ISAAC : A Virtual Environment Tool for the Interactive Construction of Virtual Worlds

Mark R. Mine

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

mine@cs.unc.edu

Abstract

This paper presents a description of ISAAC, the Immersive Simulation Animation And Construction program designed and built at the University of North Carolina at Chapel Hill (UNC-CH). ISAAC is a scene composition application for the interactive construction of virtual worlds. Working directly in a virtual environment, objects can be positioned, oriented and scaled using direct and indirect manipulation techniques. Object configurations are stored in ASCII files which can be used to recreate object configurations at a later date for further manipulation and interactive exploration. ISAAC is not a modeling program, worlds are created by manipulating pre-generated three-dimensional models (which can come from sources such as Computer-Aided Design programs or three-dimensional scanning devices). ISAAC has been designed to overcome some of the limitations of working in a virtual environment and to take advantage of the natural and intuitive forms of interaction available in a virtual world.

1. Introduction

Despite considerable promise there has been a surprising lack of real-world applications in the virtual world. Why? What is so hard about working in a virtual environment? While a great deal of research has been focused on technological limitations such as low resolution displays, limited tracking systems and end-to-end delays, some of the greatest difficulties facing virtual-environment application developers are conceptual rather than technological in nature. We are unfamiliar with this new medium we work in. We do not fully understand that many of the characteristics that give a virtual environment its power are also the source of its problems. Take, for example, two of the key advantages to working in a virtual world: immersion and the direct manipulation of objects.

Immersion within a virtual world means we can distribute controls (and other information) about the user. It also means, however, that these controls can be difficult to find in a virtual world. Even if visible, controls may obscure your view of the virtual environment or you may have to traverse the environment to reach them. This is unlike a through-the-window system where controls typically have a well defined location on the screen and are always within reach. Though we gain advantages in terms of density and distribution of information in a virtual

world, we lose some of the organization and efficiency forced upon us by two-dimensional computer interfaces. We are stuck with the problem of finding the light switch in the dark.

Similarly, many tasks are implemented as direct manipulation tasks in virtual worlds applications because direct manipulation is such a natural and intuitive form of interaction. The types of real-world problems that can be solved using unconstrained direct manipulation, however, are limited because humans lack the fine motor control necessary to precisely manipulate objects. This is compounded by the fact that virtual worlds lack both effective techniques for alphanumeric input (which we use for precise manipulation in the computer world) and haptic feedback (which we depend on to constrain our motion in the physical world). We are stuck with the worst of both worlds.

The virtual world, therefore, though a powerful new medium in which we can use our eyes, ears and hands much as we do in the real world, is far from the panacea of computer-aided applications we would like to believe. There are many challenges and limitations to be overcome.

The interaction techniques used in ISAAC and presented in this paper were chosen because they enhance the natural and intuitive forms of interaction possible in a virtual world (often in ways not possible in the real world). They augment user interaction and/or provide the types of constraints necessary for precise interaction with virtual objects. ISAAC has been developed as a testbed for virtual world interaction techniques. It is the outgrowth of research by the author into the impact of working in a virtual world. The focus of this research is the development of interaction techniques which compensate for the limitations of working in a virtual world while simultaneously exploiting its power.

The organization of this paper is as follows: I begin with a review of related work in the area of interactive creation and manipulation of objects in a virtual world in section 2. Next an overview of the ISAAC system is presented in section 3. Section 4 presents examples and discussions of some of the more effective virtual-world interaction techniques used in ISAAC. Section 5 is a discussion of the importance of constraints for precise interaction in a virtual world (with examples given from the ISAAC system). Finally, I end with a discussion of future work in section 6 and a brief conclusion in section 7.

2. Previous Work

One of the earliest systems for the creation and manipulation of three-dimensional objects in a virtual world was 3DM. Designed and built at the University of North Carolina [Butterworth 1992], 3DM is an immersive modeling program which allows users to create simple geometric objects (cylinders, spheres, etc.) interactively in a virtual environment. Though 3DM includes several grid and snap functions, it lacks many of the other aids and constraints necessary for accomplishing precise work.

JDCAD (by Liang and Green at the University of Alberta), though not an immersive system, tackled many of the issues involved in the interactive creation of three-dimensional objects. By using a 6-DOF input device, users could interact directly in three-dimensions. Liang and Green showed

encouraging results when comparing JDCAD with conventional through-the-window modeling systems (using 2D input devices) [Liang 1994].

PolyShop, developed at the Institute for Simulation and Training at the University of Central Florida, explored important issues such as two-handed interaction and the use of physical devices (such as a desktop) to provide low level haptic feedback [IST 1995].

Conceptual Design Space (CDS) developed at Georgia Tech's Graphics Visualization and Usability Center is intended for the interactive design of architectural spaces [Bowman 1995]. CDS is an immersive, head-tracked application in which objects are created by specifying vertices on a ground plane which get extruded into three-dimensional objects (such as walls). CDS includes a three-dimensional menuing system and several 3D widgets for object manipulation.

The Worlds-In-Miniature (WIM) project at the University of Virginia [Stoakley 1995] demonstrates how interaction techniques not possible in the real world can be effective tools in the virtual world. In the WIM interface, a user manipulates full sized objects in the immersive world by moving around representations in a miniature copy of the world floating in front of him. Here a user's range of interaction has been extended by taking advantage of the ability to interact at multiple levels of scale simultaneously in a virtual environment.

3. ISAAC Overview

ISAAC is a scene composition application designed for the interactive construction of virtual worlds. ISAAC is not a modeling program, worlds are created by manipulating pre-generated three-dimensional models. Models can be from any source such as Computer-Aided Design programs or three-dimensional scanning devices.

ISAAC is an immersive environment application, you work directly in the virtual world. The virtual-environment system used for ISAAC consists of a stereo head-mounted display, a tracking device (to measure the position and orientation of the user's head and hand), a five button input device (the Python) and a host and graphics computer. All images are generated in real time using the Pixel Planes 5 graphics computer [Fuchs 1989]. ISAAC was written using the C programming language and was built upon the extensive set of virtual worlds libraries developed at UNC.

In a typical ISAAC session, models are placed into the environment using a two-step process. First you begin by specifying a set of working objects by choosing the three-dimensional models to be loaded into the system at program startup. These are the objects that will *potentially* populate your environment. If, for example, you were building a model of a room environment you might specify several types of chairs, tables and other furniture. Note that not all of these objects will necessarily end up in the final environment. During the session, you choose objects from the working set (for example a particular type of chair) and interactively place them in the environment. Note that you may simultaneously have multiple copies of the same object at several locations (several instances of a column to make a colonnade for example). If you discover you are missing a required model from your working set, it can be

added to the working set at any time during the ISAAC session using keyboard commands.

Objects that have been placed in the environment can be positioned, oriented and scaled using direct and indirect manipulation techniques. Interaction with objects is *at-a-distance*, objects are selected and manipulated via ray-casting (indicated by a laser beam emanating from the user's hand). Objects are selected by pointing the laser beam at a desired object and clicking on a button on the input device. Any object visible in the environment can be selected for interaction. Objects are manipulated using direct gestural interaction (i.e. movements of the hand are mapped directly to movements of the object) or indirectly using the laser beam and special manipulation widgets. Design issues of action-at-a-distance interaction are discussed in Section 4.1 below.

Direct object manipulation can be hand-centered or object-centered. This refers to the center of rotation that is used in mapping rotations of the hand onto rotations of the object. Hand-centered manipulation is most closely analogous to object manipulation in the real world, when you grab an object it pivots about the center of your hand. Object-centered manipulation allows the user to rotate an object about its center while being physically removed from the object (standing back to get a better perspective, for example).

Object manipulation can be constrained or unconstrained. In the unconstrained mode, all changes in position and orientation of the user's hand are mapped onto the object. Constrained manipulation, which is specified using a constrained motion widget (Section 5.0), allows for one and two-dimensional changes in an object's position (along any principal axis or plane) and one-dimensional rotations about any principal axis.

When using ISAAC, the user is free to walk or fly about the environment. ISAAC includes two modes of flying: pointing mode, in which the user points at the desired destination (see [Mine 1995]) and orbital mode (see [Chung 1994]), in which the user orbits a point of interest based upon the current orientation of his head (see Section 4.2).

To allow users to make changes encompassing the entire environment, the ISAAC system also includes an implementation of the Worlds-In-Miniature interface (Section 4.3). The ISAAC WIM can be used with grid and constrained motion functions to allow for more controlled manipulation of objects.

ISAAC includes conventional editing functions such as cut, copy, paste, duplicate and delete. These are selected using ISAAC's two-dimensional menuing system (Section 4.4). Also included on the menu system are functions such as grouping controls and functions for the control of ISAAC's grid.

At any time during an ISAAC session the current object configuration can be stored in an ASCII file which can be used to recreate the configuration at a later date for further manipulation and interactive exploration.

4. ISAAC: Interaction Techniques

ISAAC includes several forms of interaction which illustrate how natural interaction techniques (such as moving your head to set your viewpoint, or

moving an object by grabbing it) can be powerfully enhanced in the virtual world.

4.1. Action-At-A-Distance

An excellent way to augment the direct manipulation of objects (i.e. reaching out and grabbing something) is to extend your normal reach using some form of action-at-a-distance (AAAD). During AAAD interaction, objects are selected via ray casting (depicted by a laser beam emanating from the user's hand - see Figure 1). Once selected they can then be remotely manipulated by moving the user's hand (with the user's hand motion mapped to object motion) or by using the laser beam to interact with some form of control widgets.

AAAD is limited by the fact that you must be able to see an object in order to select it. AAAD is therefore dependent upon the user's line of sight and is most effective when combined with some form of viewpoint specification such as orbital mode or WIM navigation discussed below. Also, object manipulation in AAAD mode is imprecise (in particular it can amplify noise in hand-tracking data) and is most effective in directions orthogonal to the direction of the laser beam. More precise object positioning requires additional constraints and control devices.

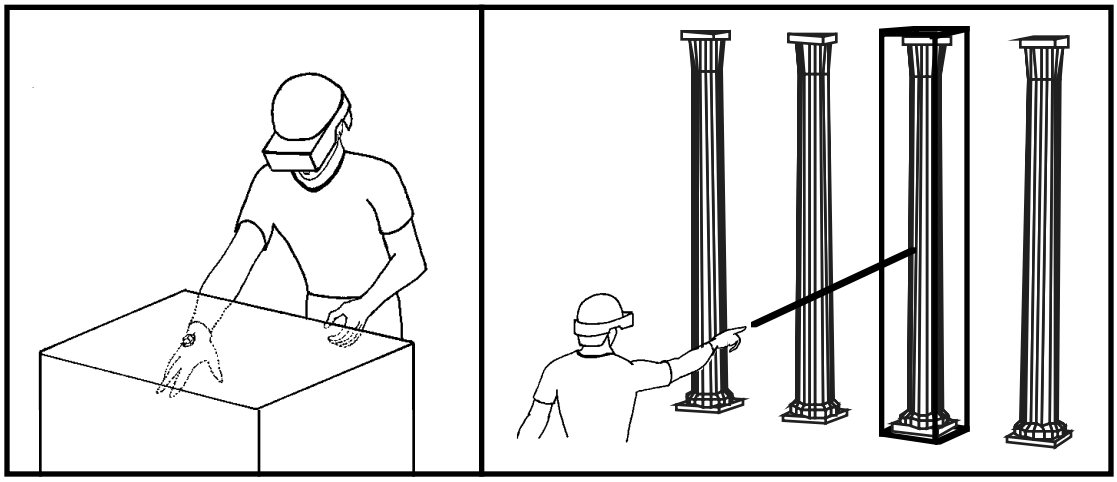

Figure 1: Local vs. Action-at-a-distance

4.2. Orbital Mode

Orbital mode takes advantage of one of the most intuitive means of view specification, moving your head. In orbital mode, objects of interest are constrained to always stay in front of the user. The side of the object currently visible depends on which direction you're looking: look up and you see the object's bottom, look down and you see its top (see Figure 2). This allows you to quickly and naturally see all sides of an object as it *orbits* your head (or you orbit the object). Orbital mode augments viewpoint specification by coupling location in the virtual world with head motion. It has the additional advantage that your hands remain free during viewpoint specification. Note that orbital mode is good for local viewpoint specification and is not a global navigation technique. It does not solve the general problem of navigation through complex virtual worlds.

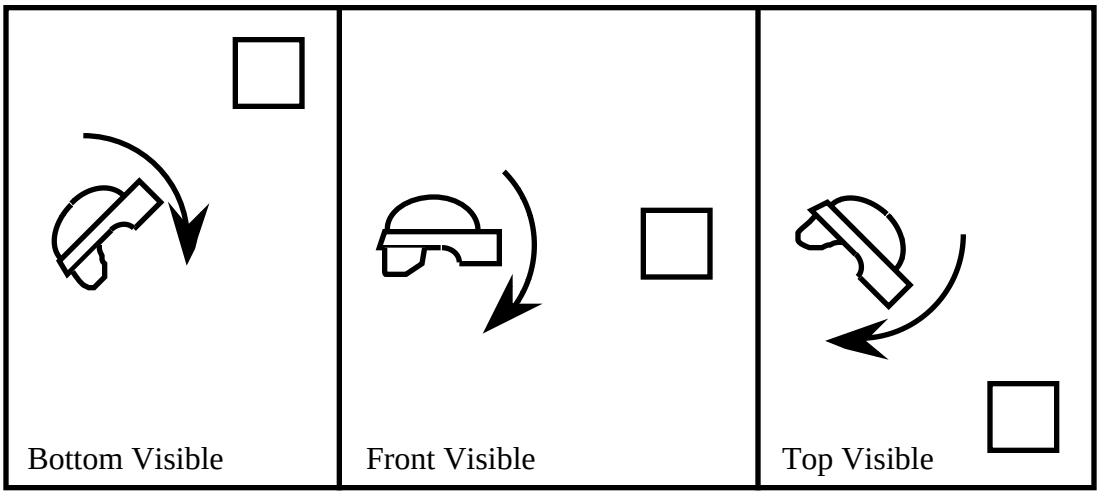

Figure 2: Orbital Mode

4.3. Worlds-In-Miniature

Worlds-in-Miniature (WIM) is a powerful technique which has applications in two areas: navigation and object manipulation. Worlds-in-Miniature interaction involves the manipulation of objects in a miniature representation of the world floating in front of the user, easily within arms reach (see Figure 3). Movement of an object in the WIM results in the corresponding movement of the object in the immersive environment surrounding the user. Movement of a representation of the user in the WIM results in movement of the user through the immersive environment.

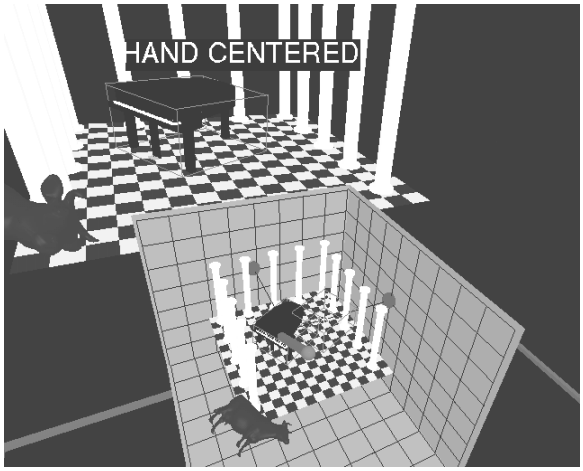

Figure 3: Worlds-in-Miniature

Interaction using a WIM extends your reach and allows you to manipulate objects naturally by reaching out and grabbing them. The WIM brings the world to the user (vs. extending the user's reach as in AAAD interaction). Since you are working with a miniature representation of the entire environment you can reach objects anywhere in the world and quickly move them across the entire environment. Working with a WIM has the advantage of giving both global and local context at the same time. You can see the entire virtual world in your WIM while you are immersed in your current location. This brings all objects within reach while simultaneously allowing you to perceive the full size object. Thus with a WIM it is possible to adjust a painting while standing across the room to see if it's straight. The WIM interface, however, does not solve the problem of direct manipulation and precise object positioning discussed above.

Techniques such as WIM and AAAD are particularly effective because they allow people to interact with the entire environment without having to move.

4.4. Menu Interaction

Two types of menuing systems are included in ISAAC: a rotary tool chooser and a two-dimensional menu.

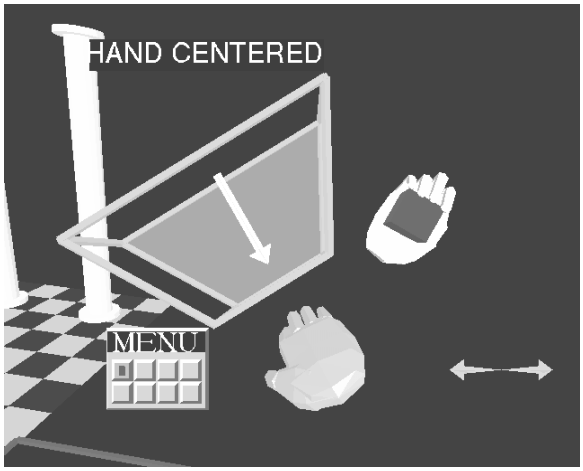

Figure 4: Rotary Tool Chooser

The rotary tool chooser is used for the selection of ISAAC's high level tools (fly, manipulate, scale) and to invoke the more complex two-dimensional menu. The rotary tool chooser is a one-dimensional menu based upon J.D. Liang's ring menu [Liang 1994]. A one-dimensional menu is one in which a user only has to vary one degree of freedom to move between menu selections (see [Mine 1995]). To select a tool the user simply invokes the tool chooser (by pressing a button on the input device) and then turns his hand about a chosen axis (like turning a dial). Rotation of the hand causes the tools (which are displayed in an arc about the hand when the tool chooser button is pressed - see Figure 4) to slide past a selection box. A tool is selected when it falls within the selection box. The important thing to note is that only changes along the chosen dimension are critical in the selection of the tool, all other changes are ignored. This simplifies interaction with the rotary tool chooser (but limits the number of options which can be handled in this way).

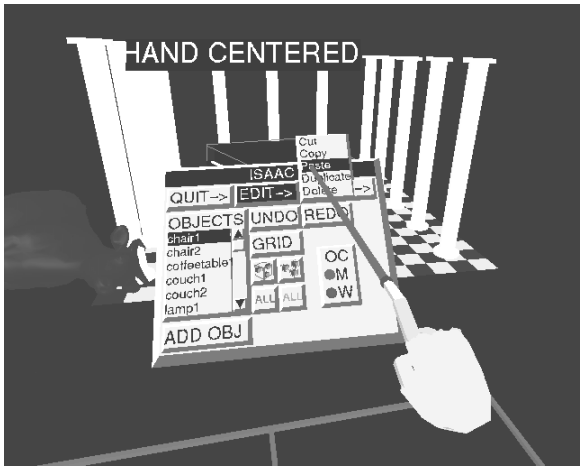

Figure 5: Two-dimensional Menus

More complex functions are included in ISAAC's two-dimensional menus. Two-dimensional menu are the virtual equivalent of conventional workstation's pull down menus and dialog boxes, except instead of being locked into screen space the menu floats in three-dimensional space (see Figure 5). Built upon UNC's Immersive Widgets library, ISAAC's two-dimensional menus include: single-shot buttons, on/off buttons, radio buttons, sliders, dials, scrollable lists, popup menus and displays (readouts for sliders and dials). Functions included on the two-dimensional menus include: edit functions (cut, copy, paste, duplicate, delete), grouping functions (group/ungroup), grid control functions (such as translational and rotational grid spacing), functions to add new objects, reset objects and other environmental controls. To simplify interaction with these menus a point-and-shoot paradigm is used. The menu is treated as a two-dimensional entity floating in space. The user invokes the various buttons and sliders using a laser beam emanating from his hand which

intersects with the plane of the menu. The user does not have to be able to reach a button to invoke it.

5. Constrained Object Manipulation

The absence of constraints when interacting with virtual objects is one of the major limitations in current virtual worlds applications. Users are restricted to gross interactions and are unable to perform precise object manipulation. Virtual solutions to real-world problems will depend upon the implementation of constraints to augment and control interaction within the virtual world.

I define two types of constraints that can be used in a virtual world application: virtual and physical. Virtual constraints are those in which extraneous degrees of freedom in the user's input data are ignored. Thus, for example, even though the user's hand moves in all three directions during object positioning, objects are constrained to move in a plane (ignoring any out of plane motions of the user's hand). Physical constraints are actual physical objects which are used to restrict the motion of the user's hand. Examples include a desktop on which the user can slide an input device or something like the University of Virginia's tracked tablet [Stoakley 1995], which the user holds in one hand and slides the input device on with the other hand.

The current focus of ISAAC system development is the implementation of virtual constraints. The goal is to incorporate constrained manipulation modes and grid functions which will help in the precise manipulation of objects. In designing ISAAC's virtual constraints, every effort has been made to keep interaction as natural and intuitive as possible. Direct manipulation techniques, for example, are enhanced to include controlled manipulation instead of providing sliders and dials on control panels. Several of the virtual constraints currently used by the ISAAC system are discussed below:

Snap-to-grid is used to place objects on evenly spaced grid intersections. The snap point (the point on the object aligned to the nearest grid intersection) is defined to be the center of the object's bounding box. The size of the grid increment can be set interactively using a slider on one of ISAAC's control panels. At present the grid is aligned with world coordinates though efforts are underway to include a user-definable grid which the user can position and orient interactively.

Snap-to-orientation is an important function which is used to re-orient selected objects to the nearest cardinal orientation (defined to be one of the 24 possible alignments of the object's model coordinates with the eight octants of the world coordinate frame). Snap-to-orientation is a corrective function, i.e. it is used after an object has been moved. Implementation of snap-to-orientation takes advantage of the fact that orientations in ISAAC are represented as quaternions and quaternions can be thought of as points on a four-sphere. With this in mind, if quaternion q_O represents the object's orientation and quaternion q_i represents one of the cardinal orientations, then the *nearest* cardinal orientation is the one in which:

$$\{q_i \ (1 \le i \le 24) \mid \mathrm{ABS}(q_O \bullet q_i) \text{ is closest to } 1\}$$

The absolute value function is necessary because a quaternion and its complement represent the same orientation (see [Shoemake 1985]).

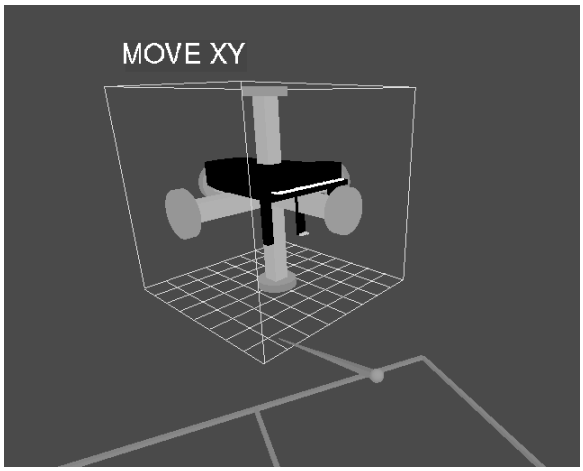

Figure 6: Constrained Motion Widgets

Three types of controlled motion widgets are used for the constrained manipulation of objects in ISAAC: one-dimensional translation, two-dimensional translation, and one-dimensional rotation (see Figure 6). A widget is defined to be a three-dimensional object with behavior [Conner 1992]. To use a constrained motion widget you first select a widget by pointing at it with the laser beam, *grab* it (indicated by pressing a button on the input device), and then (depending on the widget selected) use the widget to move the object along a line, in a plane or rotate it about one of its axes. The constrained motion widgets can either be aligned with the model's coordinate system or the world's coordinate system.

Careful consideration was given to the interaction between snap-to-grid and the constrained motion widgets. Initially it was decided that the snap-to-grid function should take precedence. Thus even if an object is being positioned using one of the constrained motion widgets, it would jump to the nearest grid intersection (often not on the constrained path) if snap-to-grid was on (see Figure 7). This resulted in erratic object motion and had the undesirable side effect that it was difficult to move an object back to its original position once moved.

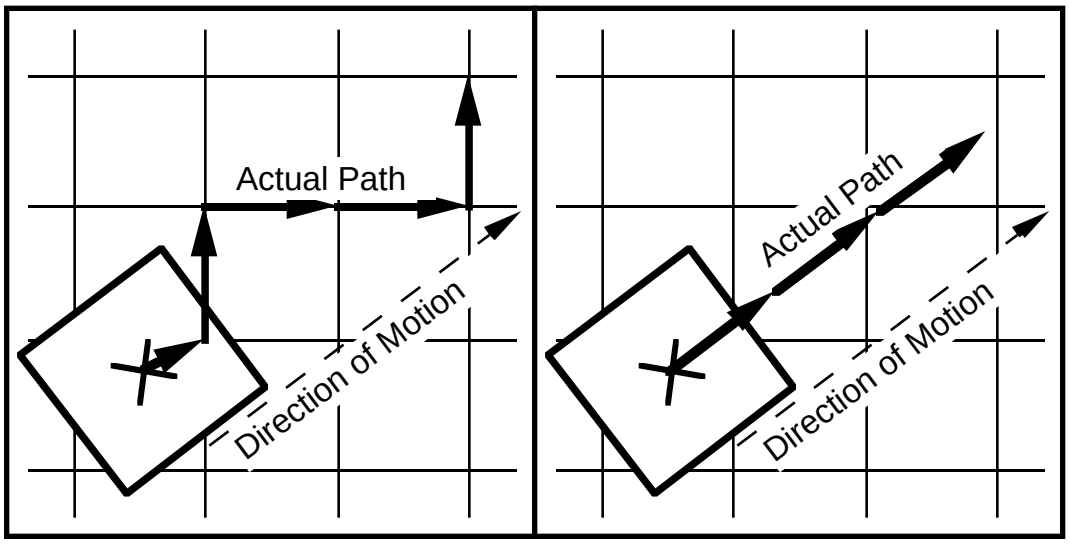

Figure 7: Snap-to-grid precedence vs. Constrained-motion-widget precedence

After further evaluation it was decided that it was more useful and intuitive if the constrained motion widgets took precedence over snap-to-grid. Thus an object moved using a constrained motion widget always stays on its constrained path.

To allow for more controlled motion an object-centered snap was implemented. This means that when using a constrained motion widget with snap-to-grid on, objects are constrained to move in even increments *relative to their starting position* (their location at the start of the grab - see Figure 7). The size of the motion increment matches the size of the current grid spacing. Since the constrained motion widgets can be aligned with either the world or model coordinate systems (see above), you can move an object precisely in world coordinates or relative to the object's current orientation. Using a model-aligned constrained motion widget, for example, you can move an object one unit to its left even if it is not aligned with the world coordinates.

6. Future Work

The primary focus of future work on ISAAC will be the implementation of additional forms of constrained interaction. This will include user definable grids (which can be moved interactively to any position and orientation) and additional constrained motion widgets (such as rotation-only and translation-only object-positioning widgets). Of particular importance will be the implementation of techniques for the specification of object-to-object alignments and constraints. Both user-specified and automatic techniques (such as the innovative techniques described in [Bukowski 1995]) will be explored.

The relative merits of distributed vs. localized control will be investigated. This will involve the investigation of the advantages and disadvantages of moving controls from ISAAC's control panels into the environment (to be registered with the objects they control). Grid spacing controls, for example, might be placed directly on their associated grid lines. Though this reduces the cognitive distance between object and control and is potentially more intuitive, it introduces additional problems of visibility and controllability (due to problems such as those associated with direct manipulation described in this paper).

Techniques for the management and control of the increasing number of interaction modes must be developed. The goal is to avoid extensive use of menus for mode specification and tool selection. Though a modeless interface is desirable, real-world applications will require numerous modes of interaction (such as the various modes of constrained object manipulation used in ISAAC) . The use of voice recognition is one possible avenue of exploration.

A final long term goal will be the incorporation of shape generation tools as part of an eventual move towards the implementation of an immersive modeling tool.

7. Conclusions

Before we can solve real-world problems in the virtual world we must gain a better understanding of the nature of the virtual environment. Current virtual worlds applications suffer because we are misusing this new medium we work in; we do not take full advantage its power and we are unsuccessful in compensating for its limitations. We must learn to better exploit the natural forms of interaction possible in the virtual environment by extending them in ways only possible in the virtual world. We must learn to compensate for the lack of haptic feedback and the absence of effective techniques for alphanumeric input, both essential for precise interaction with virtual objects.

The ISAAC system, presented in this paper, gives examples of the kind of effective techniques which must be developed if we are to create real-world applications for the virtual-world; techniques which take advantage of the unique characteristics of this exciting new medium.

References

Bukowski, R. W., C. H. Sequin (1995). Object Associations: A Simple and Practical Approach to Virtual 3D Manipulation, *ACM Computer Graphics: Proceedings of 1995 Symposium on Interactive 3D Graphics,* Monterey, CA, pp. 131-138.

Bowman, D. (1995). *Conceptual Design Space*. Available at WWW URL http://www.cc.gatech.edu/gvu/virtual/CDS.

Butterworth, J., A. Davidson, S. Hench, and T. M. Olano (1992). 3DM: A Three-Dimensional Modeler Using a Head-Mounted Display. *ACM Computer Graphics: Proceedings of 1992 Symposium on Interactive 3D Graphics*, Cambridge, MA, pp. 135-138

Chung, J. (1994). *Intuitive Navigation in the Targeting of Radiation Therapy Treatment Beams*. Ph.D. Thesis, University of North Carolina, Chapel Hill, NC.

Conner, D., S. Snibbe, K. Herndon, D. Robbins, R. Zeleznik, A. van Dam (1992). Three-Dimensional Widgets, *ACM Computer Graphics: Proceedings of 1992 Symposium on Interactive 3D Graphics*, Cambridge, MA, pp. 197-208.

Fuchs, H., J. Poulton, J. Eyles, et al. (1989). Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories. *ACM Computer Graphics: Proceedings of SIGGRAPH 89,* Boston, MA, pp. 79-88.

IST (Institute for Simulation and Training) (1995). *PolyShop* . Available at WWW URL http://www.vsl.ist.ucf.edu/~polyshop.

Liang, J., M. Green (1994). JDCAD: A highly interactive 3D modeling system. *Computers & Graphics: Proceedings of the Conference on Computer Aided Design and Computer Graphics*, Beijing, China, pp.499-506.

Mine, M. (1995). *Virtual Environment Interaction Techniques,* UNC Chapel Hill Computer Science Technical Report, TR95-018.

Shoemake, K. (1985). Animating Rotations Using Quaternion Curves, *ACM Computer Graphics: Proceedings of SIGGRAPH 85,* San Francisco, CA, pp. 245-254.

Stoakley, R., M. J. Conway, R. Pausch (1995). Virtual Reality on a WIM: Interactive Worlds in Miniature. *Proceedings of SIGCHI 1995,* (to appear).