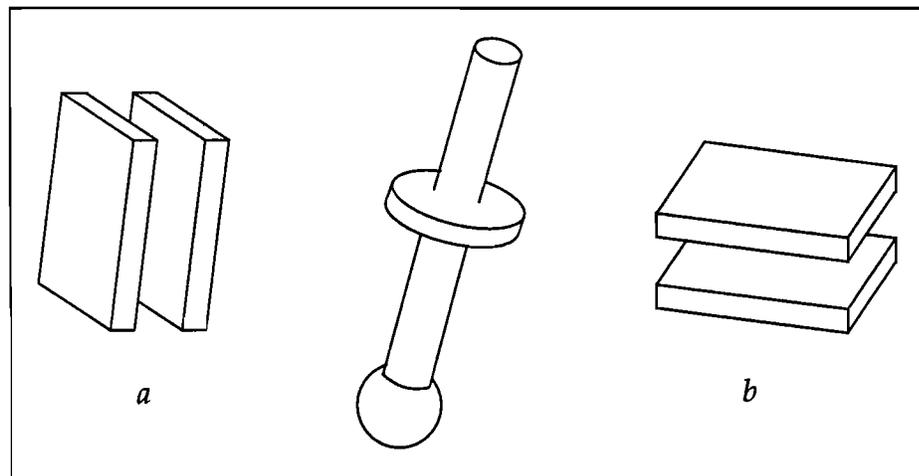


FIGURE 9. Orientation selection technique. For this technique the cursor geometry is a pair of parallel plates which indicates the current orientation of the tracker. This cursor may be used by itself or in conjunction with a VID such as the drumstick. a) shows the cursor orientation that would select 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.

4 VR User Interface Package for UGA

The Virtual Reality User Interface (VRUI) package for Brown's UGA system is a framework for implementing interaction techniques. The touch, laserbeam, flashlight, aperture, and aperture with orientation direct-manipulation techniques described in Sections 3 and 4 have been implemented and are part of the VRUI package.

This section presents an overview of the VRUI package, gives technical notes on the implementation, and describes the classes that define the package.

4.1 Overview

VRUI provides a framework for interpreting data from an input device and translating it into modifications of the scene. In the UGA system, an *input device* generates *events* that are passed to an *event handler*. The handler interprets each event and may take actions that *modify* the *scene* (see Figure 10). This design allows the implementation of general input devices and handlers.

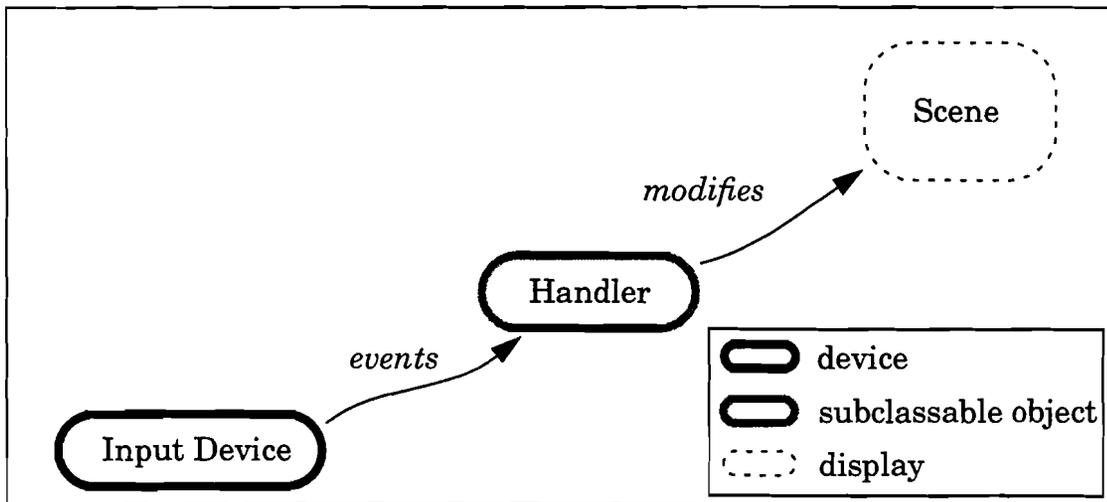


FIGURE 10. The UGA model for translating input device data to scene modifications.

VRUI modifies the UGA interaction model by replacing the handler object with an *interaction technique* object and passing input device events to a *tracked device* object before they reach the handler (see Figure 11). The interaction technique is a subclass of the handler object that adds a visual representation of the technique to the scene. The tracked device object transforms input device events into a format suitable for the interaction technique.

The model in Figure 11 can be used for general interaction techniques. A broad range of interaction techniques is better supported by subclassing the interaction technique object to a *direct-manipulation technique* object (see

Figure 12). The direct-manipulation object extends the interaction technique object with a state variable and two additional member objects, a *selection technique* and a *transform technique*, both of which are subclasses of the interaction technique object. The state variable indicates whether the direct-manipulation object is in selection or transformation mode. Events reaching the direct-manipulation object are passed to the selection technique or the transform technique depending on the value of the state variable. Visual feedback may be added to the scene by the direct-manipulation, selection, or transform techniques. This is the model used by VRUI for implementing interaction techniques.

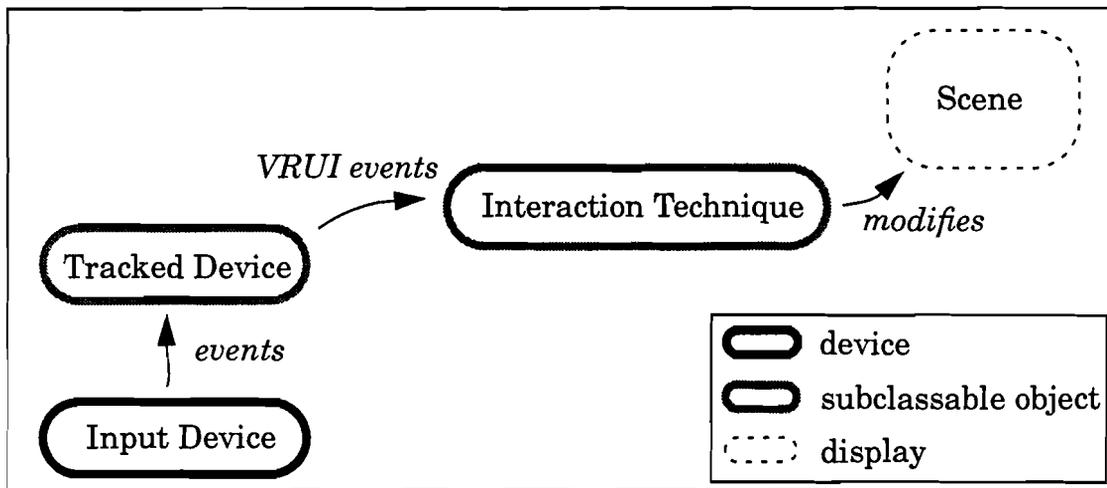


FIGURE 11. The VRUI model for translating input device events to scene modifications through an interaction technique.

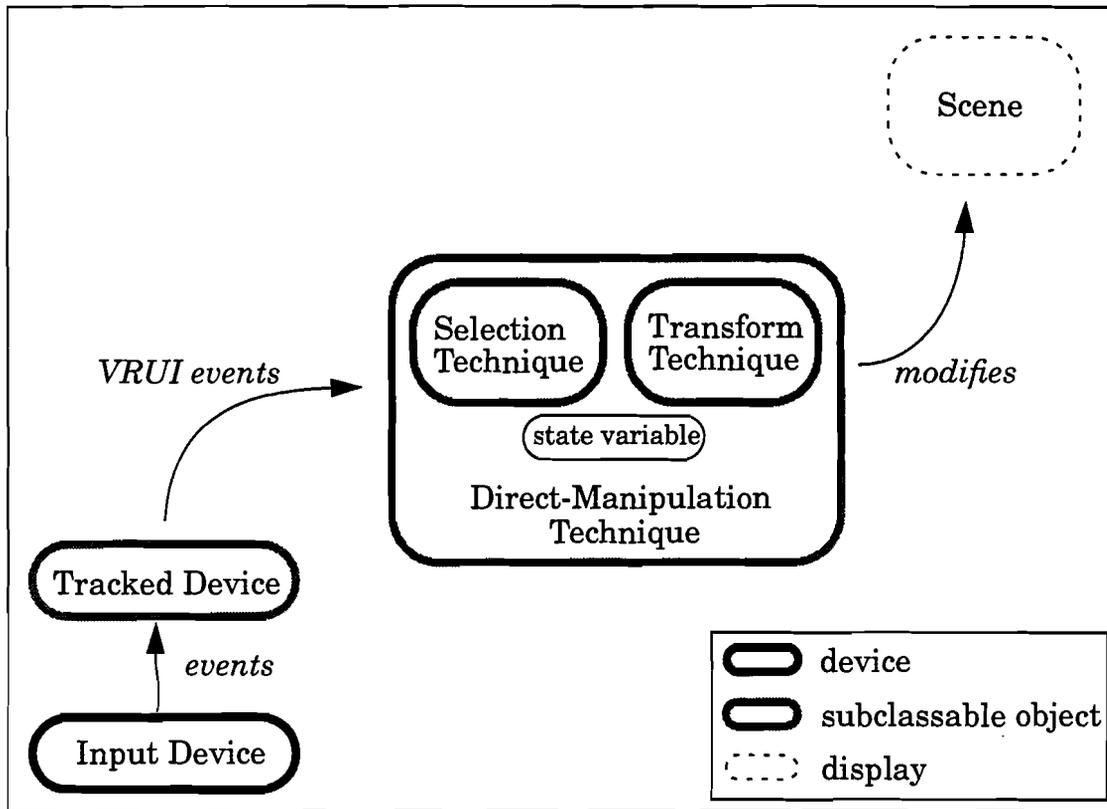


FIGURE 12. The VRUI model for translating input device events to scene modifications through a direct-manipulation interaction technique.

4.2 Definitions

A *selection cone* is a cone defined by an apex, direction, and spread angle, or by an apex location and a circle that is a cross-section of the cone.

A *selectable* object is an object that is a candidate for selection—that is, it meets the current criteria for selection. It is *selectable* as opposed to *selected* because, when a user wants to select a single object, a metric is used to pick the “best” object from the set of selectable objects.

The metric used to pick the best object from a set of selectable objects is the *disambiguation metric*.

4.3 Implementation notes

This section discusses important technical issues addressed in VRUI’s implementation. They serve as examples of implementation issues drawn from previous work and implementation experiences and are not intended to be an exhaustive list of issues to address in implementing a direct-manipulation technique.

4.3.1 Visual feedback

A measure of the success of a selection technique is how well it selects the object the user intended and an important contributing factor is the visual feedback provided. Poorly designed feedback can render a selection technique unusable. The following sections describe types of visual feedback used in this implementation to improve the effectiveness of each selection technique.

4.3.1.1 Pre-select feedback

During the selection phase of direct-manipulation, objects in the scene are tested for selection when a frame is drawn. When the user signals the computer to manipulate objects, the selected object from the most recent frame is manipulated in subsequent frames.

The computer must inform the user which object will be selected if the user signals for selection at any given moment. In this implementation, objects that would be selected if the user were to press the button at any instant are highlighted by drawing a bounding box around each object and changing its color.

4.3.1.2 Guidelines

The laserbeam technique requires the user to select objects in the IVE by pointing at them. Providing users with a guideline showing where the selection ray begins and how it is oriented is critical, since users need this information to judge how they should adjust the current ray position and direction to hit their target object. When the ray does not intersect an object, it should extend to infinity. Note that when an object is selected, the ray should extend only far enough to intersect that object.

4.3.1.3 Transparency

Transparency is a way to provide feedback without obstructing objects behind the transparent object. In the flashlight technique, the selection cone is drawn transparent so that the user can judge the flashlight's current position and the position and orientation adjustments necessary to target an object.

4.3.1.4 Shadows and depth cues

Providing cues to aid the user in making spatial judgments increases the effectiveness of a selection technique. Shadows are a means for determining height and spatial proximity of neighboring objects. A textured groundplane can aid the user in a similar way. This implementation uses a grid groundplane that provides a fixed frame of reference with regularly spaced marks to help in positioning objects.

4.3.2 Touch

One danger of the touch technique is it lends itself to simple implementations that are difficult to use in practice. Ideally, an implementation of a touch technique should model the way we touch and grab objects in the real world, including haptic feedback, precise tracking of the fingers and palm, and effects of hand-object interaction. Each of these points is a significant research topic in itself and implementation at interactive rates is currently impossible.

The first implementation of touch selection tested whether the tracker was inside an object. From this first implementation came several observations:

- users often wished to select an object that the cursor was not precisely touching.
- users often pushed the tracker through an object and out the other side— it would appear to the user that the object should be selected, but because the tracker was not mathematically inside the object, it was not selected.
- the tracker occasionally was not sampled as the user passed through an object due to the tracker's finite sampling rate. This situation misleads the user into thinking the tracker had not hit the object.

A second implementation of the touch technique incorporated two heuristic rules to attempt to correct the above problems. As before, a test for whether the tracker was in an object was performed. If the tracker was not inside, the line segment between the last reported tracker position and the current tracker position was intersected with the object to fix the temporal aliasing problem. If this ray did not intersect the object, the distance between the tracker and the last surface point touched was determined and, if below some threshold, the object was selected.

4.3.3 Approaches to volume intersection tests

The flashlight and aperture selection techniques require volume selection tests: both require knowledge of which objects in the scene fall within a conic volume. These volume intersection tests ask the question: "Does object A intersect object B?" When objects A and B are three-dimensional objects, this is not a trivial problem. Analytical solutions for volume intersections between primitive objects such as spheres or parallelepipeds with uniform orientation are straightforward, but there is no general analytical solution to volume intersection. Algorithmic approaches that iterate over the features (vertices, edges, and faces) of polygonalized representations of the given objects must be used.

The following sections describe my attempts to handle volume intersection. Each technique builds on the shortcomings of the previous technique. Because

each is an approximation, they all have exceptional cases in which they incorrectly report “no collision” to a intersection query. The approximations exploit the fact that one of the objects being tested for intersection is always a cone to simplify the intersection problem.

Experience has shown that a partial solution to this problem is unacceptable. The user relies on the result of this test ten or more times per second to highlight the appropriate objects during selection. Unpredictable visual feedback confuse users and they will want to stop— clearly an undesirable effect of a user interface.

4.3.3.1 Sampling the selection cone

The first approach I took to the volume intersection problem was to use ray-intersection tests: rays were “shot” from the apex of the selection cone and passed through a discrete number of representative areas of the cone. An object was reported as intersecting the cone if any rays hit it. Figures 11 and 14 illustrate representative sample points in a cross-section of the aperture cone through which rays pass.

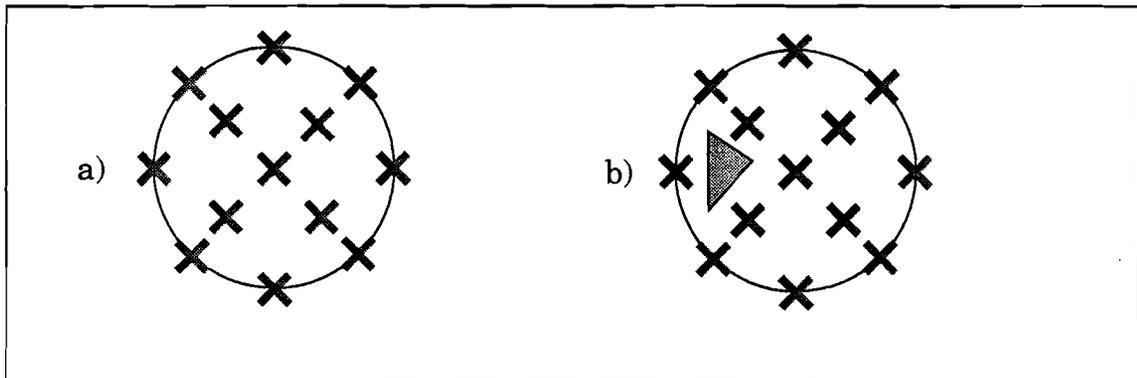


FIGURE 13. A cross-section of an aperture cone is shown. a) Sample points through which rays are “shot.” b) A situation in which this approach fails: the shaded triangle is not intersected by any of the sample rays and is not selected, although it lies inside the aperture cone.

In practice, objects are often located within the aperture selection cone but not intersected by one of the sample points. The technique is especially poor at selecting objects that are distant or small. An object can always be found that causes the approach of sampling only a finite number of locations inside the selection cone to fail.

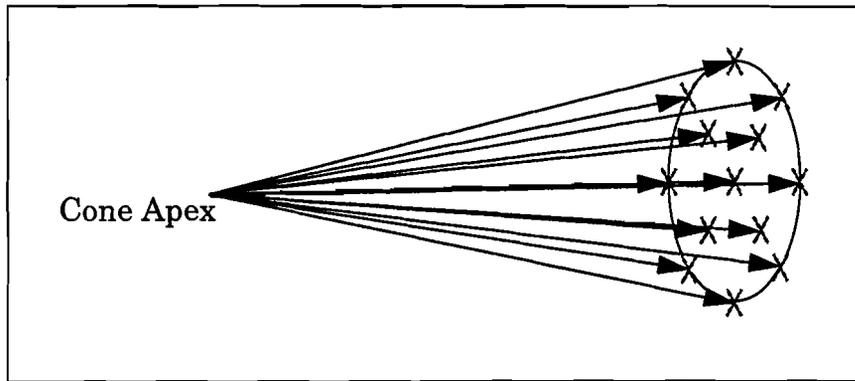


FIGURE 14. Rays are shot through the thirteen sample points in a cross-section of the aperture cone.

4.3.3.2 Vertex inclusion tests

Another approach was based on the hypothesis that a polygonalized object intersects the selection cone if any of its vertices are located inside the selection cone (see Figure 15). The hypothesis is valid but incomplete, because objects without a vertex inside the cone may have a face that intersects the selection cone. Consequently, this approach will also fail in some situations.

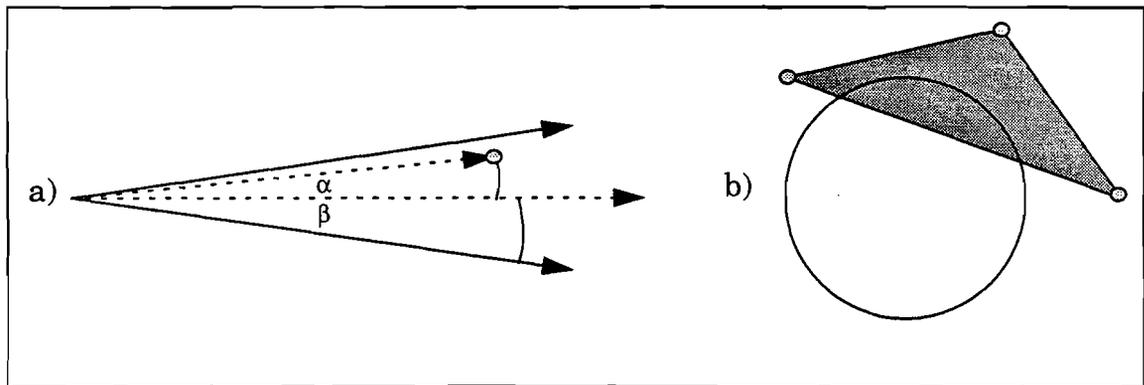


FIGURE 15. a) The vertex inclusion test compares whether α , the angle between a vertex and the centerline of the selection cone, is less than β , the maximum angle subtended by the selection cone. b) A situation in which the vertex test fails— all vertices are outside the selection cone but the triangle face intersects the cone.

This technique and the one presented in the next section require at worst computational time that is linear in the number of vertices or edges of an object. Linear time cost is a significant penalty compared to the constant time required for a ray intersection with primitive surfaces. Polygonalized objects with many vertices such as a sphere can reduce the frame rate to unacceptable levels.

One approach to reducing the average time for intersection tests is to randomly test an object's features. The geometry data structures tend to store neighboring features near each other. The idea is to recognize that if a feature does not intersect the selection cone then its neighbors probably do not either. Non-repeating pseudo-random indices are used instead of sequential indices to access an object's features. The iterative loop "jumps" around the object, testing topologically different areas at each iteration so that, on average, the time to find a vertex inside the selection cone is reduced. In the worst case, however, the problem still requires $O(n)$ time where n is the number of features an object has.

4.3.3.3 Projected edges

The third approach properly handles the exceptional case in the previous section. Each edge of an object is projected to a cross-section of the selection cone. If any projected edge intersects the circular cross-section, then the object intersects the selection cone. Projecting edges in this way reduces the 3D intersection problem to a 2D intersection problem.

As in the vertex inclusion test, however, this technique incorrectly dismisses intersections when edges do not intersect the selection cone but a face of the object does (see Figure 16). These situations arise frequently in practice, causing this technique to behave unpredictably.

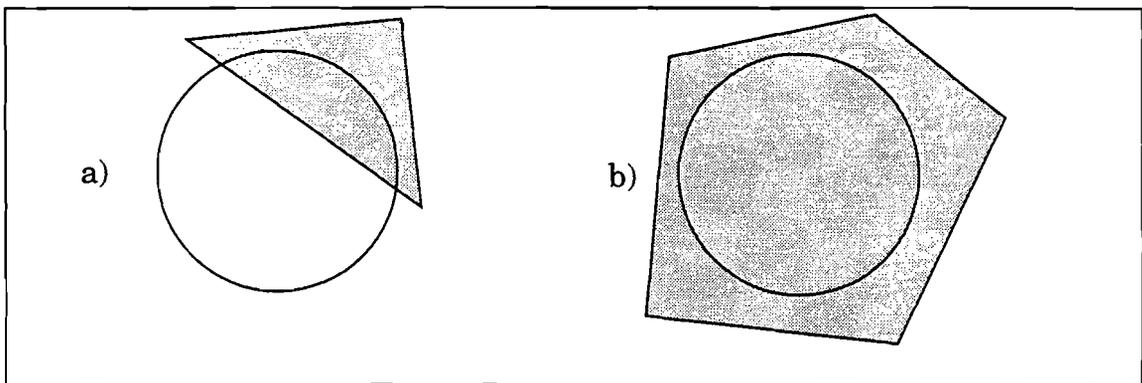


FIGURE 16. a) One edge of the shaded triangle intersects the selection cone cross-section. b) None of the edges of the shaded pentagon intersect the selection cone cross-section. However, the pentagon does intersect the selection cone and therefore should be selected.

4.3.3.4 I-COLLIDE

The I-COLLIDE [2] package was developed at the University of North Carolina at Chapel Hill for interactive collision detection. Polytopes are registered with the package. The position and orientation of the objects can be updated and an exact list of intersecting objects is reported.

I-COLLIDE has been incorporated in my flashlight and aperture selection techniques by querying for collisions between each object in the environment and a cone polytope. The cone acts as the flashlight's "beam" and the aperture technique's "conic volume," with apex at the user's dominant eye and passing through the aperture circle. This is an exact solution superior in completeness and efficiency to the approaches discussed in Sections 4.3.3.1 - 4.3.3.3.

I-COLLIDE tracks closest points between pairs of convex polytopes. Efficiency is achieved by exploiting temporal and geometric coherence: the closest features change infrequently as polytopes move along paths. The Lin-Canny[12] method is used to determine the collision status between features. Updating a closest pair for each object pair takes expected $O(1)$ time. The running time for I-COLLIDE is $O(n + m)$ where n is the number of simultaneously moving objects and m is the number of objects whose axis-aligned bounding boxes overlap.

4.4 C++ class descriptions

4.4.1 Classes related to input devices and events

4.4.1.1 VRtracked_device

A `VRtracked_device` class is an abstraction of an input device. The details of an input device are hidden from a direct-manipulation technique. The direct-manipulation techniques implemented in this work require device events as input. In addition, they expect to be able to query the `VRtracked_device` object at any time for a 3D position and orientation record.

A `VRtracked_device` object must provide three functions:

- events generated by the input device it is representing,
- access to a 3D position and orientation record from the input device,
- a graphical representation of the input device it represents.

4.4.1.2 VRhandler

The `VRhandler` class is the base class for objects that receive events from a `VRtracked_device` object. The handler has access to the objects in the scene and is responsible for interpreting events and modifying the scene objects.

4.4.2 Interaction Techniques

4.4.2.1 VRinteraction_technique

The `VRinteraction_technique` class is a subclass of `VRhandler` and serves as a base class for interaction techniques. An interaction technique receives events from a `VRtracked_device` and may add geometry to a scene.

4.4.2.2 VRdirect_manipulation_technique

The `VRdirect_manipulation_technique` class is a subclass of the `VRinteraction_technique` class. It contains a pointer to a `VRselection_technique` and a `VRtransform_technique` object. The default behavior of a `VRdirect_manipulation_technique` object is to pass events received from a `VRtracked_device` object to the selection or transformation objects depending on the state of the direct-manipulation task.

4.4.2.3 VRselection_technique

The `VRselection_technique` class is a subclass of `VRinteraction_technique`. A selection technique object is responsible for interpreting events from a `VRdirect_manipulation_technique` object and testing objects in the scene for selection. A `VRselect_test` (see below) object may be used to help automate this action.

4.4.2.4 VRtransform_technique

The `VRtransform_technique` class is a subclass of `VRinteraction_technique`. A transform technique object is responsible for interpreting events from a `VRdirect_manipulation_technique` object and manipulating selected objects. Manipulation of objects must maintain the constraints that were originally met to select objects being manipulated (e.g., objects selected with the aperture selection technique should remain under the aperture circle).

4.4.3 Select test classes

Determining which objects in a scene are selected is done by iterating through the scene object database and performing a selection test on each object.

4.4.3.1 VRselect_test

The `VRselect_test` class handles the details of iterating over the objects in the object database. The abstract method

```
int test_obj( GEOobj *obj, double *metric) = 0;
```

is expected to be defined by subclasses of `VRselect_test`. This method should supply the details of a selection test for the object pointed to by `obj`. If the object meets the conditions for selection, the method should return `TRUE` and supply a value for the `metric` variable indicating how well the object met the selection conditions. If the object is not selected, the `test_obj(...)` method should return `FALSE`.

If multiple objects meet the criteria for selection and only a single object can be selected, then the object with lowest metric value is selected.

4.4.4 Examples of direct-manipulation techniques

Each class below subclasses from the `VRdirect_manipulation_technique` class and supplies a `VRselection_technique` and `VRtransform_technique` object. Initially, the direct-manipulation class sends events received from the `VRtracked_device` object to the selection object. When the user signals to begin manipulation of the selected objects, events are sent to the transform technique until the user signals for selection again.

The behavior of the `VRselection_technique` object is described for each selection technique below. The objects in the scene database are unselected at each frame and then tested by the `VRselection_technique` object.

When the user has signaled to transform objects, the `VRtransform_technique` begins to receive events from the `VRdirect_manipulation_technique`. The transform object updates the position and orientation of each selected object so that the conditions that initially led to that object's selection are met at each frame as the user manipulates the input device.

4.4.4.1 VRtouch_select

The touch selection object tests whether the 3D position event information passed by the direct-manipulation object is contained in each object in the object database. If the cursor and object meet the conditions outlined in Section 4.3.2 for the touch technique, then that object is selected.

4.4.4.2 VRray_laserbeam

The laserbeam selection technique defines a ray from the position and orientation passed from the `VRtracked_device` and intersects each object in the scene database with it. Intersected objects are selected for the current frame.

4.4.4.3 VRray_from_eye

The ray-from-eye technique intersects the objects in the scene database with a ray based at the user's dominant eye and running along a vector from that eye through the 3D point passed by the VRtracked_device.

4.4.4.4 VRaperture_flashlight

The aperture flashlight technique defines a cone object with apex on the VRtracked_device and aimed along one of the tracked device's coordinate system's primary axes. The spread angle of the cone is usually fixed, though we experimented with dynamically changing the spread angle with limited success. I-COLLIDE is used to intersect objects in the scene database with the selection cone. The objects reported in the collision list are selected.

4.4.4.5 VRaperture_from_eye

The aperture-from-eye technique defines a conic selection volume with apex located at the user's dominant eye. The centerline of the selection volume passes through the tracked point on the VRtracked_device. The spread angle of the cone is calculated at each frame so that the cone exactly passes through the aperture circle displayed on the VRtracked_device. Again, I-COLLIDE is used to test for object intersections with the selection cone. The objects reported in the collision list are selected.

4.4.4.6 VRorientation_select

The orientation selection technique tests how well an object's orientation matches the orientation of a cursor. Because UGA works with uniformly scaled, canonical primitives, the scale of an object reveals information about its shape. For example, an object scaled more in the y axis and less in the x and z axes will appear elongated in the y direction. This analysis of the scale of an object can reveal the axis along which the object is scaled the most. This axis of maximum scale is defined to be the *longitudinal axis* of the object.

The length of the vector returned by the cross product of an object's longitudinal axis and the vector perpendicular to the cursor plates (see Figure 7) is a measure of how well the orientations of the two objects match. In words, if the angle between the normal to the plates and an object's longitudinal axis is close to 90 degrees, then the cursor has been oriented such that it could surround the target object. As the angle approaches zero, the similarity between the orientation of the cursor and object's longitudinal axis deteriorates. Objects whose angle measure is greater than some threshold (~10 degrees in the current implementation) are selected.

4.4.4.7 VRaperture_orientation

The aperture-and-orientation technique combines the aperture technique with the orientation selection technique. Objects falling in the aperture cone are selectable. If more than one object is positioned in the aperture cone, then the orientation information is used to find an object with which the user has aligned the cursor plates. If a match is found, that object is selected. Otherwise, the object nearest the centerline of the selection cone is selected.

Note that when the orientation information is not used, this hybrid technique selects objects as well as the aperture technique alone.

5 Future Work

Future work will include a study of how the Virtual Technologies Cyber Glove can be used instead of the stick prop input device. While the glove requires more overhead in terms of calibration and setup time it should provide a more natural interface. The touch, flashlight, and aperture techniques will extend to use with the glove. The glove should provide alternative ways to set parameters, a more intuitive interface to the orientation selection technique, and other opportunities to use the collision detection package for directly manipulating objects by touch.

Further study of the relationship between an interaction technique and widgets is necessary. In particular, the situation in which a widget has control over which of its components are selected should be further studied.

An exploration of how scientific visualization techniques perform in fishtank VR where the user is not fully immersed in a world, but are presented with stereo views and interact with 6-DOF devices could prove an interesting contrast our present BOOM setup.

Users tend to use minimal head and body movement in our system when viewing and, especially, when interacting with objects. On a related note, arm movements during selection and manipulation are minimal. Exploration of interfaces which require or encourage greater body motion should prove useful. The current situation in which users are generally tense is not desirable.

7 Acknowledgments

I would like to thank Professors Andries van Dam and John Hughes for making this work possible through their support and guidance. I would also like to thank the graphics lab's sponsors: NASA, NSF, Sun, HP, SGI, Microsoft, and Taco.

Kenneth Herndon and Robert Zeleznik have provided invaluable ideas and suggestions on user-interface design, the aperture and orientation selection techniques presented in Section 3, and approaches to the implementation.

Seung Hong deserves special acknowledgment for his efforts to design, run, and analyze the user studies described in this work. I would also like to thank members of the graphics group for their suggestions. In particular, Seung Hong, Matthias Wloka, Timothy Miller, Sumi Choi, Daniel Goldstein, Paton Lewis, Kenneth Drew, Rob Mason, Jeff Beall, Diego Velasco, and Jim Buckhouse offered useful suggestions.

A special thank-you to the I-COLLIDE group at the University of North Carolina-Chapel Hill and to Kenrick Drew for his work on porting I-COLLIDE to Brown's UGA system — without that package as a tool this work would not be as interactive nor as robust.

I am grateful to Professor Bruce Kay and graduate students Li Li, Andrew Duchon, and Jeff Saunders of the Cognitive Science Department at Brown University for their general input as well as suggestions on our user-study procedures, and to Katrina Avery, John Hughes, Bob Zeleznik, Ken Herndon, and Matt Ayers for reviewing draft versions of this paper.

Finally, I'd like to thank my parents, Joanne and Charles Forsberg, and brothers Bob, Scott, and Jim for their constant support and encouragement over the past two years. I am also grateful for their technical assistance in building the drumstick prop and installing the signal button, which performed reliably as the primary input device throughout this work.

- [14] Meyer, T. and Globus, A., *Direct-manipulation of isosurfaces and cutting planes in virtual environments*, Technical Report 93-54, Department of Computer Science, Brown University, 1993.
- [15] Meyer, T. and Hughes, J.F. Scheduling time-critical graphics on multiple processors, paper submitted to *SIGGRAPH '95*.
- [16] Mine, M., ISAAC: A virtual environment tool for the interactive construction of virtual worlds, Technical Report TR95-020, Department of Computer Science, UNC Chapel Hill, 1995.
- [17] Shneiderman, B., Direct-manipulation: a step beyond programming languages, *IEEE Computer*, 16(8), August 1983, pp. 57-69.
- [18] Shoemake, K., ARCBALL: A user interface for specifying three-dimensional orientation using a mouse, in *Proceedings of Graphics Interface '92*, May 1992, pp. 151-156.
- [19] Snibbe, S.S., Herndon, K.P., Robbins, D.C., Conner, D.B. and van Dam, A. Using deformations to explore 3D widget design, in *Proceedings of SIGGRAPH '92*, 26(2), July 1992, pp. 351-352.
- [20] Stoakley, R., Conway, M. J., and Pausch, R., Virtual reality on a WIM: interactive worlds in miniature, in *Proceedings of ACM SIGCHI '95*, 1995.
- [21] Wloka, M. M., Lag in multiprocessor virtual reality, *PRESENCE: Teleoperators and Virtual Environments*, 4(1), Winter 1995, pp. 50-63.
- [22] Wloka, M. M. and Greenfield, E. S., The virtual tricorder, in *Proceedings of UIST '95*, November 1995, pp 39-40.
- [23] 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 *Proceedings of SIGGRAPH '91*, July 1991, pp. 105-112.
- [24] Zhai, S., Buxton, W., and Milgram, P., *The "silk cursor": investigating transparency for 3D target acquisition*, Technical note, University of Toronto, 1994.