Volume 016, (1997) number 005 pp. 269–281

A Survey of 3D Interaction Techniques

Chris Hand

Department of Computer Science
De Montfort University
The Gateway
Leicester LE1 9BH, UK
cph@dmu.ac.uk

Abstract

Recent gains in the performance of 3D graphics hardware and rendering systems have not been matched by a corresponding improvement in our knowledge of how to interact with the virtual environments we create; therefore there is a need to examine these further if we are to improve the overall quality of our interactive 3D systems. This paper examines some of the interaction techniques which have been developed for object manipulation, navigation and application control in 3D virtual environments. The use of both mouse-based techniques and 3D input devices is considered, along with the role of feedback and some aspects of tools and widgets.

Keywords: 3D interaction; object manipulation; navigation; feedback; virtual environments; widgets

1. Introduction

Although many interactive computer graphics systems are now able to render high-quality shaded 3D models in real time, there remains a problem of how to interact with virtual environments in a natural and error-free manner. This paper presents a survey of the many techniques which have been used to perform 3D tasks such as object manipulation and navigation.

Virtual environments may be presented to users via many different configurations of computer system. Even the simplest desktop set-up, with a standard monitor and mouse, is capable of presenting interactive 3D graphics to some extent. In domains such as CAD or visualisation we commonly find the desktop system being extended through the use of 3D joysticks or sometimes stereoscopic displays, using shutter glasses for example. More traditional virtual reality systems may use six degrees-of-freedom (6-DOF) tracking devices to measure the position and orientation of a pointing device and a head-mounted display (HMD), which allows the user's viewpoint to change interactively as the head is moved. An alternative to the "encumbering"[1] technology of the HMD is to use one or more projection displays to create a CAVE[2], possibly with tracking being performed using video cameras.

The 2D techniques of the "desktop metaphor", such as pull-down menus and dialogue boxes, are inappropriate for a large class of applications, particularly where a HMD is worn (since the keyboard and mouse cannot be seen) or where a glove or 6D pointing device is being used. Although the system configuration used (especially the number of degrees of freedom of the input devices) does have an impact on the interaction techniques which are feasible, the main aim of this paper is to provide an overview of the techniques we can implement in software to make use of whatever input devices are available, rather discussing the best configurations for given tasks.

The design of the human-computer interface should be informed not only by a knowledge of the capabilities of the human sensorimotor system[3], but also by the way in which we conceptualise 3D tasks. By the time we reach adulthood we have perfected many manipulation and navigation tasks to the point where we can perform them without conscious attention. It

is this level of naturalness and transparency which virtual environments seek to attain — the interface almost becomes invisible when we can manipulate the virtual objects as if they were really there. Thus the user can focus more on the task, becoming completely engaged with the virtual world and feeling as if they are interacting with the objects directly, with no intermediary[4]. This paper discusses interaction techniques with regard to their ability to provide natural or direct interaction, as well as considering the role of feedback in general for each kind of interaction task.

The remainder of this paper presents an overview of the field of 3D interaction. By examining the state of the art in interaction techniques in addition to their history, some conclusions can be drawn concerning trends, problems and future possibilities. In order to allow the other elements of interaction (such as feedback) to be covered in detail, input devices themselves are not discussed in any great depth here — see instead other surveys[5, 6] and especially Buxton's discussion on designing for device idiosyncracies[7].

The paper is structured as follows. Section 2 presents the background and introduces some terminology. Sections 3, 4 and 5 describe interaction techniques used to perform object manipulation, viewpoint manipulation and application control respectively. Section 6 compares the features of tools and widgets, section 7 discusses the work reported earlier in the paper and conclusions are drawn in section 8.

2. Background

Although the development of virtual environments can be traced back to the 1960's [8], it is mainly in the past 12 years or so that we have seen a large number of 3D systems being developed. Early research concentrated very much on the technology which had recently made this kind of work possible, and so little is reported on the interaction techniques used. Around the middle of the 1980's it became increasingly common to experiment with interactive 3D systems, resulting in a large number of reports on using new techniques or input devices

Much of the recent work on 3D interaction is applied to simplified task domains, rather than fully-fledged systems. Examples of typical tasks include 3D Modelling [9, 10, 11, 12, 13], Scene composition[14, 15, 11], Simple furniture layout[16, 17], Orientation matching [18, 19], Exploration/Movie-making[20, 21] and Visual Programming/Widget Construction[22].

Most of these domains have at least three tasks in common: *object manipulation*, *viewpoint manipulation* and *application control*. (The term "viewpoint manipulation" is used here rather than "viewpoint movement"[23] to avoid excluding the control of parameters such as Field of View and Zoom Factor.) In general all three of these sub-tasks will be performed as part of a larger task. The relationship between object manipulation and viewpoint manipulation is an interesting one, since some systems treat the latter as the manipulation of an object which represents the camera or a "virtual eye". Similarly, application control in some systems is performed by manipulating objects (*tools* or *widgets*). However, in this paper the three tasks are treated separately.

(As an aside, the separation of an interface into these three components is already widely performed by those who use the Model-View-Controller paradigm[24] for implementing user interfaces in object-oriented systems.)

The next three sections describe some of the techniques used to implement *object manipulation*, *viewpoint manipulation* and *application control* in 3D systems.

3. Object Manipulation

A typical task performed in a 3D virtual environment will include the direct manipulation[25, 26] of graphical objects in that environment: selecting, scaling, rotating, translating, creating, deleting, editing and so on. Some of these techniques correspond directly to actions we perform in the real world (*e.g.* translation, rotation), while others would be impossible in real life but are required for the sake of working on the computer-based application (*e.g.* scaling, deletion).

In the literature we can identify two distinct phases in the development of 3D interaction techniques: the evolution of techniques based on the use of the 2D mouse, and the new ideas generated by the introduction of true 3D input devices.

3.1. Evolving Mouse Techniques

Of the three sub-tasks enumerated above, object manipulation is much more widely reported in the literature. A sizeable repertoire of interaction techniques has evolved, initially from ideas associated with the 2D mouse such as dragging a movable cursor.

Eric Bier's "Skitters and Jacks"[27] used two types of cursor (not manipulated simultaneously) in a scene composition task. The Skitter is a cursor which looks like a wire-frame representation of the positive unit vectors in x, y and z, typically with the x and y axes corresponding to the surface of an object and the z axis corresponding to the surface normal. The direction of the three vectors define an orientation, while the "origin" of the axes defines a point in Euclidean

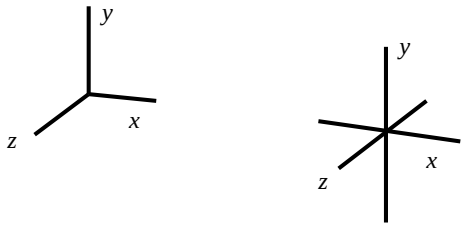


Figure 1: *Skitter (left) and Jack (right)*

space (see Figure 1). By moving the mouse the Skitter may be moved over the surface of the object. The Jack is similar, but is represented by vectors extending in both positive and negative directions. The Jack is used to mark points during a manipulation, and to align with other jacks.

Similar techniques are described by Nielson and Olsen [28], with the skitter known as a "triad cursor" and the Jack known as a "full space cursor" — the term *Jack* has come to be used to describe a three-dimensional cross-hair cursor, defining a point at the intersection of three infinitely long orthogonal lines.

These techniques are typical of the approach of the late 1980's, in that the user must conceptualise the tasks in terms of co-ordinate geometry, vectors and rotation angles. Although this may be appropriate for engineers, in general this goes against one of the goals of virtual environments, namely to allow users to apply natural skills which they already use in the real world.

Skitters and Jacks evolved into "Snap-dragging in 3D" [14], and along with the earlier work this influenced the design of techniques such as "Tail-dragging"[15]. The important work of the graphics group at Brown University also directly acknowledges its heritage ("linking is related to snapping" [22]).

One of the early limiting factors in creating 3D interfaces was the relative lack of suitable 3D input devices, and so techniques inevitably turned to using the 2D mouse along with various modes and mappings to make up for the lack of degrees of freedom. An influential early paper by Chen *et al* [18] described techniques for working in 3D using 2D input devices, in particular the "virtual track-ball", which although not particularly easy to use for many tasks has since been implemented in many mouse-based applications.

The lack of suitable input devices led to further experiments to design interaction techniques which compensate for this in software. Houde's experiments with hand-style cursors and bounding boxes with handles[16] allowed subjects to move furniture around a room in an arrangement task using a mouse. The task itself had a reduced number of degrees of freedom since objects were constrained in their movements — reasonable enough for this particular problem domain.

Another example of creating extra abstract objects to assist interaction in the virtual world is the "laser pointer", used in the DIVE VR system[29], *Zashiki-Warashi* [17] and JDCAD[30] among others. The beam is a ray, cast from a given point (*e.g.* the user's pointer) in a straight line. The first object to be intersected by the ray is selected for manipulation. This has an advantage in that it allows "action at a distance" (AAAD): the user does not need to be close to an object in order to manipulate it. An extension of this technique, known as cone selection[30] or spotlight selection[31], uses a conical selection volume rather than a simple line to overcome problems associated with selecting small objects.

A further set of AAAD techniques, known collectively as image plane interaction, is described by Pierce *et al* [32]. These make use of the fact that, when using a head-mounted display, we can resolve distant and close objects simultaneously (unless some simulated focus or depth of field mechanism is used), and so gestures made with a virtual hand in front of the face can be seen to "frame" distant objects in the virtual environment (similar to the spoof photographs showing a distant person "standing" on the palm of someone in the foreground, or showing a tourist "holding up" the leaning tower of Pisa).

These latter techniques really require a 3D pointer or glove device in order to be fully effective. The next section describes how such devices have influenced the evolution of interaction techniques.

3.2. The Influence of 3D Input Devices

Once 3D input devices became available many research projects began to look into what was possible with the new technology. Early work at the Massachusetts Institute of Technology (MIT) such as the gesture and speech-based *put-that-there*[33] and Schmandt's stereoscopic workstation[34] used a polhemus electromagnetic 6-DOF tracker. Due to its small size, the ease of using it to instrument familiar objects, [35, 36, 5] and the ability to measure both position and orientation, the polhemus device became very popular:

The VPL Dataglove used a polhemus device for tracking the position and orientation of the hand[37].

3-Draw was another MIT project[36] using two polhemus devices to track a clipboard and stylus, thereby enabling the creation of curves in 3D, either by free-form drawing or using constraints.

Sculpting was the name given to a technique developed by Galyean and Hughes which allowed direct manipulation of a volumetric representation by chiselling away parts of the volume with a polhemus-based tool[10].

The appearance of instrumented gloves was significant as this enabled what were arguably the first *true* direct manipulation interfaces. Using a virtual hand controlled by a glove instead of a cursor has the potential to be very direct, expressive and natural, except that many applications had no provision for tactile or force feedback and so users were unable to feel the virtual objects as being there.

Another possibility opened up by glove devices was gestural input, and much of the research into the use of gloves has concentrated on ways of recognising gestures[38, 39, 40] rather than the techniques for interacting with the virtual world *per se*. Glove-based interaction is still an under-explored area in its own right, but is considered to be outside the bounds of the work reported here. An useful overview of glove-based input may be found in Sturman and Zeltzer's survey[6], while David Sturman's PhD thesis[41] presents a much more detailed discussion.

Other 6-DOF tracking devices have continued to appear, among them the Spaceball, a multi-axis joystick controller using a force-sensing ball. Although widely used, opinions vary on how best to incorporate the Spaceball into a 3D system. Le Blanc *et al* claimed their Spaceball and mouse combination presented the user with a "sculpting metaphor"[9], and although the system demonstrates the usefulness of two-handed interaction, the mouse-based interactions are much less direct than, say, Galyean and Hughes' technique[10].

The simultaneous use of two hands for input has long been recognised as beneficial[42], so it is perhaps surprising that there have not been more two-handed interfaces to 3D systems until recently. Shaw and Green's THRED[13] (Two Handed Refining EDitor) allowed two-handed polhemus-based surface modelling, while Sachs *et al* made good use of both hands in 3-Draw[36], noting that the use of an instrumented clipboard and stylus allowed *kinaesthetic feedback* since the user was manipulating physical objects. Feedback is considered in more detail in the next section.

3.3. Natural Feedback

Most systems will provide visual and possibly auditory feedback during a task, but exteroception and proprioception, particularly via the kinaesthetic and tactile senses, can also be extremely important factors determining a user's success in virtual manipulation[43]. (Proprioception refers to the perception of the position and orientation of one's own body, while exteroception refers to the perception of external phenomena using the senses of hearing, vision, touch and so on.)

During our interactions with the natural world, *kinaesthetic feedback* allows us to know the position of our limbs and manipulators relative to the rest of the body, while the touch sensors in the manipulators and throughout the skin allow *tactile feedback* on the texture, shape and temperature of a surface. When using an input device to control an object in a virtual environment it is often helpful if there is a natural kinaesthetic correspondence[19] between the movement of the hand (or other manipulator) and the manipulated object.

Much of the recent work in 3D input devices has aimed to exploit these types of feedback. For example, the use of an instrumented deformable shape to manipulate a virtual control volume [12] has enabled both tactile and kinaesthetic feedback. Some systems (*e.g.* the SANDPAPER force display system [44]) go even further than exploiting the inherent kinaesthetic feedback provided by physical input devices, by using actuators to provide *force feedback* to the user under direct control of software.

Providing feedback by manipulating physical input devices which closely correspond to the virtual objects is an important step towards bridging the gap between knowing what we want to do and knowing how to do it, or the gap between perceiving the state of the system and knowing what it means (also known as the *Gulfs of execution and evaluation*[26]). Some generality is lost by providing specialised input devices, but this may be acceptable to improve the quality of the interaction for the user. (Traditionally programmers have been in favour of generic input devices, groups of which may all be mapped to the same interaction technique. Buxton[7] and others have argued that this is over-simplistic and denies us the opportunity to design according to each device's strong points or idiosyncrasies. Hence the trend has been away from generality.) Devices may also be augmented to produce feedback which is more appropriate to the task — for example Galyean and Hughes[10] suspended a Polhemus device inside a crate using elastic, a set-up they dubbed "the poor man's force feedback unit".

Another recent trend, building on the approach exemplified by 3-Draw, is to move the interaction even closer to the task domain by using instrumented "props" specific to the task. For example, Hinckley *et al* describe the use of instrumented tools such as a cutting plane and a model head for controlling the visualisation of brain scan data[35].

Thus we can see a trend which has gradually moved the emphasis away from performing 3D tasks in the

computer's domain (by specifying rotation angles and vectors in the machine's Euclidean representation) to working directly in the user's domain, by instrumenting the tools with which the user can work in a natural manner thanks to feedback from the real world as well as from the virtual environment. This can be seen as a positive step towards redressing the imbalance (noted by Buxton[45]) between our natural skills and the opportunity for using them to interact with virtual environments.

4. Viewpoint Manipulation

Viewpoint manipulation encompasses the tasks of navigating in the virtual environment by movement of the point-of-view (*i.e.* the camera or "virtual eye"), as well as controlling the viewing parameters such as Zoom Factor and Field of View.

Unfortunately it seems that the design of a viewpoint manipulation technique is often seen as relatively unimportant, and typically little thought is allocated to implementing anything other than a "flying metaphor" (as Chuck Blanchard, late of VPL, has commented: "nobody walks in VR — they all fly"[46]). Unconstrained 3D flying is an easy technique to program but is not a natural way of moving around (except perhaps for trained pilots) — if naturalness is one of our aims then alternative techniques must be found. For example, Fellner and Jucknath[47] used a ray-casting technique to intelligently select an appropriate navigation method (walk, fly, fall or step) based on the distance to the closest obstacle.

Mackinlay, Card and Robertson's techniques for 3D viewpoint movement (or "egocentric motion")[23] were developed as part of the Xerox 3D ROOMS system[48]. A simulated head-up display uses two movement icons superimposed on the 3D display of a graphical room. By clicking on a four-way arrow icon the viewpoint may be translated in the plane of the virtual body, or by clicking on an eye icon the gaze may be rotated. The use of the arrow icon is interesting, since it operates as a "virtual joystick": clicking the mouse on the icon and dragging causes a vector to be drawn, the length of which determines the speed of movement. The rubber-banding of the line also provides visual feedback. This technique has been adopted as a mouse-based navigation technique in many 3D environments, such as the DIVE VR system[29] and VRML browsers such as Community Place and VRweb (Figure 2).

Mackinlay *et al* describe this technique as a "walking metaphor". Elsewhere[23] they include this when they distinguish four types of viewpoint movement: *general movement* – exploratory, including walking;

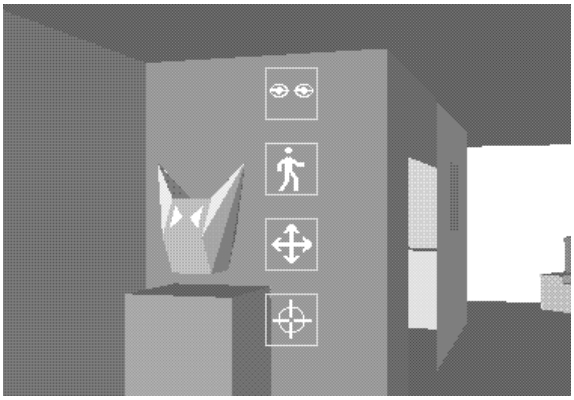


Figure 2: *Head-Up Navigation Icons in the VRweb VRML Browser*

targeted movement – with respect to a specific target; *specified coordinate movement* in which the exact position and orientation are supplied; and *specified trajectory movement* in which the path of the viewpoint is specified (as with a camera fly-through). The targeted movement technique[23] moves the user towards a point of interest with a speed logarithmically related to the distance from it, which has the effect of moving at greater speed when the distance is greater but slowing down dramatically as the point of interest is approached.

Brooks[49] describes the navigation techniques used in the WALKTHROUGH architectural visualisation system as the *helicopter metaphor* (using joysticks), the *eyeball metaphor* (using a 6-DOF tracker to manipulate the viewpoint directly), and the *shopping cart metaphor*, in which the user walks on a treadmill and steers using handlebars similar to pushing a trolley around a supermarket.

Ware and Osborne implemented and described three interaction metaphors they called *Scene in Hand*, *Eyeball in Hand* and *Flying Vehicle Control* [20]. In the Eyeball in Hand technique a 3D input device has its position and orientation data mapped directly onto the viewpoint, later modified to include a clutch mechanism to allow larger movements to be made. The Scene in Hand technique maps the movements of the 3D input device onto the *virtual world*, so that exploratory viewing is performed by keeping the viewpoint still and moving the world around. Again, a clutch mechanism was included to allow a greater range of movement. Flying Vehicle Control uses a less direct mapping, with the velocity being controlled by the cube of the displacement of the input device.

Again, these three techniques have been adopted for use in other systems: the Scene in Hand technique was

also used by 3Draw [36] and in the "Ball and Mouse" sculpting system of LeBlanc *et al* [9] among others, while DIVE [29] implemented several different "vehicles" including a mouse vehicle (which is also based on the heads-up icons of Mackinlay *et al* [23]) and a Head-Mounted Display vehicle which follows the movement of a head tracker — effectively an "eyeball on head" technique which is nothing if not based on natural principles.

It might be useful if the viewpoint could move automatically according to the task being undertaken or the area of interest in a virtual world. Phillips *et al* [50] describe a system for automatically adjusting camera angles to avoid degenerate rotation or translation conditions (due to using a 2D input device mapped onto 2 of the 3 dimensions at any one time), although this is intended as a way of avoiding errors rather than doing away with the need to manipulate the viewpoint altogether. Gleicher and Witkin's "through-the-lens camera control" [51] is a collection of techniques based on computing higher-order camera parameters (such as velocity) interactively according to user input. A system using eye-tracking might be able to go further towards automatic viewpoint manipulation — for example noticing that a fixated object is moving out of the field of view and automatically panning to follow it.

It is also possible to move the viewpoint by attaching it to an object and controlling it using object manipulation techniques as described in section 3. A technique based on a physical model using Newtonian mechanics described by Turner *et al* [52] allows the user to control the viewpoint using a virtual camera object, as do the *Zashiki-Warashi* system[17] and the popular ALIAS 3D modelling package[53]. Complementary to this is the World-In-Miniature (WIM) technique[54] which enables a small virtual model of the scene (Figure 3a) to be held in the hand (scene-in-hand) and manipulated to allow rapid changes to the viewpoint, as well as object selection. This is another props-based technique, with a similar input device arrangement to 3-Draw (see Figure 3b).

The HMD Vehicle used in DIVE is an example of an important technique: coupling head movement with that of the viewpoint. One possibility this suggests is to use the head position to accurately compute the stereoscopic projection[55] rather than simply assuming a given viewing position. This would overcome the problem with non-immersive stereo displays of the image "following the user around" as the head is moved.

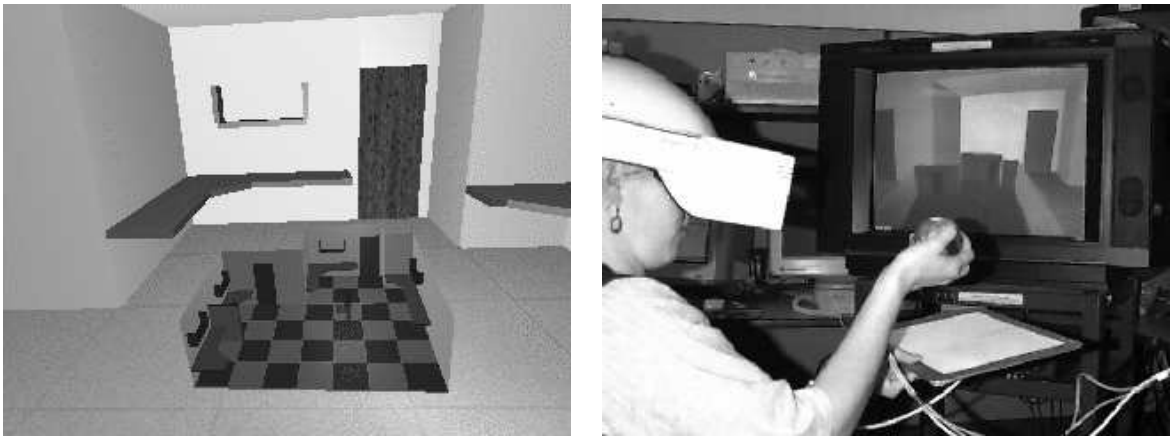
Apart from being a particularly good example of employing a natural mapping from input device to task, coupling the viewpoint to the user's head position also allows a certain amount of visual exploration of the virtual environment *without using the hands*, which may then be used to control other input devices. All VR systems using tracked head-mounted displays exhibit this feature, along with some non-immersive systems such as Ware's "fish tank VR" system[56] which used liquid crystal shutter glasses along with a mechanical head-tracker to overcome tracking delay problems. This is another example of kinaesthetic correspondence.

4.1. Feedback during Viewpoint Manipulation

Vision, locomotion and manipulation are three major faculties used by humans to perform tasks in their real environment. James Gibson, founder of Ecological Psychology, acknowledges that the three are very closely interrelated: visual perception depends on locomotion ("a moving point of observation is necessary for any adequate acquaintance with the environment. So we must perceive in order to move, but we must also move in order to perceive"[57]). Howard[58] considers vision, locomotion, kinaesthesis and audition to be the major controlling factors of human orientation behaviour, including judging direction, egocentric orientation and relative orientation of body parts. Hence the feedback provided during viewpoint manipulation is not just important for knowing where we are and how fast we are moving, but for understanding the scene as a whole.

The feedback created by the *optic flow* [57] as we move through a virtual environment is important to viewpoint manipulation in the same way that natural kinaesthetic and tactile feedback are important to object manipulation. Further, the optic flow is actually interpreted as locomotion, to such an extent that it can induce nausea (so-called *simulator sickness*) if large delays are present between head movement and scene update[59], or if there is no corresponding vestibular feedback[60].

Vestibular feedback provides us with information concerning our orientation and acceleration relative to our surroundings. Apart from expensive flight simulators and entertainment systems which use software-controlled motion platforms, very few virtual environments are able to provide active vestibular feedback. However, we have already seen that passive feedback may also be provided if it can be made inherent in the system. Slater *et al* made use of this approach in their "Virtual Treadmill"[61], which tracks the movements of the head to detect when the user is "walking". This action is then translated into locomotion within the virtual world, ensuring a certain amount of kinaesthetic correspondence.



(a) Miniature World

(b) WIM User with Instrumented Clipboard and Tennis Ball

Figure 3: *The World-In-Miniature (WIM) Technique (adapted from Stoakley et al [54])*

4.2. Frames of Reference

Moving the viewpoint through a virtual space creates in the user the sense of being at the centre of that space, which is our normal everyday experience during locomotion. This is known as an *egocentric* frame of reference[58]. Conversely, an *exocentric* approach gives a feeling of looking in from the outside, with the centre of attention being the manipulated objects. The Eyeball in Hand and Flying Vehicle Control techniques described above may be classified as egocentric techniques, while the Scene in Hand technique is exocentric.

It may be the case that we can only operate in either egocentric or exocentric mode at any one time, and must switch modes when we want to work in the other frame of reference. If this is so, then it may be possible to exploit this by using the same 3D input device for manipulation/navigation at different times, as long as the device is appropriate for both.

5. Application Control

The term Application Control describes communication between user and system which is not part of the virtual environment. Changing operating parameters, dealing with error conditions, accessing on-line help and changing mode are all examples of this kind of task.

This aspect of the 3D interface is the least reported of the three under discussion here. Perhaps one reason for this is that it is often possible to produce a usable system by carrying over the application control techniques used in 2D interfaces, such as buttons and menus, and to implement them on top of (or despite) the 3D environment. One danger here is that by "converting" the task from 2D to 3D it will become much more difficult to perform. For example, it is not uncommon for a system to implement a 3D menu floating in space, so that to choose from the menu the user must to make the 3D cursor intersect the appropriate menu choice. Not only does this change a one-dimensional task (choosing from a list) into a three-dimensional one, but it also increases the possibility of making errors — if the cursor is not at the correct depth then the menu isn't even activated. Worse still, if a stereoscopic display is not being used then it is almost impossible to judge the depth correctly. One solution to this kind of problem, used in the CHIMP system[31], is to use a "laser pointer" (as described in section 3.1) to select the menu choices, providing useful feedback at the same time as simplifying the task by reducing it to a matter of intersecting an object with a line, rather than intersecting two objects.

This approach of "converting" to 3D has other difficulties — if a stereoscopic display is being used it can be difficult to integrate 2D user interface components into the stereo image being generated. For example, should flat menus appear in the region of zero parallax (at the same depth as the screen)? If so, they will be occluded by any objects which are rendered in the nearer depth planes. But if the 2D objects are placed in 3D space, then there may still be problems with disparity, occlusion, and the manipulation of the cursor in three dimensions. In any case, software (and possibly the user interface architecture) has to be modified to handle the 3D aspects of the interface.

This is also important from the point of view of

maintaining a user interface metaphor, or supporting the feeling of immersion or directness. In fact this is a general problem with application control, since it may require the user to change from talking directly *to* the interface objects, to talking *about* them (the *use-mention distinction*), thereby stepping outside the frame of reference used when manipulating objects or the viewpoint. If this shift is too great (or, to use Laurel's terminology[4], we step outside the *mimetic context*) then the engagement of the user may be broken.

A further problem arises with systems using sensory immersion. If the user is wearing a HMD this may prevent the use of traditional 2D tools such as the keyboard or mouse since they can no longer be located visually. Furthermore, the use of an instrumented glove for gestural and spatial input means that the hand is too encumbered to use other tools.

In these situations it is essential to develop new interaction techniques which may be used to control the application *within* the 3D environment, or within the mimetic context of the interface (*e.g.* a particular metaphor). Most of the literature which deals with this kind of application control concentrates on either gestural input or on the use of 3D *tools* and *widgets*.

6. Tools and Widgets

As reported in the literature covered here, *Tools* are generally based on a metaphor while *Widgets* are an abstraction (although "widget" is often used to refer to a general user interace component, a distinction is made here for the sake of this discussion). For example, the Unified Graphics Architecture system's Colour Selector Widget [62] is shown as three orthogonal solid arrows representing Red, Green and Blue, each of which has impaled on it a sliding sphere that can be moved along the direction of the arrow to express the Red, Green or Blue intensity value. This is a Widget since it is an abstract device, not intended to represent any real-world object (although sliders are based to some extent on sliding potentiometers), and hence more easily modified or adapted to a particular task. An example of a *tool* is Balaguer and Gobbetti's "Dr Light"[11], used to specify lighting in a scene, which looks and behaves much like a coloured spotlight as used in the theatre.

The "virtual tricorder"[63] of the Brown Graphics Group takes the tool idea a step further by using a representation within the virtual environment which matches exactly with the real input device being used (in this case a Logitech 3D mouse). This allows a one-to-one mapping between input device and virtual controller to be in operation at all times (Wloka refers to the enhanced kinaesthetic correspondence simply as "tactile feedback"). The virtual tricorder, actually a re-programmable multi-purpose tool, uses "2D anchored menus" [63] as a way of overcoming the problem of interacting with menus in 3D. When pop-up menus appear, they are attached to the virtual tricorder which allows them to be brought closer for easier viewing. The selection mechanism uses the up/down/enter buttons on the 3D mouse, which recognises menu selection for what it is — a 1-DOF task (contrast this with the problem of selecting a menu option by intersecting with it in 3D as described in section 5). As Wloka points out [64], using a Tool to interact with a user interface object is more direct than interacting with the traditional elements of the desktop.

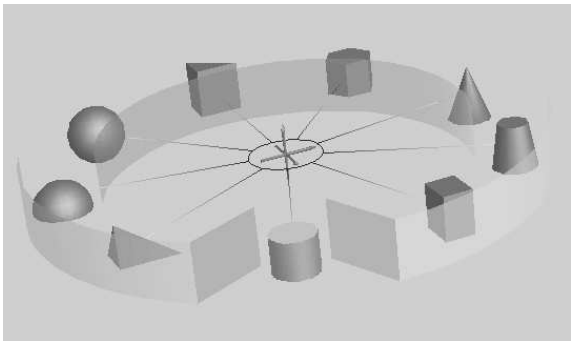


Figure 4: *JDCAD's Ring Menu (courtesy of Jiandong Liang)*

It is quite common for interactive 3D systems to provide a number of virtual tools, which then creates another problem in that the user must be able to locate and select an appropriate tool when it is required. Figure 4 shows the "ring menu" used in the JDCAD system[30]. This presents the available tools in a circular configuration with one "slot", which always points towards the user, containing the selected tool. Selection of a new option requires simply rotating the ring, which is again a 1-DOF task.

Using well-designed Tools and Widgets allows the user to control the operational parameters of the application in a way which is less damaging to the feeling of directness than using more abstract or intrusive user interface techniques (although, as with other interaction techniques, this will also depend on how the input devices are mapped onto the task). A good example of a technique which might cause this is "clutching", whereby the mapping of input device to the virtual controller is temporarily suspended, perhaps by pressing a button. Ware and Osborne's navigation metaphors (section 4) were found to require a clutch mechanism after using them for a short time. This is typically done because an absolute positioning device

is being used to control the value of a parameter which is outside normal operating range. Picking up a mouse when it reaches the edge of the mat (and replacing it in the middle) is another example.

Performing clutching operations may make the interaction seem unnatural, reducing the feeling of engagement, so techniques for avoiding it are worth investigating. Wloka[63] describes how, when the input device is mapped one-to-one onto a graphical object in the virtual environment, the clutching becomes a natural action for the user in two ways. Firstly, when the limit of rotation of the hand is reached, the other hand may hold the object while the first hand moves back to starting position. Secondly the tool controlled by the input device may be applied to the object or may be disengaged, as necessary (as when tightening a screw with a screwdriver).

However, it may be that constraints external to the interaction actually contrive to reduce the directness. Examples of this include bumping into an "invisible" real-world object which is unseen due to the user wearing a head-mounted display, or being unable to move a 3D tracking device outside a certain range due to the length of its cable. Laurel suggests that these "extrinsic constraints, when they cannot be handled invisibly, should be expressed in terms of the mimetic context" [4]. This suggests that tools which mimic real-world objects may be superior to abstract widgets where external constraints are a problem.

7. Discussion

There is evidently a large body of literature describing many techniques for interacting with 3D graphics, the majority being for mouse-based object manipulation or navigation. Most of the work has been implemented as part of proprietary or research systems, hindering widespread re-use, evaluation and evolution of the techniques. Also, unlike 2D graphical user interfaces, there is no "standard look and feel" for 3D user interfaces. (Although standards may be considered by some to stifle creativity in the design of new interfaces, they do allow systems to fulfil the expectations of users, especially novices. There is still plenty of room for both viewpoints in 3D user interfaces.)

This section discusses the future of 3D interaction techniques with respect to four main areas: Human-Computer Interaction (HCI), Specification and Dissemination, System Configurations and the importance of VRML.

7.1. Human-Computer Interaction

Often the technical difficulties surrounding the implementation of interaction techniques have meant that evaluating their usability has received less attention than it should. Ideally all the techniques implemented would be subject to rigorous human factors evaluations using standard techniques. However, the HCI community has little in the way of standard evaluation methods for 3D interaction short of adopting techniques such as Fitt's Law, which has already caused much discussion in its adaptation from 1D to 2D tasks[65].

A better appreciation of the issues surrounding feedback will also guide the development of interaction techniques. Gibson suggests[66] that haptic perception (which is a combination of tactile and kinaesthetic feedback) may play a more fundamental role in the control of manipulatory activity than vision, which might help to explain why viewpoint manipulation techniques (using almost exclusively visual feedback) seem to be simpler to use and more successful than those for object manipulation. Without advances in haptic feedback, or a greater understanding of the application of cross-modal feedback (such as substituting sound for haptic cues), this situation may not improve.

7.2. Specification and Dissemination

It is important that we begin a process of consolidation, during which the existing 3D interaction techniques are collected, implemented and distributed widely. The adoption of open standards such as VRML97 (see section 7.4) would allow implementations to be distributed and shared throughout a large user base, and would mean that a greater gamut of techniques would be available for evaluation by human factors specialists.

Another approach might be to develop a standard way of specifying formally the interaction techniques, allowing them to be documented, archived and implemented (perhaps even automatically generated) on a wide variety of systems. Some attempts have already been made to achieve this kind of specification [67, 68] using techniques such as grammars and state transition diagrams, while the more formal approaches use Z[69] or LOTOS[70], for example.

7.3. System Configurations

A wide range of 3-DOF and 6-DOF input devices is now available, and although it is often possible to map the data from these devices directly to simple manipulation and navigation tasks, the more complex tasks (which are even less well suited to 2D input devices) still require careful consideration.

In the earlier, mouse-based techniques, the main problem being addressed was the lack of degrees of

freedom (interestingly, little consideration seems to have been given to using multiple devices, such as a mouse and a 2D joystick, simultaneously, despite being a low-cost solution which is easy to implement). If 3D input devices are used, the challenges change — for example, implementing a mechanism for menu selection or numerical input which is accurate enough to be usable despite tracker data which suffers from interference. Mine[31] presents a useful collection of techniques which address these and other problems.

Through a combination of real and virtual controller[48] devices it may be possible to create a range of interaction techniques that work best with 3-DOF/6-DOF devices but which can still be used with a mouse or other common 2D device. These techniques would be useful when distributing applications widely with little control over the delivery platform and its peripherals — via the Internet for example.

7.4. The Importance of VRML

The Virtual Reality Modelling Language (VRML) has made a significant impact on interactive computer graphics despite the relatively short time between its conception (early 1994), the refinement and publication of the VRML 1.0 and 2.0 standards, and the adoption of VRML97 as an ISO/IEC standard[71] (scheduled for late 1997).

VRML allows and encourages the sharing (via the Internet) of re-usable, interoperable components with a well-defined interface (using its prototyping mechanisms). Three-dimensional user interfaces may easily be prototyped using VRML objects, while the event/route mechanism allows even inexperienced users to re-configure existing interface objects to suit their own requirements (we describe elsewhere[72] how, using only a few VRML nodes, it is possible to create a "Navigation Metaphor Construction Kit" capable of implementing at least 12 distinct metaphors, including several described in Section 4, simply by changing routes and the location of objects in the scene graph). These components — known as "first-class" user interface objects [73, 72] — are tightly-coupled to the application or world objects in the scene, and have all the functionality of VRML available to them.

The VRML standard itself is the result of a co-ordinated effort by a large community spread across the Internet. As it becomes more widely-adopted we might expect to see increased development and sharing of user interface objects among this community, which in turn will result in the accelerated evolution of 3D interaction techniques. One of the community's "working groups", the *Widgets Working Group*[74] aims to produce a repository of re-usable 3D user interface components along with a taxonomy for classifying them. This repository will initially contain a set of core components, but will grow as developers submit their own work. It is recognised that there is a tension between establishing a "standard" widget set and allowing developers the creativity to create their own solutions (potentially bad ones: "novel and useful interfaces as well as novel and useless interfaces"[73]), but in contrast to many other systems, VRML makes it easy for developers to re-use and extend the widgets used. In particular, the appearance and geometry of widgets may easily be overridden when they are used. The vocabulary of 3D interaction is still so immature at this stage that this kind of flexibility is important.

One current shortcoming of VRML is that most of the interactive behaviour attached to user interface components must be written in scripting languages such as Java or JavaScript, which even for simple interaction techniques can result in complex logic if constraints are to be maintained. An alternative approach, adopted by VR toolkits such as Metis[75] and VB2[11] might be to add an engine for solving constraints; describing the behaviour of interaction techniques is often much simpler when using constraints, and may offer a performance benefit[75]. How such a constraint engine might exist alongside VRML's event model requires investigation.

8. Conclusions

This paper has presented a range of 3D interaction techniques for object manipulation, viewpoint manipulation and application control, as well as a description of the evolution of these techniques up to the present.

A number of conclusions may be drawn from this discussion:

- There is a large body of work on 3D interaction techniques, but this is presented in papers or is embodied within a variety of different systems. Consolidation of this information is required so that the various techniques can be easily described, shared and implemented by those interested in advancing the field.
- The dissemination of these descriptions and implementations will be helped by the adoption of open standards such as VRML.
- As they become more widespread there will be more opportunities to evaluate the usability of 3D interaction techniques and their relationship to various input devices and system configurations. Alongside this there should be an increased understanding of 3D evaluation techniques.

9. Acknowledgements

The author would like to thank Sabine Coquillart and the anonymous referees who provided valuable feedback on this paper. Thanks also to Howell Istance, Peter Innocent, John Edwards, Russell Turner, Matthias Wloka, Greg Seidman and Sascha Becker for helpful discussions on 3D user interfaces.

References

1. Myron Krueger. *Artificial Reality II.* Addison-Wesley, Reading, MA, 1991.
2. C Cruz-Neira, D J Sandin, T A DeFanti, R V Kenyon, and J C Hart. The CAVE: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):65–72, June 1992.
3. R J K Jacob, J J Leggett, B A Myers, and R Pausch. Interaction styles and input/output devices. *Behaviour & Information Technology*, 12(2):69–79, 1993.
4. Brenda K Laurel. Interface as mimesis. In D. A. Norman and S Draper, editors, *User-Centered System Design*, pages 67–85. Lawrence Erlbaum Associates, 1986.
5. Chris Hand. A survey of 3-D input devices. Technical Report TR94/2, Department of Computer Science, De Montfort University, UK, 1994.
6. David J Sturman and David Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, pages 30–39, January 1994.
7. William Buxton. There's more to interaction than meets the eye: Some issues in manual input. In J Preece, L Keller, and H Stolk, editors, *Human-Computer Interaction*, pages 122–137. Prentice-Hall, 1990. Also appeared in Norman, D A and Draper, D W (Eds) (1986) *User Centered System Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates: Hillsdale, NJ. USA.
8. I E Sutherland. Head mounted three dimensional display. In *Proceedings of the Fall Joint Computer Conference*, volume 33, pages 757–764, 1968.
9. André LeBlanc, Prem Kalra, Nadia Magnenat Thalmann, and Daniel Thalmann. Sculpting with the 'ball and mouse' metaphor. In *Proceedings of Graphics Interface '91*, pages 152–159, 1991.
10. Tinsley A Galyean and John F Hughes. Sculpting: an interactive volumetric modeling technique. In *Proceedings of SIGGRAPH'91*, pages 267–274. ACM: New York, July 1991.
11. Enrico Gobbetti, Jean-Francis Balaguer, and Daniel Thalmann. VB2: An architecture for interaction in synthetic worlds. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST'93)*, pages 167–178, 1993.
12. Tamotsu Murakami and Naomasa Nakakima. Direct and intuitive input device for 3-D shape deformation. In *Proceedings of CHI'94*, pages 465–470. ACM SIGCHI, April 1994.
13. Chris Shaw and Mark Green. Two-handed polygonal surface design. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST'94)*. ACM, November 1994.
14. Eric A. Bier. Snap-dragging in three dimensions. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, pages 193–203. ACM SIGGRAPH, March 1990.
15. Dan Venolia. Facile 3D direct manipulation. In *Proceedings of INTERCHI'93*, pages 31–36, April 1993.
16. Stephanie Houde. Iterative design of an interface for easy 3-D direct manipulation. In *Proceedings of CHI'92*, pages 135–142. ACM SIGCHI, May 1992.
17. T Yoshimura, Y Nakamura, and M Sugiura. 3D direct manipulation interface: Development of the zashiki-warashi system. *Computers & Graphics*, 18(2):201–207, 1994.
18. Michael Chen, S Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. In *Proceedings of SIGGRAPH'88*, pages 121–129. ACM SIGGRAPH, August 1988.
19. Colin Ware. Using hand position for virtual object placement. *The Visual Computer*, 6:245–253, 1990.
20. Colin Ware and Steven Osborne. Exploration and virtual camera control in virtual three dimensional environments. In *Proceedings of 1990 Symposium on Interactive 3D Graphics*, pages 175–183. ACM SIGGRAPH, 1990.
21. E. Gobbetti and J.-F. Balaguer. An integrated environment to visually construct 3D animations. In *Proceedings of the 22nd annual ACM conference on Computer graphics (SIGGRAPH'95)*, pages 395–398, Los Angeles, CA, August 1995.
22. Robert C Zeleznik, Kenneth P Herndon, Daniel C Robbins, Nate Huang, Tom Meyer, Noah Parker,

and John F Hughes. An interactive 3D toolkit for constructing 3D widgets. In *Proceedings of SIGGRAPH'93*, pages 81–84. ACM SIGGRAPH, 1993.

23. Jock D Mackinlay, Stuart K Card, and George G. Robertson. Rapid controlled movement through a virtual 3D workspace. In *Proceedings of SIGGRAPH'90*, pages 171–176. ACM SIGGRAPH, August 1990.

24. Glenn E Krasner and Stephen T Pope. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *JOOP*, 1(3), August 1988.

25. Ben Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, pages 57–62, August 1983.

26. Edwin L. Hutchins, James D. Hollan, and Donald A Norman. Direct manipulation interfaces. In D. A. Norman and S Draper, editors, *User-Centered System Design*, pages 87–124. Lawrence Erlbaum Associates, 1986.

27. Eric A. Bier. Skitters and jacks: interactive 3D positioning tools. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, pages 183–196. ACM: New York, October 1986.

28. Gregory M Nielson and Dan R Olsen, Jr. Direct manipulation techniques for 3D objects using 2D locator devices. In *1986 Symposium on Interactive 3D Graphics*, pages 175–182. ACM: New York, October 1986.

29. Christer Carlsson and Olof Hagsand. DIVE - a multi-user virtual reality system. In *Proceedings of VRAIS'93*, 1993.

30. Jiandong Liang and Mark Green. JDCAD: A highly interactive 3D modeling system. *Computers & Graphics*, 18(4):499–506, 1994.

31. Mark Mine. Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. Technical Report TR96-029, Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, USA, 1996.

32. Jeffrey S Pierce, Andrew Forsberg, Matthew Conway, Seung Hong, Robert Zeleznik, and Mark R Mine. Image plane interaction techniques in 3D immersive environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, Providence, RI, 1997. ACM: New York.

33. Richard A Bolt. 'put-that-there': Voice and gesture at the graphics interface. In *Proceedings of SIGGRAPH'80*, pages 262–270. ACM: New York, July 1980.

34. Christopher Schmandt. Spatial input/output correspondence in a stereoscopic computer graphics work station. *Computer Graphics*, 17(3):253–261, July 1983.

35. Ken Hinckley, Randy Pausch, John C Goble, and Neal F Kassell. Passive real-world interface props for neurosurgical visualization. In *Proceedings of CHI'94*, pages 452–458. ACM SIGCHI, April 1994.

36. Emanuel Sachs, Andrew Roberts, and David Stoops. 3-Draw: A tool for designing 3D shapes. *IEEE Computer Graphics and Applications*, pages 18–26, November 1991.

37. Thomas G Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. In *Proceedings of CHI + GI'87*, pages 189–192, 1987.

38. S S Fels. Building adaptive interfaces with neural networks: The glove-talk pilot study. Technical Report CRG-TR-90-1, Department of Computer Science, University of Toronto, Toronto, Canada, February 1990.

39. Gregory B Newby. Gesture recognition using statistical similarity. In *Proceedings of the conference Virtual Reality and Persons with Disabilities*, 1993.

40. C Hand, I Sexton, and M Mullen. A linguistic approach to the recognition of hand gestures. In *Ergonomics Society/IEE conference on 'Designing Future Interaction'*, University of Warwick, England, April 1994.

41. David J Sturman. *Whole-Hand Input*. PhD thesis, Massachusetts Institute of Technology, February 1992.

42. W Buxton and B A Myers. A study in two-handed input. In *Proceedings of the CHI'86 Conference on Human Factors in Computing Systems*, pages 321–326. ACM: New York, 1986.

43. Rupert England. Sensory-motor systems in virtual manipulation. In Karen Carr and Rupert England, editors, *Simulated and Virtual Realities: Elements of Perception*, pages 131–177. Taylor & Francis, 1995.

44. Margaret Minsky, Ming Ouh-Young, Oliver Steele, Frederick P Brooks, and Max Behensky. Feeling and seeing: Issues in force display. In *Proceedings of SIGGRAPH'90*, pages 235–243, 1990.

45. W Buxton. The three mirrors of interaction: a

holistic approach to user interfaces. In L W MacDonald and J Vince, editors, *Interacting with virtual environments*. Wiley, 1994.

46. Nancy Hays. Nobody walks in VR – they all fly. *IEEE Computer Graphics and Applications*, page 85, May 1993.
47. Dieter W Fellner and Oliver Jucknath. MRTspace – multi-user 3D environments using VRML. In *Proceedings of WebNet'96*, San Francisco, CA, October 1996.
48. Jock D Mackinlay, Stuart K Card, and George G Robertson. A semantic analysis of the design space of input devices. *Human Computer Interaction*, 5:145–190, 1990.
49. Frederick P Brooks, Jr. Grasping reality through illusion – interactive graphics serving science. In *Proceedings of CHI'88*, pages 1–11. ACM: New York, May 1988.
50. Cary B. Phillips, Norman I Badler, and John Granieri. Automatic viewing control for 3D direct manipulation. In *1992 Symposium on Interactive 3D Graphics*, pages 71–74. ACM SIGGRAPH, 1992.
51. M Gleicher and A Witkin. Through-the-lens camera control. In *Proceedings of SIGGRAPH'92*, pages 331–340. ACM, 1992.
52. Russell Turner, Francis Balaguer, Enrico Gobbetti, and Daniel Thalmann. Physically-based interactive camera motion control using 3D input devices. In *Proceedings of Computer Graphics International '91*, pages 135–145, 1991.
53. Alias Research, Inc. *ALIAS Reference Guide*. Toronto, Ontario, Canada., 1994.
54. R Stoakley, M J Conway, and R Pausch. Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings of CHI'95*, pages 265–272. ACM SIGCHI, March 1995.
55. M Deering. High resolution virtual reality. In *Proceedings of SIGGRAPH'92*, pages 195–202. ACM SIGGRAPH, 1992.
56. Colin Ware, Kevin Arthur, and Kellogg S Booth. Fish tank virtual reality. In *Proceedings of INTERCHI'93*, pages 37–42, April 1993.
57. James J Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.
58. Ian P Howard and W B Templeton. *Human Spatial Orientation*. John Wiley & Sons, 1966.
59. Richard H Y So and Michael J Griffin. Head-coupled virtual environment with display lag. In Karen Carr and Rupert England, editors, *Simulated and Virtual Realities: Elements of Perception*, pages 103–111. Taylor & Francis, 1995.
60. C M Oman. Sensory conflict in motion sickness: an observer theory approach. In Stephen R Ellis, editor, *Pictorial Communication in Real and Virtual Environments*, pages 362–376. Taylor & Francis, 1991.
61. Mel Slater, Anthony Steed, and Martin Usoh. The virtual treadmill: A naturalistic metaphor for navigation in immersive virtual environments. In M Goebel, editor, *Proceedings of the Eurographics Workshop on Virtual Reality*, pages 71–83, Barcelona, September 7th 1993.
62. Larisa Matejic. LOG: Building 3D user interface widgets by demonstration. Technical Report CS-93-22, Dept of Computer Science, Brown University USA, May 1993.
63. Mathias M Wloka and Eliot Greenfield. The virtual tricorder. Technical Report CS-95-05, Department of Computer Science, Brown University, Providence RI, USA, March 1995.
64. M M Wloka. Interacting with virtual reality. In J Rix, S Haas, and J Teixeira, editors, *Virtual Prototyping – Virtual Environments and the Product Development Process*. Chapman & Hall, 1995.
65. I. Scott MacKenzie and William Buxton. Extending fitts' law to two-dimensional tasks. In *Proceedings of CHI'92*, pages 219–226, May 1992.
66. J J Gibson. *The Senses Considered as Perceptual Systems*. Houghton Mifflin, Boston, 1966.
67. David A Carr. Specification of interface interaction objects. In *Proceedings of CHI'94*, pages 372–378. ACM: New York, April 1994.
68. Lutz Kettner. A classification scheme of 3D interaction techniques. Technical Report B 95-05, Freie Universitat Berlin, Takustr. 9, 14195 Berlin, Germany, April 1995.
69. D J Duke and M D Harrison. Abstract interaction objects. Report SM/WP1, ESPRIT AMODEUS-2 Project, University of York, UK, November 1992.
70. Fabio Paternò. An object-oriented approach to the design of graphical user interface systems. Technical Report TR-92-046, ICSI, Berkeley, California, USA, August 1992.
71. The virtual reality modeling language draft for international standard (ISO/IEC DIS 14772-1). http://www.vrml.org/Specifications/VRML97/DIS/, April 1997.

72. John D M Edwards and Chris Hand. MaPS: Movement and planning support for navigation in an immersive VRML browser. In *Proceedings of The Second Symposium on the Virtual Reality Modeling Language (VRML'97)*, pages 65–73, Monterey, CA, February 1997. ACM: New York.

73. D. Brookshire Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-dimensional widgets. *Computer Graphics*, 25(2):183–188, March 1992.

74. Widgets working group draft charter. http://zing.ncsl.nist.gov/~gseidman/vrml/wwg/charter.html, June 1997.

75. Russell Turner, Song Li, and Enrico Gobbetti. Metis: An object-oriented toolkit for constructing virtual reality applications. In Farhad Arbab and Philipp Slusallek, editors, *Proceedings of the Sixth Eurographics Workshop on Programming Paradigms in Graphics*, pages 79–90, Budapest, Hungary, 8th September 1997.