ErgoDesk: A Framework for Two- and Three-Dimensional Interaction at the ActiveDesk

Andrew S. Forsberg, Joseph J. LaViola Jr., Robert C. Zeleznik

Brown University Site of the NSF Science and Technology Center
for Computer Graphics and Scientific Visualization
PO Box 1910, Providence, RI 02912 USA

1 Abstract

We present ErgoDesk, a software and hardware framework for interacting at an ActiveDesk, a rear-projected drafting table-sized display. ErgoDesk represents a new interaction paradigm, based on physical props, multimodal input and a stereo-on-demand display surface, that seamlessly supports transitions between a variety 2D and 3D interaction techniques. We provide a detailed discussion of the ErgoDesk framework's major features including: the layout of physical props; the transitions between 2D and 3D interactions; the use of gestural and spoken input; and the transformation model for converting between real-world and virtual environment coordinates. We also discuss the design of ErgoSketch, a conceptual 3D modeling application built on the ErgoDesk framework, and our usability experiences with both over the past year.

Keywords: ActiveDesk, Sketch, 3D Interaction, 3D Modeling, Stereoscopic Viewing, Two-handed Interaction

2 Introduction

ErgoSketch is a conceptual modeling system that adapts the Sketch interface [18][19] to the ErgoDesk framework. The ErgoDesk framework is based on the concept of an ActiveDesk (see Figure 1), a variant of the responsive workbench [13], that minimally supports 2D pen-based gestural interaction, 3D interaction, and stereoscopic 3D viewing. The ErgoSketch name derives from the design goal to extend the 2D gestural Sketch application to a more ergonomic physical setup. This setup not only supports natural 2D input, since users draw directly on the display surface with a pen, but also enables seamless transitions between 2D and 3D interaction as dictated by specific interaction tasks.

This paper discusses several aspects of the design of the ErgoDesk framework including:

Figure 1: A user creates 3D geometry at the ActiveDesk with a pen and performs camera operations using the 3D tracker in his non-dominant hand.

- seamless transitions between tools
- 2D pen-based input
- 3D tracked input
- a device layout that supports two-handed interactions
- speech input

Since the design of the ErgoDesk framework is closely tied to the ErgoSketch application, our discussion of the details of ErgoDesk will be given in the context of ErgoSketch. In particular, the the discussion of ErgoDesk will cover specific ErgoSketch tasks including: 3D modeling with 2D gesture lines (as in Sketch [18]), non-dominant hand camera control, stereoscopic model examination in 3D (using an "object-in-hand" metaphor), stereoscopic model annotation in 3D, toolglasses and magic lens interaction [1]. In addition, we will discuss the calibration procedure we are using with our magnetic trackers and how we transform from the ActiveDesk's coordinate system to the virtual environment's coordinate system. The following sections describe each of these components of the framework in further detail.

3 Seamless Transitions Between Tools

Users switch between a variety of different tools in most modeling applications. It is important to support a seamless transition between these tools to enable an effective interface dialog such that the user may concentrate on the modeling task instead of the tools themselves. We have found that the novel hardware configuration and predominantly gestural input style of ErgoDesk enable more compelling and more seamless transitions than conventional interfaces.

There are several types of transitions in ErgoSketch that appear seamless to the user when contrasted with traditional desktop-style modeling systems. First, transitions between physical tools that support specific tasks are seamless because the tools are positioned on the table top within reaching distance. To switch tools, the user simply puts down one tool and picks up another. Each physical object has a specific purpose just as a pencil, eraser, ruler, and protractor do in traditional drawing. Second, some physical tools serve multiple purposes; in these cases the selection of logical tools is also seamless. For example, a 3D tracker can act as a 3D camera manipulator, a 3D annotation tool, or as a magic lens[1]. The transition between logical tools is accomplished seamlessly by speaking to the tool or automatically based on context. Third, virtual tools are activated and deactivated by a drawn gesture, a spoken command, by selection of a different virtual object, or, less naturally, through a 2D menu selection. A colorpicker, for example, can be activated through selection in a 2D widget-based menu, by drawing a “C” gesture, by speaking “colorpicker”, or by selecting the colorpicker object if it happens to be on the display. Fourth, the system may change aspects of the environment automatically when it perceives certain actions of the user. For example, the world is displayed monoscopically when the user draws 2D lines with a lightpen tool, but when the system recognizes that the user has switched to a tracked 6 DOF proxy tool, the display automatically switches to stereoscopic viewing.

ErgoSketch has a range of 2D and 3D interactions each tailored to best support specific modeling tasks. For example, basic tasks such as object instantiation, object editing, and camera manipulation are accomplished using a lightpen to draw 2D gesture lines that are interpreted by the interface. In addition, a trackball can also be used to manipulate the virtual camera. Other tasks such as virtual object inspection and some forms of object annotation are performed in 3D, closely simulating how such tasks would be performed in the real world. When drawing 2D gestures on a 3D model at the ActiveDesk, a stereoscopic view is problematic because the lightpen can only interact with the display surface. Furthermore, if the 3D model is “above” the display surface, then the lightpen can break the user’s illusion of the object floating above the desk surface if it is positioned inside or behind the object. Consequently, a monoscopic view is required when drawing 2D gesture lines. However, when interacting in 3D, a stereoscopic view is a more effective way to perceive a 3D model. Therefore, we support a seamless, dynamic transition between monoscopic and stereoscopic viewing depending on which type of tool is being used. In general, environmental transitions are automatically triggered when a particular tool is activated by the user.

3.1 2D Pen-based Input

The Sketch interface [18] is an effective gestural interface for creating and editing 3D conceptual models. Sketch’s first implementation used a conventional workstation setup (a three-button mouse, a monitor, and a keyboard) for input and output. However, the indirect nature of the mouse and monitor display is not ideal for the Sketch interface which interprets “drawn” lines and 2D gestural input to directly manipulate objects and the view.

The ActiveDesk’s drafting table-like design and large display surface augmented with a lightpen naturally supports direct drawing and direct manipulation. There are three buttons on our lightpen: one is triggered when the pen is depressed on the display surface and two others are positioned such that the index finger can press either. We follow the convention of the sketch system where one button is used for drawing gestures, one button is used to manipulate objects, and the third button is used to modify camera parameters (zoom, pan, rotate).

3.2 3D Tracked Input

While sketching in 2D is an effective way to create 3D models, it does not support some inherently 3D tasks such as rapidly viewing an object from different sides. A user might perform this task when examining an object or discussing it with another person. We support this type of interaction through the use of a physical prop that acts as a proxy for a virtual object (similar to [8]). The physical prop is a 6 DOF tracker that normally is attached with Velcro to the edge of the desk surface. When the user picks up the prop, two things happen: first, the object they were designing becomes “attached” to the end of the tracker and interactively translates and rotates as the user manipulates the tracker. Second, the rendering mode dynamically switches to a stereoscopic rendering mode to heighten the illusion of examining a 3D object. In addition to examining the object, the user may use a second 3D tracker to draw 3D virtual annotations around the object.

When the user puts down the tracker that acts as a proxy for the virtual object, the rendering mode transitions back to a monoscopic display. The user may then pick up the lightpen tool to modify the existing model or create new geometry.

3.3 Device Layout Supporting Two-handed Interaction

There are many situations where two-handed interaction is more effective than one handed interaction. Recent work has demonstrated the benefits of this style of interaction [3][7][14] [16][19]. Our layout of physical devices around the ActiveDesk encourages several types of two-handed interactions. When the user is drawing with the lightpen, the non-dominant hand can manipulate a trackball which can be used to rotate and zoom the camera view with respect to the lightpen. When the user has "picked up" a virtual object with a 3D tracker, the second hand can use a second tracker to create 3D annotations with respect to the first hand– an action that closely mimics similar real world two-handed interactions.

3.4 Speech Input

Speech naturally augments many modeling and general interaction operations. For example, simply saying "colorpicker" can activate a colorpicker widget. In addition, speech can take the place of the keyboard for specifying commands or possibly simplify the task of selecting from a large group of items. Recent advances in speech recognition technology are resulting in robust speaker independent speech recognition for small vocabularies which is nearly adequate for our requirements at this time. We are using a speaker dependent system, [9], in our framework, but are working on integrating a speaker independent system [6]. In most situations, speech is not an exclusive means for performing an operation. The colorpicker widget can also be activated through a drawn gesture (by drawing a "C").

Spoken commands can be used in the context of operations performed by virtual tools. For example, when the user is holding a tool (e.g., a 3D tracker or the lightpen), he or she can specify additional operations or parameters by speaking. Voice commands are useful in situations where alternative resources are not available. For example, when annotating an object in ErgoSketch both of the user's hands are occupied (i.e., the non-dominant hand is holding the object and the dominant hand is "drawing" 3D annotations). In this situation, spoken commands provide a way to clear unwanted annotations which previously was done by pushing a button on a keyboard. As another example, the user can issue the voice command "copy" while pointing to an object to duplicate it or speak a color name to change its color.

4 Magnetic Tracker Calibration and Transformations

There are two aspects to the integration of 6 DOF magnetic trackers into our framework for the ActiveDesk. The first is a calibration step which converts *tracker coordinates* into *desk coordinates*. The second is a transform that maps points in the *desk* coordinate system to points in the virtual environment's (VE's) coordinate system. This transform is used by the application programmer to easily and intuitively position objects in "3D" in the viewing volume above, on, and below the desk surface. This is especially useful if the application supports general navigation (e.g., translation, orientation, and zooming) through the VE.

The following two subsections describe our calibration procedure and the coordinate transform from desk's coordinate system to VE's coordinate system.

4.1 Calibration

We use magnetic tracking technology to perform 3D interactions at the ActiveDesk. To incorporate 3D trackers, it is necessary to calibrate the tracking system because, in general, the tracking system's coordinate system is not aligned with the ActiveDesk's coordinate system. However, once calibrated the system does not have to be recalibrated unless the ActiveDesk or the magnetic transmitter is moved.

Figure 2 depicts the six points used to define the ActiveDesk's coordinate system. To calibrate the desk, the user positions the tracker at each of these six points. Using these samples, a transform (see derivation below) from tracker coordinates to *desk* coordinates is defined. During the calibration, points 1 through 5 are graphically drawn on the desk surface to show the user where to position the tracker. Point 6 is positioned at approximately the user's eye-level and perpendicular to point 5. Although the height of point 6 above the desk is arbitrary, we have found it convenient to choose an average user's eye height. We typically use a wooden box as a tool to aid in positioning the tracker perpendicular to the display surface when specifying point 6.

The *desk* coordinate system is defined as follows: the origin (0,0,0) is at the center of the display surface; the x-axis points to the *right* and is in the plane of the display surface; the y-axis is perpendicular to the x-axis and in the plane of the display surface; and the z-axis is perpendicular to the x-axis and the y-axis (according to the right-hand rule). The upper-right, upper-left, lower-left, and lower-right corners of the desk are mapped to the coordinates (1,1,0), (-1,1,0), (-1,-1,0), and (1,-1,0), respectively. Point 6 is mapped to the coordinate (0,0,1). Although unit distances along the X, Y and Z axes are different, this is not an issue for the programmer who typically uses *desk* coordinates only to

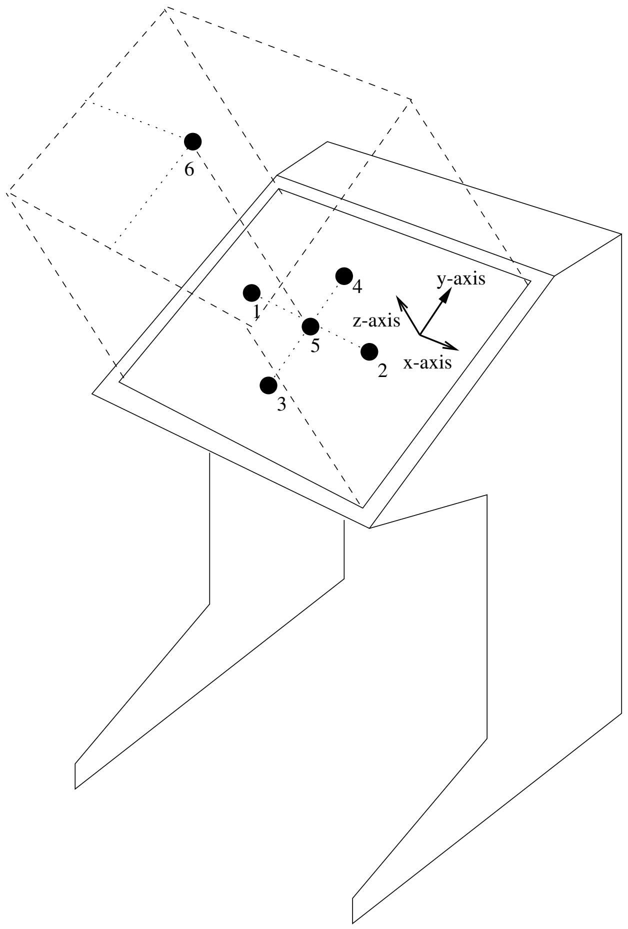

Figure 2: The six points shown are used in the calibration procedure to define the ActiveDesk's coordinate system. Point 5 is mapped to the origin (0,0,0); the x-axis points in the direction from point 1 to 2; the y-axis points in the direction from point 3 to 4; and the z-axis points in the direction from point 5 to 6.

specify positions relative to the unit cube. Figure 3 illustrates the ActiveDesk's coordinate system by showing the coordinates of several key 3D points around the viewing area. We now show the derivation of $T_{\text{desk}\leftarrow\text{tracker}}$, the transformation from tracker to *desk* coordinates.

We define each point n from Figure 2, notated P_n, in terms of a fraction [1], $\frac{1}{f}$, of the desk surface:

$$P_{1_{desk}} = (-\frac{1}{f}, 0, 0) \tag{1}$$

$$P_{2_{desk}} = (\frac{1}{f}, 0, 0) \tag{2}$$

$$P_{3_{desk}} = (0, -\frac{1}{f}, 0) \tag{3}$$

[1] Although this fraction can take any value between zero and one, it is practical to choose a value that is far enough from the origin to minimize the effect of both tracker noise and human placement error, but close enough to the origin that the sampled position is robust (i.e., not on the fringe of a magnetic tracker's range where there is a lot of noise). f can be any small positive integer– we use a value of 5.

$$P_{4_{desk}} = (0, \frac{1}{f}, 0) \tag{4}$$

$$P_{5_{desk}} = (0, 0, 0) \tag{5}$$

$$P_{6_{desk}} = (0, 0, 1) \tag{6}$$

We can represent the principal unit vectors of the desk coordinate system (i.e., the vectors $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$– see Figure 3) in tracker coordinates:

$$\vec{X} = \frac{f}{2} \cdot (P_{2_{tracker}} - P_{1_{tracker}}) \tag{7}$$

$$\vec{Y} = \frac{f}{2} \cdot (P_{4_{tracker}} - P_{3_{tracker}}) \tag{8}$$

$$\vec{Z} = (P_{6_{tracker}} - P_{5_{tracker}}) \tag{9}$$

The transformation from *desk* to *tracker* coordinates is then given by the column matrix:

$$T_{\text{tracker}\leftarrow\text{desk}} = \begin{bmatrix} \vec{X}_x & \vec{Y}_x & \vec{Z}_x & P_{5_{tracker_x}} \\ \vec{X}_y & \vec{Y}_y & \vec{Z}_y & P_{5_{tracker_y}} \\ \vec{X}_z & \vec{Y}_z & \vec{Z}_z & P_{5_{tracker_z}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Finally, the transformation from *tracker* to *desk* coordinates is the inverse:

$$T_{\text{desk}\leftarrow\text{tracker}} = (T_{\text{tracker}\leftarrow\text{desk}})^{-1} \tag{11}$$

It is interesting to note that in our current system, we do not make any attempt to compensate for static distortion of the magnetic field. However, we have found that the error in the reported samples was essentially undetectable for our application when the magnetic transmitter was positioned about 1.5 feet off the ground, behind the user, and as close to the front of the desk as possible (approximately 2 feet from the front of the ActiveDesk). If greater precision is required, an additional and more involved calibration step is necessary to eliminate static error (i.e., the error in tracked values due to the distortion of the magnetic field due to objects in the physical environment). There are several approaches to reducing static error such as [2].

4.2 Transforming Between Desk and Virtual Environment Coordinates

ErgoSketch uses another transformation between *desk* and VE coordinates to support object placement within the desk viewing volume and to navigate within the VE. By converting between *desk* and VE coordinates it is easy to position an object relative to the *desk* coordinate system. For example, we can "attach" virtual objects to 3D trackers that report values within the unit cube volume above the desk (see Figure 3) thereby causing the object to appear to float above the desk.

In addition, the *desk* to VE transformation simplifies the programmers interface to moving through the VE. This is similar to the platform concept presented in [17].

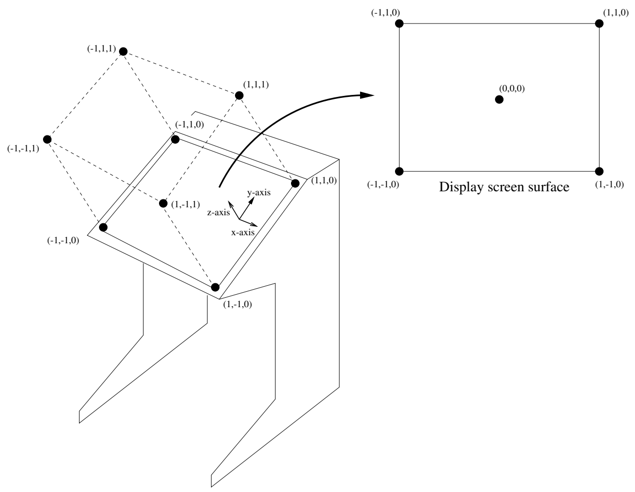

Figure 3: A sampling of points in the ActiveDesk's coordinate system.

5 Usability Discussion

Over the past few months we have gathered information about the usability of the basic ErgoDesk framework and ErgoSketch modeling application. Although we are primarily interested in the usability of the *framework*, most of our results have been complicated by details of the *application* that uses the framework. In this section we discuss both of these issues in more detail.

In terms of the ErgoDesk framework, we found subjects could easily draw 2D gestures, switch between devices, and transition to visualizing objects stereoscopically. With only a brief explanation it was clear what each tool was for and what effect it could have on the environment. However, subjects had difficulty using the speech recognition which frequently misinterpreted spoken commands, had a limited vocabulary and grammar, was affected by ambient noises, and frequently required the user to carefully enunciate in a constant tone of voice. Some subjects did not like the fact that some devices were fixed or cumbersome to move around in the environment (e.g., the trackball). Further, two-handed interaction was limited to essentially sequential operations (e.g., camera manipulation with the non-dominant hand followed by drawing with the dominant hand). 2D drawing was hindered by the use of a lightpen instead of a cordless device– although cordless solutions exist from ITI[10]. Finally, subjects expected to be able to use their hands and fingers to *directly* interact with objects on the display surface.

In terms of the ErgoSketch modeling application, users had difficulty creating interesting 3D models. This is in part due to the nature of learning a gestural user interface. The underlying modeling functionality was also relatively incomplete compared to full-featured commercial modeling systems. Lastly, users were strongly affected by the many deficiencies in the underlying hardware (i.e., display blurriness, the tethered lightpen and its noisy data, and difficulty drawing in 3D).

6 Future Work

There are several extensions we would like to add to ErgoSketch that our current framework does not support. We discuss three issues that we will research in order to expand our framework and extend the ErgoSketch system.

At this time, we are not tracking the user's head position. However, head tracking would result in an improved 3D viewing experience because motion parallax resulting from head motions is a principle means of extracting depth from a scene by the visual system. The user should be able to move their head from side to side when viewing objects in the stereoscopic mode as means for examining objects from different perspectives. While magnetic trackers could be attached to the user's head in some way, we would prefer to use an unobtrusive camera-based solution. [12] demonstrates such a system for head tracking is robust in various lighting conditions and that no calibration is necessary after the tracking algorithm has begun running. The main advantage of the camera-based solution is that no cabling or method for mounting a tracker on a user is required. The effects of the accuracy of a camera-based head tracker and the sampling rate may be a problem requiring further exploration.

We will also expand the role of speech input in the interface. We are currently supporting simple voice commands. There are situations in which context could further be leveraged and voice commands can be more tightly coupled with individual physical and virtual tools. The QuickSet system [4][11] serves as an example of how speech and drawn gestures can be integrated effectively. In addition, we found most speech commands were spoken in an unnatural way (i.e., overly enunciated) and we believe a much more informal style of communication is necessary.

Lastly, except for 3D annotations, geometry can only be created using 2D gesture lines created with the 2D lightpen. We want to support the generation of both free-form and precise 3D geometry through 3D hand gestures. To support this functionality, we will add glove devices or camera-based hand tracking to the existing framework and develop interaction techniques to create geometry from 3D hand gestures. The interface for these operations will most likely be multi-modal interactions incorporating speech to modify hand gestures or state the type of geometry to create.

7 Conclusion

We believe ErgoSketch is representative of the next generation of modeling systems which has an interface more appropriate for designers and that reuses learned skills. The ErgoSketch system is built on top of the framework we have developed which enables both 2D and 3D interaction at a projection table and supports seamless transitions between the interactions and tasks.

The prototype system we have developed using the framework has some inadequate implementations. However, many of these problems can be eliminated by purchasing appropriate hardware. For example, a large untethered see-through tablet in place of lightpen technology is less obtrusive way to draw on the desk surface and eliminates the need to calibrate the lightpen. In addition, tablet technology, in theory, supports the tracking of multiple devices that can have unique identifiers which would support a range of interaction possibilities such as personal pens, "Pick-and-Drop" user interface (see [15]), and simplifying the implementation and use of specialized physical tools.

In summary, while the ErgoSketch application was difficult for many users to operate, the underlying ErgoDesk framework appears to be more promising. We believe that many applications can benefit from the ErgoDesk's integration of 2D gestural input, multiple physical tools, transitions between 2D and 3D interactions as well as display modes, and speech input.

8 Acknowledgements

This work is supported in part by the NSF Graphics and Visualization Center, Alias/Wavefront, Autodesk, Microsoft, Sun Microsystems, and TACO. We especially thank Bill Buxton of Alias/Wavefront for donating the ActiveDesk to our research group.

9 References

[1] Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T., "Toolglass and Magic Lenses: The See-through Interface", *In Proceedings of SIGGRAPH '93*, pp. 73-80, August, 1993.

[2] Bryson, S., "Measurement and Calibration of Static Error for Three-Dimensional Electromagnetic Trackers", *SPIE Conference on Stereoscopic Displays and Applications*, pp. 244-255, San Jose, CA, 1992.

[3] Buxton, W., and Myers, B. A., "A Study in Two-Handed Input." *In Proceedings of CHI'86 Human Factors in Computing Systems*, pp. 321-326, 1986.

[4] Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith I., Chen, L., and Clow, J., "QuickSet: Multimodal Interaction for Distributed Applications", *In Proceedings of the 5th ACM International Multimedia Conference*, Seattle, WA, pp. 31-40, November, 1997.

[5] Forsberg, A. S., LaViola, J. J., Markosian, L., Zeleznik, R. C. (1997), "Seamless Interaction in Virtual Reality", *IEEE Computer Graphics and Applications*, 17(6), pp. 6-9, November/December 1997.

[6] Hark Recognizer Reference Manual, BBN, Inc. Cambridge, MA 1994.

[7] Hinkley, K., Pausch, R., Proffitt, D., Patten, J., and Kassell, N. "Cooperative Bimanual Action." *In Proceedings of CHI'97 Human Factors in Computing Systems*, pp. 27-34, 1997.

[8] Hinkley, K., Tullio, J., Pausch, R., Kassell, N. "Usability Analysis of 3D Rotation Techniques." *In Proceedings of User Interface Software and Technology '97*, pp. 1-10, 1997.

[9] The In-Cube User Guide, available from Command Corporation, Inc., Atlanta, GA, 1997.

[10] Input Technologies, Incorporated., http://www.iti-world.com.

[11] Johnston, M., Cohen, P. R., McGee D., Oviatt, S. L., Pittman, J. A., Smith I., "Unification-based Multimodal Integration", *35th Annual Meeting of the Assoc. for Computational Linguistics and 8th Conference of the European Chapter of the Assoc. for Computational Linguistics*, Madrid, Spain, July 7-12, 1997.

[12] Khuner, H., Personal communication and in print.

[13] Krueger, W. and Frohlich, B., "The Responsive Workbench", *IEEE Computer Graphics and Applications*, pp. 12-15, 1994.

[14] Mapes, D. J., and Moshell, M. J., "A Two-Handed Interface for Object Manipulation in Virtual Environments." PRESENSE Teleoperators and Virtual Environments, 4(4), pp. 403-416, Fall 1995.

[15] Rekimoto, J., "Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments", *In Proceedings of User Interface Software and Technology '97*, pp. 31-39, Banff, 1997.

[16] Shaw, C. and Green M., "THRED: A Two-Handed Design System.", *In Multimedia Systems Journal*, Volume 5, Number 2, ACM/Springer Verlag, 1997.

[17] Sowizral, H., Rushforth, K., Deering, M., *The Java 3D API Specification*, Addison Wesley Longman, Inc.: Reading, MA, 1998.

[18] Zeleznik, R.C., Herndon, K., Hughes, J. "Sketch: An Interface for Sketching 3D Scenes." *In proceedings of SIGGRAPH'96*, pp. 163-170, 1996.

[19] Zeleznik, R.C., Forsberg, A.S., and Strauss, P.S., "Two Pointer Input for 3D Interaction." *In Proceedings of the 1997 Symposium on Interactive 3D Graphics*, Providence, RI, April, 1997.