

# Rapid Controlled Movement Through a Virtual 3D Workspace

Jock D. Mackinlay, Stuart K. Card, and George G. Robertson

Xerox Palo Alto Research Center\*

## Abstract

Computer graphics hardware supporting real-time interactive 3D animation has the potential to support effective user interfaces by enabling virtual 3D workspaces. However, this potential requires development of viewpoint movement techniques that support rapid and controlled movement through workspaces. Rapid movement through large distances avoids wasted work time; controlled movement near target objects allows the user to examine and interact with objects in the workspace. Current techniques for viewpoint movement typically use high velocities to cover distances rapidly, but high velocities are hard to control near objects. This paper describes a new technique for targeted viewpoint movement that solves this problem. The key idea is to have the user indicate a point of interest (target) on a 3D object and use the distance to this target to move the viewpoint logarithmically, by moving the same relative percentage of distance to the target on every animation cycle. The result is rapid motion over distances that slows as the viewpoint approaches the target object. The technique can be used with 2D and multidimensional input devices. We also extend the technique to move objects in the workspace.

**CR Categories and Subject Descriptors:** I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction techniques; D.2.2 [Software Engineering]: Tools and Techniques - User interfaces

**Additional Key Words and Phrases:** Viewpoint movement, object movement, virtual reality, interactive graphics, 3D graphics, logarithmic motion, 3D workspaces

## 1 INTRODUCTION

Advances in computer graphics hardware have enabled the practical realization of real-time interactive 3D animation systems. These systems have the potential to provide simulated 3D workspaces for user interaction with CAD/CAM, medical information, scientific visualization, "artificial reality", and general information access. An important require-

\*3333 Coyote Hill Road, Palo Alto, CA 94304, 415-494-4335, Mackinlay@Xerox.com

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

ment for such systems is a technique that allows the user to move the viewpoint (1) rapidly through large distances, (2) with such control that the viewpoint can approach very close to a target without collision. We call this the problem of *rapid and controlled, targeted 3D viewpoint movement*. This problem arises in large information spaces, such as for complex machine parts in a CAD system or in simulated landscapes. Large information spaces contain numerous objects and/or highly detailed objects that require the user to move back and forth from global, orienting views to manipulate detailed information.

Current techniques for moving the viewpoint are not very satisfactory for targeted viewpoint movement. Some techniques fail to support rapid movement because of inefficient interactions or movement trajectories. Techniques supporting rapid movement either use coarse-grained scale factors for direct positioning of the viewpoint over large distances or use high velocities for flying the viewpoint rapidly through large distances. Coarse-grained scale factors do not allow fine-grained control and high velocity flight is difficult to control once a target object is reached.

This paper describes a new, more effective technique for targeted 3D viewpoint movement. The key idea is to have the user select a 3D point of interest (the target) on the surface of an object. On each animation cycle, the user's viewpoint is moved the same relative percentage of the distance to the target, resulting in an approach that is rapid for large distances, but logarithmically slower as the target becomes closer. Since the technique only requires the mouse or another 2D input device, it integrates with existing interfaces and work environments. It can also be used with multidimensional input devices. In the paper, we summarize current viewpoint movement techniques, describe the *Point of Interest* logarithmic movement technique, and show how the ideas can be extended to include general object movement.

## 2 3D VIEWPOINT MOVEMENT

Developing an effective technique for 3D viewpoint movement is difficult for several reasons. One problem is the number of parameters to be controlled by the user. 3D viewpoint movement involves at least six degrees of freedom: three dimensions for position and three dimensions for rotation. The actual number of parameters depends on the movement metaphor, which typically involves either the direct positioning of the viewpoint in the workspace or the flying of the viewpoint through the workspace. Direct positioning metaphors typically involve a scale factor parameter for the input device and flying metaphors typically involve

velocity or direction parameters. Specialized tasks can involve additional viewpoint parameters (typically associated with the viewing matrix). For example, a cinematographic application might have a parameter for controlling the zoom of the field of view.

Another problem is the type of viewpoint movement required by a given task. We can distinguish at least four types of viewpoint movement for interactive 3D workspaces:

*General movement.* Exploratory movement, such as walking through a simulation of an architectural design.

*Targeted movement.* Movement with respect to a specific target, such as moving in to examine a detail of an engineering model.

*Specified coordinate movement.* Movement to a precise position and orientation, such as to a specific viewing position relative to a molecule or a CAD solid model.

*Specified trajectory movement.* Movement along a position and orientation trajectory, such as a cinematographic camera movement.

A technique appropriate for targeted movement—the focus of this paper—may not be appropriate for another type of movement. For example, general exploratory movement may proceed at a relatively uniform speed, and it may not be necessary to approach very close to objects. In that case, a technique based on the metaphor of walking or driving a car may be satisfactory even though it is relatively slow and is hard to control near objects.

In addition to the inherent difficulties of controlling the viewpoint parameters, we also desire a technique that satisfies the following general interface requirements: (1) is easy to use, (2) prevents user disorientation, (3) integrates with other user interface and work environments, and (4) supports the perception of the virtual workspace.

### 3 CURRENT VIEWPOINT MOVEMENT TECHNIQUES

3D viewpoint movement can be accomplished by either moving the viewpoint through the workspace [1,4,7,8,11,14] or by using object movement techniques [1,2,3,5,6,9,11,13,14,15] to move the workspace around the viewpoint. This section focuses on current viewpoint movement techniques and discusses difficulties they have supporting targeted viewpoint movement. Experimental evidence suggests that the object movement approach does not work very effectively in complex multi-object workspaces [14]. Furthermore, except for Bier's snap-dragging technique [3], object movement techniques also have the problems described in this section, and Bier's gravity function for rapid movement of objects to targets does not support controlled movements near targets.

Current viewpoint movement techniques exhibit one or more of three basic difficulties in carrying out targeted movement: (1) inefficient interactions and movement trajectories, typically caused by 2D input devices; (2) limits on human reach and precision when the technique is based on directly positioning the viewpoint; and (3) difficulties controlling high velocities when the technique is based on flying or steering the viewpoint through the workspace.

*Inefficient interactions and movement trajectories.* Many techniques (typically based on 2D devices) require the user to accomplish a movement by shifting back and forth among simple movement modes [7,8,11]. For example, the Jack system [11] for manipulating articulated figures uses a menu

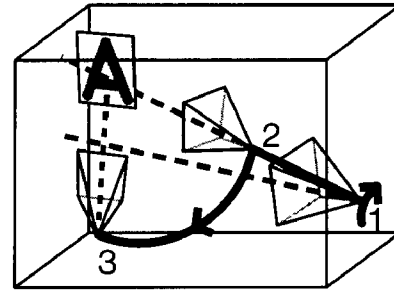


Figure 1: To move the viewpoint (shown as pyramids in the diagram) toward an object in the Jack system [11], the user must (1) pan the line of sight to face the object, (2) zoom to the desired distance, and (3) sweep to the desired orientation.

for assigning the mouse to “sweep”, “pan”, or “zoom” operations. The interaction is non-optimal because the menu must be used frequently to assign the mouse to different operations. Figure 1 also shows that the movement trajectories are inefficient because the user cannot move directly to the point of interest. Movement interfaces based on “dial boxes” also result in inefficient movement trajectories because users generally adjust only one dial at a time.

*Limits on human reach and precision.* Instead of using a low-dimensional transducer, such as a mouse, and shifting it among the multiple parameters of control, a common alternative tactic uses a six degree of freedom input device, such as the Polhemus cube, to position the viewpoint directly in the workspace [1,4,14]. Unfortunately, psychophysical constraints limit direct positioning techniques from simultaneously supporting rapid and controlled movements. For example, when the input device is scaled so that rapid movements can be accomplished in the span of the human arm, hand tremors make fine-grained controlled movements difficult. A more fine-grained scale factor requires ratcheting techniques [14] to provide a full range of motion, leading to inefficient interactions. Therefore, this alternative tactic does not handle the large distances and very close positioning of targeted movement.

*Difficulties controlling high velocities.* Some techniques address the problem of rapid movement over distances by providing users with velocity controls [7,8,14]. The difficulty is that a high velocity for covering distances rapidly is difficult to control near the target. For example, Figure 2 plots three typical user strategies for controlling the velocity while approaching an object over a distance. These plots are functions of the form

$$f(t) = v_u t$$

where the user controls the parameter  $v_u$  over time  $t$ . The strategies are: (1) *slow&sure*, (2) *fast&overshoot*, and (3) *pulse*. The *slow&sure* strategy uses a low velocity that allows the user easy control near the object. However, it takes a long time. The *fast&overshoot* strategy uses a high velocity that covers distance rapidly. However, the user will probably overshoot the point of interest because of delays due to human reaction time and the discrete timing of animation frames (shown as vertical lines). After pausing to react to the overshoot, the user moves in the opposite direction and inefficiently oscillates the viewpoint into the desired

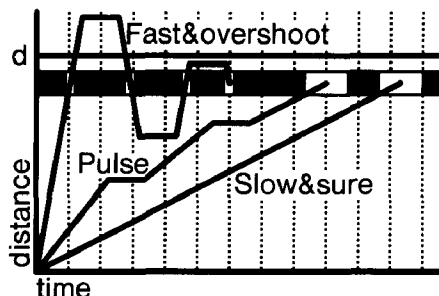


Figure 2: Motion functions of the form  $f(t) = v_{ut}$  for three user strategies for controlling the velocity of movement toward a target. The horizontal line at distance  $d$  represents a point of interest, the gray band represents the desired viewpoint range near the point of interest, and the vertical lines represent animation frames. The slow&sure strategy has a wide intersection with the desired viewing range (shown in white) making it easy for the user to stop. The fast&overshoot strategy has a narrow intersection which requires several attempts to stop. The pulse strategy involves pauses while the user plans the magnitude of the next pulse.

viewing region. The pulse strategy uses little spurts of velocity to step towards the object. However, the user must pause to assimilate the effect of each pulse and the resulting movement takes a long time.

#### 4 POINT OF INTEREST MOVEMENT

This section describes a new viewpoint movement interface, called *Point of Interest movement*, that was developed by also considering the object that is the target of the user's desire to move the viewpoint. A Point of Interest (POI) is a location on the surface of an object in the workspace. POI movement comes in two versions: *basic POI movement*, which moves the viewpoint toward (or away from) the POI, and *orienting POI movement*, which moves the viewpoint and also orients it to face the POI for improved viewing and interaction.

##### 4.1 Moving the viewpoint toward a POI

Basic POI movement requires the user first to indicate a POI on the surface of a target object and then to initiate motion toward (or away from) that POI. The user indicates a POI to the system by using the mouse cursor to select a target object in the 3D workspace. When the user pushes a mouse button, the viewing transformation is inverted and a ray is cast into the 3D workspace. The closest object pierced by this ray determines the POI and a circle is drawn on the surface of the object as feedback to the user. Since the feedback might indicate that the POI is not placed at the desired location, the user can interactively adjust the mouse cursor while the mouse button is pushed and adjust the POI along the surface of the object. The interaction is very natural. Just as the human eye rotates through a small visual angle to adjust to a point of interest, the mouse cursor can move through a small distance. While these small adjustments are being made, the POI changes position rapidly and automatically in the 3D workspace over distances and through multiple degrees of freedom. Furthermore, objects near the

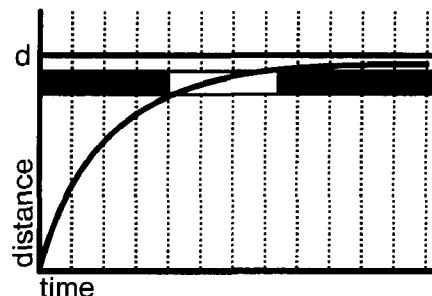


Figure 3: Motion function for logarithmic motion  $f(t) = d - de^{-kt}$ . The time when the motion intersects the desired region indicates the movement is rapid, and the width of the intersection rectangle indicates that the movement can be controlled.

viewpoint fill larger visual angles, which allows the user to make controlled adjustments as the viewpoint approaches an object. Since all motions are relative to the POI, the user need not look at menus, inset orthographic displays, or virtual sliders, which would lead to inefficient interface interactions.

The distance to the POI is used to compute a logarithmic motion function that approaches the POI asymptotically along the ray from the viewpoint to the POI. The function is

$$f(t) = d - de^{-kt} \quad (1)$$

where  $d$  is the distance to the object and  $k$  a proportionality constant for the change. The plot of this logarithmic motion function in Figure 3 demonstrates the desired properties of rapid motion over large distances to the object and controlled motion near the object.

An informal analysis of the user's control task suggests that logarithmic motion should be more effective than high velocity motion for targeted viewpoint movement. The problem with high velocity motion to a target object is that the object stays relatively small for a while and suddenly grows to fill the visual angle. Pew has studied perceptual motor performance and gives evidence that signal predictability can significantly enhance tracking performance [10]. Logarithmic motion has the property that the target object appears to grow at a constant rate of proportionality (the constant  $k$  in the motion function) as the viewpoint approaches, which makes it very easy to predict when the viewpoint will reach the desired distance from the object.

A simple and efficient implementation of logarithmic motion results from using the constant of proportionality  $k$  in equation (1) to calculate the change in the distance from the viewpoint to the POI on each cycle of the animation:

$$\begin{aligned} eye_x &\leftarrow eye_x - k(eye_x - poi_x) \\ eye_y &\leftarrow eye_y - k(eye_y - poi_y) \\ eye_z &\leftarrow eye_z - k(eye_z - poi_z) \end{aligned} \quad (2)$$

where  $eye_x$ ,  $eye_y$ ,  $eye_z$ ,  $poi_x$ ,  $poi_y$ , and  $poi_z$  are the world coordinates of the viewpoint and the POI. This technique avoids using a square root on every animation cycle for calculating the distance from the viewpoint to the POI and integrates nicely with the interactive adjustment of the POI described above. The adjusted position of the POI can be used directly to calculate the new position of the viewpoint

for the next animation frame. The calculation can also be used to move away from the object by making  $k$  negative.

POI adjustment and logarithmic motion represent the functionality of basic POI movement. We have implemented a viewpoint movement interface that includes this functionality as part of an environment with multiple workspaces called 3D Rooms [8,12]. The movement interface uses the middle mouse button (on a 3-button mouse) to adjust the POI and two keys on the keyboard to indicate either forward or backward logarithmic motion. This design separates the functionality of basic POI movement to different hands, the mouse hand for adjusting the POI and the keyboard hand for initiating logarithmic motion. The current interface is easy to use and integrates with our general viewpoint movement interface that uses virtual joysticks to control the velocity of a virtual walk around a 3D room [8].

#### 4.2 Orienting the viewpoint to face a POI

Orienting POI movement extends basic POI movement to include an adjustment of the viewpoint to face the POI for improved viewing and interaction. Orienting POI movement requires two additional operations: (1) lateral movement of the viewpoint toward the object's surface normal at the POI, and (2) a rotation to face the POI. Unlike basic POI movement, orienting POI motion has the property that it moves the screen position of the POI away from the mouse cursor, which is being used to adjust the POI's 3D position. Rather than adjusting the mouse cursor to track the POI screen location, we have found it better to turn the mouse cursor off during POI motion and let the user adjust the POI feedback circle directly as a 3D cursor. Note that the lateral operation should only be used in combination with the rotation operation to prevent movement of the POI out of the window. The entire algorithm for a single animation cycle of orienting POI movement is as follows:

1. Cast a ray from mouse cursor into workspace.
2. Draw POI circle on closest object surface.
3. If the forward (or backward) key is pushed, then use equation (2) to move logarithmically forward (or backward) along ray.
4. If either key is pushed, orient viewpoint:
  - (a) Calculate the POI normal vector.
  - (b) Multiply normal vector by the current distance to the POI to find the lateral position point.
  - (c) Move logarithmically toward the lateral position point using a lateral proportionality constant.
  - (d) Rotate the viewpoint to face toward the POI.

We have implemented an efficient version of orienting POI movement for 3D planar objects using planar equations and the Silicon Graphics graphics library. Our implementation has the additional feature that the user can make the viewpoint hover in front of the POI by simultaneously pushing the forward and backward keys for logarithmic motion. When this is done, the viewpoint will follow the POI as the user adjusts its position, allowing the user to "scroll" across the surface of an object.

We have experimented with various constants for the motion to the POI and the lateral motion to the surface normal. The proportionality constant  $k$  for the logarithmic motion to the POI is currently 0.15. A wide range around this value

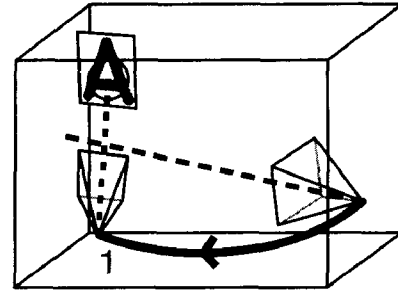


Figure 4: Compared to Figure 1, orienting POI movement has an efficient trajectory towards a point of interest with automatic adjustments of multiple degrees of freedom. The pyramids in the diagram represent two positions of the viewpoint before and after a POI movement toward a rectangle. The position of the viewpoint after movement is determined by logarithmic motion toward the target and lateral motion to the target's surface normal. The orientation of the viewpoint is also rotated to face the POI.

also works. The proportionality constant for motion to the surface normal should simply be larger than for the motion to the POI so that the viewpoint moves to the normal before the desired distance is reached. The current value is 0.25.

POI movement is subject to two additional constraints in our implementation. First, as with our general viewpoint movement using the walking metaphor [8], POI movement is constrained to keep the viewpoint in the 3D room and the virtual body upright. A second constraint is placed on the POI itself. Since the user can move the POI while moving the viewpoint, we constrain the POI to the object selected when POI movement was initiated. This prevents the user from accidentally changing his or her focus of attention from one object to another during viewpoint movement and rapidly moving away from the intended target.

#### 4.3 Discussion of POI movement

POI movement is an effective targeted movement technique. The movement is rapid and controlled, and the interface is simple. Since all movement is relative to the POI, users can place their hands on the mouse and keyboard keys for logarithmic motion and focus their entire attention on the 3D workspace as they move the viewpoint. The interface does not require users to be aware of multiple movement parameters, such as velocity parameters. As Figure 4 shows, the movement trajectories are efficient. Logarithmic motion to the POI can automatically adjust the three degrees of freedom of viewpoint position, and the surface normal operations can adjust the other degrees of freedom of viewpoint orientation. Since users can point at any object they see in the 3D workspace, they are not limited by artificial movement restrictions such as only being able to move along the line of sight. POI movement integrates with current mouse-based interfaces and work environments. Furthermore, logarithmic motion can also be used with multidimensional devices. For example, the user could point at a target object with a VPL glove and use simple hand gestures [13] to fly rapidly toward the target with logarithmic motion.

POI movement is very effective for targeted viewpoint movements, but is less effective for general viewpoint move-

ment, particularly when there is not an appropriate object to anchor the movement. We still use our general viewpoint movement interface (i.e., a walking metaphor) in our 3D Rooms system to explore the workspace.

## 5 POI OBJECT MOVEMENT

Useful 3D workspaces require an efficient and easy to use movement interface that supports both viewpoint and object movement. We have developed a simple general object movement technique, based on a 2D mouse input device, that integrates well with POI viewpoint movement. Furthermore, it can easily be integrated with Bier's snap-dragging techniques when rapid targeted object movements are required [3].

The POI object movement technique uses the mouse cursor to control a ray that determines the lateral position of the object (given the viewpoint coordinates) and uses the same keyboard keys as POI movement to control the position of the object on the ray.

The lateral movement algorithm selects the object's new coordinates by intersecting the ray (projected from the viewpoint through the mouse/cursor position) with the plane perpendicular to the user's line of sight that passes through the center of the object. Objects moved in this way are constrained to remain in the 3D room by clipping the new coordinates to the room coordinates. A desirable side-effect of this clipping is that an object moved into a wall can be dragged along the wall.

The movement of an object *along* the ray requires a more sophisticated motion function than the one used for POI viewpoint movement because in addition to requiring rapid and controlled movements of the object relative to the viewpoint, the interface must support fine-grained movements of the object relative to its own coordinate system. For example, it is impossible to move an object close to a wall with just logarithmic motion when the distance between the object and the viewpoint is large enough that calculations like those in (2) result in large differences. Our solution is to combine two motion functions: one to accelerate the object relative to its coordinate system and the other to clip this acceleration by a logarithmic motion function when the object accelerates towards the viewpoint. The combination of these two functions is shown in Figure 5. The acceleration function allows the user to make fine-grained object movement using a pulsing technique similar to the one plotted in Figure 2. The logarithmic motion function ensures that the user can still make controlled movements as the object moves towards the viewpoint.

The integration of POI object movement with 3D snap-dragging [3] should be straightforward. They both use the mouse cursor to cast a ray into the workspace for finding points of interest. The snap-dragging skitter can be used for POI feedback, with the added bonus that gravity can assist with the precise specification of the point of interest. Finally, the style of the two techniques is similar, for they both allow the user to focus on the 3D workspace during movements.

The principal advantage of 3D snap-dragging is that precise placement of objects is easy. However, precise placement, with its additional complexity of specifying alignment objects, is not always desirable or necessary. The advantage of POI object movement is that it provides an intuitive and natural way of doing approximate placement of objects. The combination of POI object movement and 3D snap-dragging will allow both general movement and precise

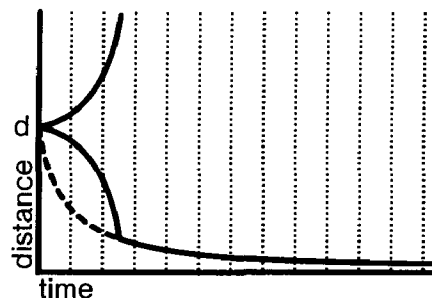


Figure 5: The motion functions for POI object movement are plotted as previous figures with the viewpoint at distance 0 and the object at distance  $d$ . The acceleration function is plotted as parabolic curves toward and away from the viewpoint. The logarithmic motion is plotted toward the viewpoint and clips off the velocity of the acceleration.

movement when needed.

The POI object movement technique has a number of advantages. It uses the same user interface conventions as POI viewpoint movement, hence is fully integrated with it. It only requires a 2D input device like a mouse, but can be used with a multidimensional device. It is very easy to learn and use. Finally, it can be used in combination with 3D snap-dragging when precision is required.

## 6 CONCLUSION

We have described a new technique, point of interest logarithmic motion, that offers an improvement in the techniques available for targeted 3D viewpoint movement, which occurs when users wish rapid access to many objects or objects with great detail. In this technique, the user selects a point of interest. The technique uses this information to simplify the user's control task, resulting in movement that is both rapid *and* controlled. On each animation cycle, the viewpoint is moved the same relative percentage of the distance toward the point of interest target. Thus the movement is rapid when the user is distant, but slow and controlled when very near the target. Like the arrow of Zeno's Paradox, the user's viewpoint continues to fly closer to the point of interest, but never actually reaches it. Instead, the user seems to see the target open at a uniform rate, revealing ever finer detail.

In addition to satisfying the goals of rapid and controlled viewpoint movement, this technique also satisfies the more general interface requirements listed in Section 2:

1. The technique is easy to use. The user only has to point into the workspace and indicate forward or backward movement. The complex parameters of viewpoint movement, such as position, orientation, and their rates of change are determined by the point of interest and are adjusted automatically.
2. The technique can help avoid disorientation. Logarithmic motion is predictable and easy to control. This technique also helps with orientation by making it fast for the user to zoom out to get orienting views and then to zoom back in.



3. The technique can be integrated with other techniques. It can be used with 2D or multidimensional input devices. It is fully integrated with the POI object movement technique for general object positioning, and can be combined with 3D snap-dragging to get precise object positioning.
4. The technique enhances the perception of the virtual workspace. POI movement allows the user to easily move through the workspace without having to shift attention to menus or other user interface artifacts.

POI movement should enhance a person's ability to interact with information spaces containing many items. It is thus a potential component of future systems that will support complex and interesting interactions between human and machine.

**Acknowledgments** We would like to thank Polle Zellweger and Eric Bier for their suggestions and comments on this paper. Lennart Lövstrand helped us implement an earlier version of this algorithm.

#### References

- [1] Badler, Norman I., Kamran H. Manoochehri, and David Baraff. Multi-Dimensional input techniques and articulated figure positioning by multiple constraints. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics* (Chapel Hill, NC, October 1986). ACM, New York, 1987, 151-169.
- [2] Bier, Eric A. Skitters and jacks: interactive 3D positioning tools. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics* (Chapel Hill, NC, October 1986). ACM, New York, 1987, 183-196.
- [3] Bier, Eric A. Snap-dragging in three dimensions. Proceedings of the 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah, March, 1990). In *Computer Graphics 24*, 2 (March 1990), 193-204.
- [4] Brooks, Frederick P. Jr. A dynamic graphics system for simulating virtual buildings. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics* (Chapel Hill, NC, October 1986). ACM, New York, 1987, 9-21.
- [5] Chen, Michael, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. Proceedings of SIGGRAPH'88 (Atlanta, Georgia, August 1988). In *Computer Graphics 22*, 4 (August 1988), 121-129.
- [6] Evans, Kenneth B., Peter P. Tanner, and Marcell Wein. Tablet-based valuators that provide one, two, or three degrees of freedom. Proceedings of SIGGRAPH'81 (Dallas, Texas, August 1981). In *Computer Graphics 15*, 3 (August 1981), 91-97.
- [7] Fairchild, Kim M., Steven E. Poltrock, and George W. Furnas. Semnet: three-dimensional graphic representations of large knowledge bases. In *Cognitive science and its applications for human-computer interaction*, Guindon, R. (ed), Lawrence Erlbaum, 1988.
- [8] Mackinlay, Jock D., Stuart K. Card, and George G. Robertson. A semantic analysis of the design space of input devices. *Human-Computer Interaction*, to appear in vol. 5, 1990.
- [9] Nielson, Gregory M., and Dan R. Olsen Jr. Direct manipulation techniques for 3D objects using 2D locator devices. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, (Chapel Hill, NC, October 1986). ACM, New York, 175-182.
- [10] Pew, R. W. Human perceptual-motor performance. In *Human Information Processing: Tutorials in Performance and Cognition*, Kantowitz, B. H. (ed), Lawrence Erlbaum, 1974, 1-40.
- [11] Phillips, Cary B., and Norman I. Badler. Jack: a toolkit for manipulating articulated figures. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, (Banff, Canada, October 1988). ACM, New York, 221-229.
- [12] Robertson, George G., Stuart K. Card, and Jock D. Mackinlay. The cognitive coprocessor architecture for interactive user interfaces. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, (Williamsburg, Virginia, November 1989). ACM, New York, 10-18.
- [13] Sturman, David J., David Zeltzer, and Steve Pieper. Hands-on interaction with virtual environments. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, (Williamsburg, Virginia, November 1989). ACM, New York, 19-24.
- [14] Ware, Colin and Steve Osborne. Exploration and virtual camera control in virtual three dimensional environments. Proceedings of the 1990 Symposium on Interactive 3D Graphics (Snowbird, Utah, March 1990). In *Computer Graphics 24*, 2 (March 1990), 175-183.
- [15] Weimer, David, and S. K. Ganapathy. A synthetic visual environment with hand gesturing and voice input. In *Proceedings of CHI'89*, (Austin, Texas, April 1989). ACM, New York, 235-240.