CHAPTER 4

3D User Interface Input Hardware

This chapter continues our discussion of the hardware commonly used in 3D UIs. We examine a variety of input devices that are used in both immersive and desktop applications and their effect on 3D UIs. We also examine how to choose input devices for different 3D applications by looking at some important device attributes, taxonomies, and evidence from empirical studies.

4.1. Introduction

As we saw in Chapter 3, choosing appropriate output devices is an important component of designing, developing, and using a 3D application, since they are the primary means of presenting information to the user. An equally important part of 3D UI design is choosing the appropriate set of input devices that allow the user to communicate with the application. For example, we might need to track the user's head or let him interact with the application using his voice. Perhaps our 3D UI has specific needs, such as letting users throw 3D virtual paint around using a paint bucket, or providing a way to record handwritten notes. As with output devices, there are many different types of input devices to choose from when developing a 3D UI, and some devices may be more appropriate for certain tasks than others.

This chapter is meant as a guide to the types of input devices available and how they are used in 3D UIs. As with the previous chapter, this

chapter is not meant to be a fully exhaustive discussion on all the input devices ever developed or the technical details of input device design; rather, it presents a representative sample of devices commonly found in 3D applications. See Sherman and Craig (2003) and Burdea and Coiffet (2003) for other expositions on 3D input hardware.

We encourage you to think about the content in this chapter while exploring the rest of the book and especially when thinking about interaction techniques. This interaction technique/input device connection is important because of the distinction between the two. Input devices are just the physical tools that are used to implement various interaction techniques (Foley and Wallace 1974). In general, many different interface techniques can be mapped onto any given input device. The question is how natural, efficient, and appropriate the mapping between a given input device and a given technique will be.

4.1.1. Input Device Characteristics

Many different characteristics can be used to describe input devices. One of the most important is the degrees of freedom (DOF) that an input device affords. A degree of freedom is simply a particular, independent way that a body moves in space. A device such as a tracker generally captures three position values and three orientation values for a total of six DOF. For the most part, a device's DOF gives an indication of how complex the device is and the power it has in accommodating various interaction techniques.

Another way to characterize input devices is by the input type and frequency of the data (i.e., reports) they generate. Data reports are composed of either discrete components, continuous components, or a combination of the two. Discrete input device components typically generate a single data value (i.e., a Boolean value or an element from a set) based on the user's action. They are often used to change modes in an application, such as changing the drawing mode in a desktop 3D modeling program, or to indicate the user wants to start performing an action, such as instantiating a navigation technique. Continuous input device components generate multiple data values (i.e., real-valued numbers, pixel coordinates, etc.) in response to a user's action and, in many cases, regardless of what the user is doing (tracking systems and bend-sensing gloves are examples). In many cases, input devices combine discrete and continuous components, providing a larger range of device-to-interaction technique mappings.

Input devices can also be described based on how much physical interaction is required to use the device. For example, *purely active* input devices are devices that require the user to actually perform some physical action before data is generated. In other words, the input device will not provide any information to the computer unless it is manipulated in some way. Otherwise, the device just sits there and does nothing. Purely active input devices can have both discrete components (e.g., buttons) and manually driven continuous components, which means that the user must manipulate the component in order to generate the device's continuous behavior. Trackballs, sliders, and dials are examples of manually driven continuous components, and they allow the user to generate sequences of values from a given range.

Purely passive input devices do not require any physical action for the device to function. In other words, these devices continue to generate data even if they are untouched. Of course, users can manipulate these devices like active input devices, but this is not a necessity. They are sometimes called *monitoring* input devices (Sherman and Craig 2003), and they are very important in many 3D UIs. For example, a tracker is a device that will continually output position and/or orientation records even if the device is not moving. These devices are important when we want to know where something is in the virtual space and we do not want to have to keep asking for it. A perfect example of this is head tracking, which is a requirement for 3D audio displays and active viewer motion parallax in visual displays.

Finally, input devices can be categorized by their intended use. For example, some devices are designed to specifically determine position and/or orientation information (locators), while others are designed to produce a real number value (valuators) or to indicate a particular element of a set (choice). Other input device characteristics include whether the device measures relative (i.e., the difference between the current and past measurement) or absolute (i.e., measurements based on a constant point of reference) values, is a direct or indirect controller, or allows for position or rate control (Jacob 1996).

4.1.2. Chapter Roadmap

Most of this chapter contains a discussion of a variety of different input devices used in 3D interfaces and how they affect interaction techniques. These devices are broken up into the following categories:

- desktop input devices
- tracking devices
- 3D mice
- special-purpose input devices
- direct human input

In section 4.2, we discuss devices that have been traditionally used in 2D desktop applications but work well in 3D UIs and 6-DOF devices that were designed specifically for 3D desktop interaction. In section 4.3, we examine user tracking devices, which are very important when we want to know a user's or physical object's location in 3D space. Section 4.4 presents a variety of 3D mice, which are defined to be devices that combine trackers with a series of buttons and other discrete components in a given configuration. In section 4.5, we present a hodgepodge of specialized input devices that do not fit well into our other categories. Finally, section 4.6 describes direct human input, which includes speech, bioelectric, and brain input. After we explore this collection of input devices, section 4.7 provides valuable information on strategies for building custom input devices. In section 4.8, we present some ideas and guidelines for choosing input devices for a particular task or application.

4.2. Desktop Input Devices

There are many input devices that are used in desktop 3D UIs. Many of these devices have been used and designed for traditional 2D desktop applications such as word processing, spreadsheets, and drawing. However, with appropriate mappings, these devices also work well in 3D UIs and in 3D applications such as modeling and computer games. Some desktop input devices have also been developed with 3D interaction in mind. These devices can provide up to 6 DOF, allowing users to manipulate objects in 3D space, and are specifically designed for interaction on the desktop. Of course, most of these devices could also be used in more immersive 3D UIs that use surround-screen displays or HMDs, although some would be more appropriate than others.

In this section, we discuss

- keyboards
- 2D mice and trackballs

- pen-based tablets
- joysticks
- 6-DOF devices for the desktop

In general, these devices are purely active because the user must physically manipulate them to provide information to the 3D application.

4.2.1. Keyboards

The keyboard is a classic example of a traditional desktop input device that contains a set of discrete components (a set of buttons). They are commonly used in many desktop 3D applications from modeling to computer games. For example, the arrow keys are often used as input for simple travel techniques in first-person shooter computer games. Unfortunately, bringing the standard keyboard into more immersive 3D environments is not practical when users are wearing HMDs or in surround-screen environments, since users are typically standing. However, some less immersive workbench and personalized hemispherical displays (see Figure 3.13 in Chapter 3) can make use of traditional keyboards.

Because entering alphanumeric characters is important in many immersive 3D applications that use HMDs and surround-screen displays, other less conventional devices are needed in these environments. An example of such a device is the chord keyboard, which was first introduced by Engelbart and English (1968). This device is held in one hand, and the user presses combinations of keys as a single chord for alphanumeric input. See Chapter 9 on symbolic input for a more detailed discussion of these devices.

4.2.2. 2D Mice and Trackballs

Two-dimensional mice and trackballs are another classic example of desktop input devices made popular by the windows, icons, menus, and pointers (WIMP) interface metaphor (van Dam 1997). The mouse is one of the most widely used devices in traditional 2D input tasks and comes in many different varieties. The trackball is basically an upside-down mouse. Instead of moving the whole device to move the pointer, the user manipulates a rotatable ball embedded in the device. One of the advantages of the trackball is that it does not need a flat 2D surface to operate, which means that it can be held in the user's hand and will still operate

correctly. Regardless of the physical design of the mouse or trackball, these devices have two essential components. The first is a manually continuous 2D locator for positioning a cursor and generating 2D pixel coordinate values. The second is a set of discrete components (usually one to three buttons). Mice and trackballs are relative devices that report how far they move rather than where they are. As with keyboards, they are commonly used in many different 3D applications and provide many different choices for mapping interaction technique to task. For example, they are often combined with keyboards in computer games to enable more complex travel techniques. The keyboard may be used for translation while the mouse or trackball is used to rotate the camera so the user can see the 3D environment (e.g., look up, look down, turn around). More details on how mice and trackballs (and 2D locators in general) are used in 3D interaction techniques can be found in Chapters 5 and 6.

Mice and trackballs have the same problem as the keyboard in that they are not designed to be brought into more immersive 3D environments. Because a mouse needs to be placed on a 2D surface in order for the locator to function properly, it is difficult to use with these displays. Since the trackball can be held in one hand and manipulated with the other, it can be used in immersive 3D environments, and it has also been successfully incorporated into a 3D interface using a workbench display (Forsberg et al. 1997). However, in most cases, 3D mice are used in immersive 3D interfaces because of their additional DOF. Section 4.5 discusses 3D mice.

4.2.3. Pen-Based Tablets

Pen-based tablets (see Figure 4.1) and handheld personal digital assistants (PDAs) generate the same types of input that mice do, but they have a different form factor. These devices have a manually continuous component (i.e., a 2D locator) for controlling a cursor and generating 2D pixel coordinate values when the stylus is moving on or hovering over the tablet surface. Additionally, the stylus or the tablet itself can have various buttons for generating discrete events. In contrast to mice, pen-based tablets are absolute devices, meaning that the device reports where the stylus is in relation to the tablet surface. Large pen-based tablets are not appropriate for most fully immersive visual displays because of their weight, but smaller tablets and PDAs have been integrated successfully into 3D UIs in these environments (Poupyrev, Tomokazu et al. 1998; Watsen et al. 1999). Larger pen-based tablets can be used in 3D applications

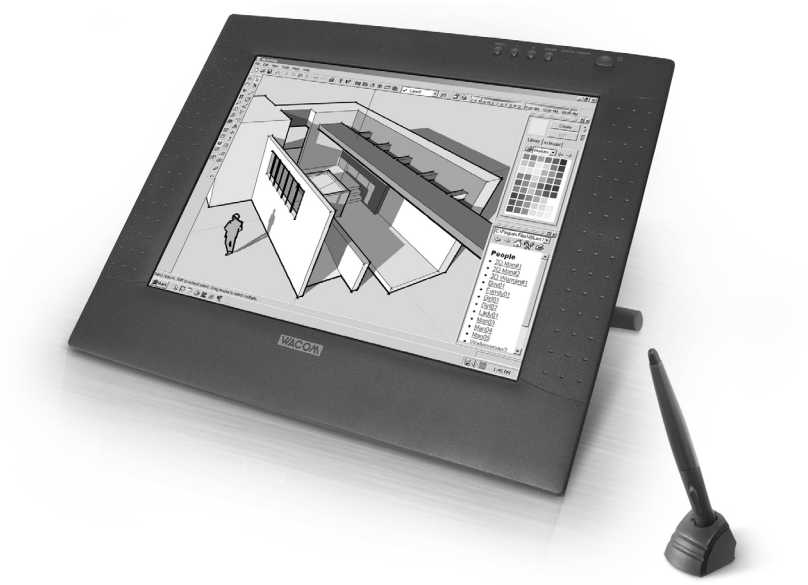

Figure 4.1 *A large pen-based LCD tablet allowing the user to draw directly on the screen. (Photograph courtesy of Wacom Technology and @Last Software)*

where the user is sitting, such as with desktop 3D displays, some workbench and single-wall displays, and small hemispherical displays. These types of devices are becoming more popular in both desktop 3D and immersive VE applications, because they give the user the ability to interact with a "pen and paper" interface, and they allow the user to bring 2D interaction techniques such as handwriting and menu-based techniques into 3D environments (Forsberg et al. 1998; Poupyrev, Tomokazu et al. 1998; Watsen et al. 1999).

4.2.4. Joysticks

Joysticks are another example of input devices traditionally used on the desktop and with a long history as a computer input peripheral. These devices are similar to mice and pen-based tablets in that they have a combination of a manually continuous 2D locator and a set of discrete components such as buttons and other switches. However, there is an important distinction between the mouse and joystick. With a mouse, the cursor stops moving as soon as the mouse stops moving. With a joystick, the cursor typically continues moving in the direction the joystick is pointing. To stop the cursor, the joystick's handle must be returned to the

Figure 4.2 *Simple joysticks have evolved into sophisticated game controllers. (Photograph courtesy of Joseph J. LaViola Jr.)*

neutral position. This type of joystick is commonly called an *isotonic joystick,* and the technique is called rate control (as opposed to position control). Many console video game systems make use of different joystick designs in their game controllers (see Figure 4.2). Joysticks can also be augmented with haptic actuators, making them haptic displays as well (Burdea 1996).

Isometric joysticks have also been designed. Isometric devices have a large spring constant so they cannot be perceptibly moved. Their output varies with the force the user applies to the device. A translation isometric device is pushed, while a rotation isometric device is twisted. A problem with these devices is that users may tire quickly from the pressure they must apply in order to use them. Figure 4.3 shows an example of such a device.

Joysticks have been used as input devices in computer games for many years. They are frequently used in driving and flight simulation games, and when integrated into game controllers, they are the input device of choice with console video game systems. Additionally, they are sometimes used in CAD/CAM applications. Since joysticks are designed primarily for desktop applications and console video game systems, they are rarely used in 3D UIs that employ HMDs or surround-screen visual displays. However, since many joysticks are handheld, they could easily be brought into these types of environments.

Figure 4.3 *An isometric 3D input device. (Photograph courtesy of Andrew Forsberg, Brown University Graphics Group)*

4.2.5. Six-DOF Input Devices for the Desktop

The devices we have discussed so far can all be used in 3D interfaces, and they can allow the user to interact with 3D objects, but they were not specifically designed for this purpose. Figure 4.4 shows two examples of 6 DOF input devices that were developed specifically for 3D interaction on the desktop. Slight push and pull pressure of the fingers on the cap of the device generates small deflections in *x*, *y*, and *z*, which moves objects dynamically in the corresponding 3 axes. With slight twisting and tilting of the cap, rotational motions are generated along the 3 axes. The particular devices shown in Figure 4.5 also have a series of buttons which can be

Figure 4.4 *Two 6-DOF desktop input devices: the SpaceMouse Plus and the SpaceBall 5000. (Developed by 3Dconnexion)*

programmed with any frequently used function or user-defined keyboard macro.

This type of device is commonly used in desktop 3D applications for manipulating virtual objects. They were originally developed for telerobotic manipulation and are commonly used today by 3D designers and artists with CAD/CAM and animation applications. They do not replace the mouse; rather, they are used in conjunction with it. One hand on the motion controller positions the objects in 3D space, while the other hand with the mouse can simultaneously select menu items and edit the object. These devices are rarely used in more immersive environments because the device works best when grounded and not carried in the user's hands. Additionally, they can be difficult to use when trying to make fine manipulations, and it takes practice to become proficient with them.

4.3. Tracking Devices

In many 3D applications, it is important for the UI to provide information about the user or physical object's location in 3D space. For example, the system might need the user's head position and orientation so that full-motion parallax and stereoscopic depth cues can be included in the application. In another system, the UI might require information about the bending of the user's fingers so that a virtual hand corresponding to the user's physical hand can be rendered. In most cases, we want this information sent to the 3D application automatically without the user having to signal the computer system to collect it. Therefore, most of these devices are purely passive and generate information continuously. In this section, we examine three of the most common tracking devices:

- motion trackers
- eye trackers
- data gloves

We also explore their relationship to 3D UIs for desktop 3D and VE applications.

4.3.1. Motion Tracking

One of the most important aspects of 3D interaction in virtual worlds is providing a correspondence between the physical and virtual environments. As a result, having accurate tracking is a crucial part of making

interaction techniques usable within VE applications. In fact, motion tracking is fundamental to many interaction techniques described in Part III of this book. The critical characteristics of motion trackers include their range, latency (delay between the time a motion occurs and when it is reported), jitter (noise or instability), and accuracy. Currently, there are a number of different motion-tracking technologies in use, which include

- magnetic tracking
- mechanical tracking
- acoustic tracking
- inertial tracking
- optical tracking
- hybrid tracking

See Foxlin (2002); Welch and Foxlin (2002); and Allen, Bishop, and Welch (2001) for more details on motion-tracking technology.

Magnetic Tracking

Magnetic trackers use a transmitting device that emits a low-frequency magnetic field. A small sensor, the receiver, determines its position and orientation relative to this magnetic source. The range of such trackers varies, but they typically work within a radius of 4 to 30 feet. Figure 4.5 shows an example of a magnetic tracking system. It uses a small emitter and receivers and has better accuracy than larger range systems. However, its range is limited to a 4-foot radius, which means the device is not appropriate for large display environments such as surround-screen visual displays or even HMDs where the user needs a lot of space to roam. It is primarily used with conventional monitors (for fishtank VR) and small workbench displays where range of the device is not a critical factor.

In general, magnetic tracking systems are accurate to within 0.1 inches in position and 0.1 degrees in orientation. Their main disadvantage is that any ferromagnetic or conductive (metal) objects present in the room with the transmitter will distort the magnetic field, reducing the accuracy. These accuracy reductions can sometimes be quite severe, making many interaction techniques, especially gesture-based techniques, difficult to use. Distortions can be handled with calibration routines and filtering algorithms (Kindratenko 2000), but doing so can increase start-up time and online computational overhead.

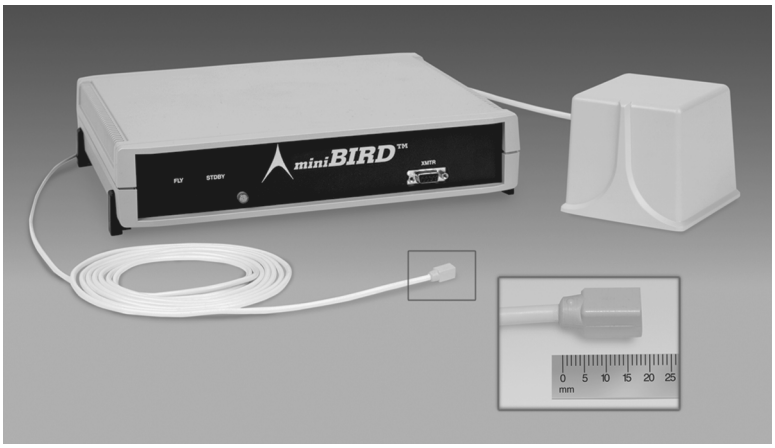

Figure 4.5 *A magnetic tracker consisting of an electronics unit, a magnetic field generator (on the right), and receivers (the small sensors connected to the front of the electronics unit, also shown in the inset) that track the user or object. (Photograph courtesy of Ascension Technology Corporation)*

Mechanical Tracking

Mechanical trackers have a rigid structure with a number of interconnected mechanical linkages combined with electromechanical transducers such as potentiometers or shaft encoders. One end is fixed in place, while the other is attached to the object to be tracked (usually the user's head or hand). As the tracked object moves, the linkages move as well, and measurements are taken from the transducers to obtain position and orientation information. Arm-mounted visual displays use this type of tracking technology (see Figure 3.16 in Chapter 3). Additionally, many ground-referenced force-feedback devices (see section 3.4.3 in Chapter 3) are mechanically based, making them trackers as well as force displays. Mechanical trackers are very accurate and transmit information with very low latencies. However, they are often bulky, limiting the user's mobility and making it difficult to use physically based navigation techniques (see Chapter 6).

Acoustic Tracking

Acoustic tracking devices (see Figure 4.6) use high-frequency sound emitted from source components and received by microphones. The source may be on the tracked object, with the microphones placed in the

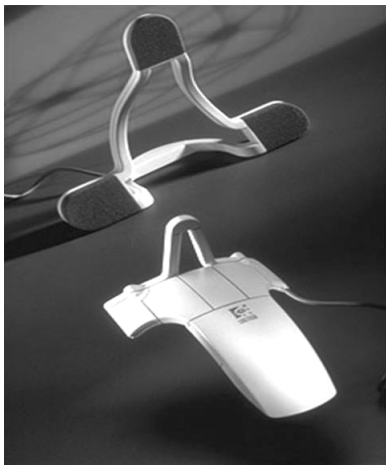

Figure 4.6 *An acoustic tracking device (the Fly Mouse). The mouselike device generates the acoustic signals from which the receiver determines position and orientation information. (Photograph courtesy of Logitech International)*

environment (an *outside-in* approach), or the source may be in the environment, with the microphones on the tracked object (an *inside-out* approach). The dominant approach to determining position and orientation information with acoustic tracking is to use time-of-flight duration of ultrasonic pulses. In other words, the distance between emitter and receiver can be determined by the time it takes for an ultrasonic pulse to travel from source to destination multiplied by the speed of sound. From this distance, position can be estimated, and with more receivers, a set of three points can be used to determine orientation using triangulation.

The advantages of acoustic tracking systems are that they are relatively inexpensive and lightweight. However, these devices often have a short range and low sampling rates (compared with optical and inertial trackers, which can have sampling rates above 1 KHz), and their accuracy suffers if acoustically reflective surfaces are present in the room. Another disadvantage of acoustic tracking is that external noises such as jingling keys or a ringing phone can cause interference in the tracking signal and thus significantly reduce accuracy. As with any tracking system that has accuracy problems, many interaction techniques are difficult to use if an ultrasonic tracker loses signal, has significant jitter, or suffers from distortions anywhere within the tracking range (distortions usually increase significantly as the user moves toward a tracking boundary).

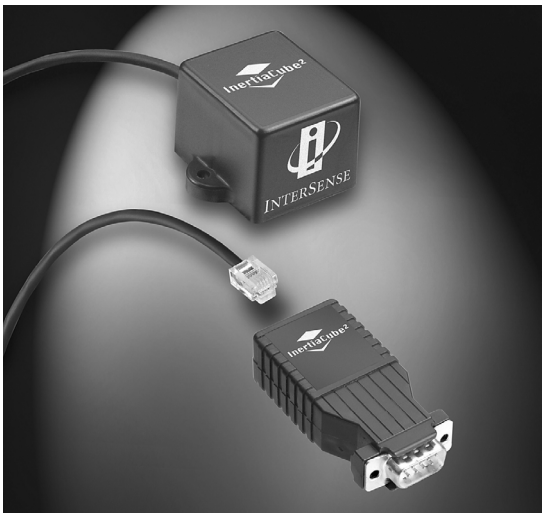

Figure 4.7 *An inertial tracker. The inertial sensors are located in the cube shown in the picture. (Photograph courtesy of InterSense, Inc.)*

Inertial Tracking

Inertial tracking systems (see Figure 4.7) use a variety of inertial measurement devices such as angular-rate gyroscopes and linear accelerometers. These devices provide derivative measurements (i.e., gyroscopes provide angular velocity, and linear accelerometers provide linear acceleration), so they must be integrated to obtain position and orientation information. Since the tracking system is in the sensor, range is limited to the length of the cord that attaches the sensor to the electronics unit (wireless tracking is also possible with these systems). In addition, these devices can produce measurements at high sampling rates. Inertial tracking systems were originally used in ships, submarines, and airplanes in the 1950s (Welch and Foxlin 2002). However, the weight of these devices prohibited their use in motion tracking until they became small enough to fit in microelectronic mechanical systems (MEMS).

The major limitation of inertial trackers is that they suffer error accumulation from sensor biases, noise, and drift (see Foxlin [2002] for a detailed discussion on error accumulation). Error accumulation can be severe with linear accelerometers, which is why most purely inertial tracking systems only track orientation. Although there are inertial navigation systems that can track position and orientation, they are used on ships and submarines, where tracking error within a mile is often accept-

able, in contrast to the subcentimeter accuracy required in 3D UIs. Gyroscopes also suffer from error accumulation, but this is less severe, and there are methods for compensating for this problem. For example, inertial trackers often handle error accumulation by using a gravitometer and compass measurements to prevent accumulation of gyroscopic drift.

Not tracking position severely limits what the interface designer can do in terms of using common interaction techniques. However, orientation-only tracking systems can be used in VEs for head tracking where the user will basically stand in one place and look around. A virtual travel technique (Chapter 6) can be used to allow translation through the environment.

Optical Tracking

Another approach to position and orientation tracking of users and physical objects is from measurements of reflected or emitted light. These types of trackers use computer vision techniques and optical sensors such as cameras, infrared emitters, or lateral effect diodes, which generate signals proportional to the position of incoming light along one axis (i.e., 2D displacement measurement). A variety of different cameras can be used, from simple desktop webcams to sophisticated high-resolution cameras with high sampling rates and pixel densities.

Like acoustic trackers, optical tracking systems use either outside-in or inside-out configurations. Outside-in systems have their sensors mounted at fixed locations in the environment, and tracked objects are marked with active or passive landmarks such as retroreflective makers or colored gloves. The number and size of these landmarks vary depending on the type of optical tracking system and how many DOF are required. In some cases, no landmarks are used at all (Starner et al. 1998). Inside-out systems place optical sensors on the user or tracked object while the landmarks are placed in the environment. There are many different landmarks that can be used, such as active LED beacons (Welch et al. 2001) or passive fiducials such as cards with recognizable patterns (Foxlin and Naimark 2003).

Setting up vision-based tracking systems can be difficult, since many parameters must be set in order to track the user or physical objects properly. These parameters include the number of cameras, the placement of the camera, what visual background is put up, and the design and placement of landmarks—whether they are in the environment or on the tracked user or object.

Figure 4.8 *An optical tracking device. The user wears the triangular sensor for tracking, and the scanner generates laser beams throughout the environment. (Courtesy of Ascension Technology Corporation)*

Figure 4.8 shows an example of an optical tracking device. It delivers accurate position and orientation tracking without environmental interference or distortion. Its miniaturized scanner reflects laser beams throughout the workspace. Each sensor, attached to a tracked object, instantly picks up the laser beams. Signals are then directed back to the scanner's DSP electronics for processing and transmission to the computer. Another vision-based approach is the HiBall system (see Figure 4.9), composed of a set of small, high-speed cameras, which are aligned in a self-contained housing and affixed to the user's head (Welch et al. 2001). A set of LEDs are placed in the environment, and their positions are surveyed. The LEDs are flashed at high speed in a specific known sequence, and providing the camera can see enough LEDs, the position and orientation of the tracker can be calculated directly through triangulation.

Theoretically, vision-based tracking systems have a clear advantage in that the user can be completely untethered from the computer. However, in practice this is often not the case. The tracking systems in figures 4.8 and 4.9 both have wires that attach to the device even though they are vision-based solutions. Even if the tracking system allows the user to be completely untethered, other input or output devices (an HMD or force-feedback display, for example) will require tethering between user and machine. With wireless technology becoming more powerful, these limitations are receding, making fully untethered systems more practical (Liebe et al. 2000; Hogue et al. 2003), especially in conjunction with projection-based displays (see Chapter 3, section 3.2.3).

Figure 4.9 *The HiBall Tracking System. The LED beacons are mounted on the ceiling, and the camera device is located in the handheld object. (Photograph of the HiBall-3000 Tracker courtesy of 3rd Tech, Inc.)*

The major disadvantage of vision-based trackers is occlusion. In many cases, the camera cannot pick up information about different parts of the user's body that are occluded by other parts. For example, it is difficult to extract information from all of the fingers when the hand is oriented in certain ways. Adding more cameras and landmarks can help to reduce the occlusion problem, but this increases the complexity of the tracking algorithm.

Hybrid Tracking

Hybrid trackers put more than one tracking technology together to help increase accuracy, reduce latency, and provide a better overall 3D interaction experience. In general, the individual tracking technologies are used to compensate for each other's weaknesses. An example of such a device is shown in Figure 4.10. This example combines inertial and ultrasonic tracking technologies. The inertial component measures orientation, and the ultrasonic component measures position, enabling the device to attain 6 DOF. Moreover, information from each component is used to improve the accuracy of the other. As a side note, this tracking system has the added advantage of being wireless, with the user wearing a small battery powered electronics box on her belt (Wormell and Foxlin 2003). The major difficulty with hybrid trackers is that more components produce more complexity. The extra complexity is warranted, however, if tracking accuracy is significantly improved.

Figure 4.10 *A wireless inertial/ultrasonic tracker. (Photograph courtesy of InterSense, Inc.)*

Another type of hybrid tracking combines video cameras and structured digital light projectors. Combining these two technologies allows for the capture of depth, color, and surface reflectance information for objects and participants in the environment. This approach was used at the University of North Carolina at Chapel Hill in its Office of the Future project (Raskar et al. 1998). An idealized drawing is shown in Figure 4.11.

Figure 4.11 *A stylized drawing of the Office of the Future. The room uses both cameras and structured light to track the user and perform scene acquisition. (Artwork by Andrei State, © University of North Carolina at Chapel Hill).*

Finally, hybrid trackers may incorporate other technologies into their devices, such as global positioning systems (GPS). GPS is often used in conjunction with accelerometers or gyroscopes for tracking in large-scale, outdoor, augmented reality environments, where it is impossible to fix sources or receivers to the environment (see Chapter 12 for more information on augmented reality).

4.3.2. Eye Tracking

Eye trackers are purely passive input devices used to determine where the user is looking. Eye-tracking technology is primarily based on computer vision techniques: the device tracks the user's pupils using corneal reflections detected by a camera. Devices can be worn (Figure 4.12) or embedded into a computer screen, making for a much less obtrusive interface. Other eye-tracking techniques include electro-oculography, which measures the skin's electric potential differences using electrodes placed around the eye, and embedding mechanical or optical reference objects in contact lenses that are worn directly on the eye (Duchowski 2003).

From a generic interaction perspective, eye-tracking systems have been used both as an evaluation tool and to interact with an application. For example, these devices are used to collect information about a user's eye movements in the context of psychophysical experiments, to get

Figure 4.12 *This user-worn eye-tracking device uses corneal reflections and pupil tracking to determine eye position. (Photograph courtesy of SR Research, Ltd.)*

application usage patterns to help improve the interface, or for training in visual inspection tasks (Duchowski et al. 2001). Eye-tracking systems are also used as input devices. An example would be a user controlling a mouse pointer strictly with his eyes. In the context of 3D interface design, active eye-tracking systems have the potential to improve upon many existing 3D interaction techniques. For example, there are numerous techniques that are based on gaze direction (e.g., gaze-directed steering, gaze-directed manipulation), which use the user's head-tracker as an approximation to where she is looking. Since the gaze vector is only accurate if the user is looking straight ahead, usability problems can occur if the user looks in other directions while keeping the head stationary. Eye-tracking devices might help improve these gaze-directed techniques, since the actual gaze from the user can be obtained. See Duchowski (2003), Wilder et al. (2002), and Jacob (1995) for more information on eye-tracking systems and their use in 3D applications.

4.3.3. Data Gloves

In some cases, it is useful to have detailed tracking information about the user's hands, such as how the fingers are bending or if two fingers have made contact with each other. Data gloves are input devices that provide this information. Data gloves come in two basic varieties: bend-sensing gloves and pinch gloves. In this section, we examine both of these glove types and also a research prototype that was developed to combine bend-sensing and pinch functionality.

Bend-Sensing Gloves

Bend-sensing data gloves are purely passive input devices used to detect postures (static configurations) of the hand. For example, the device can distinguish between a fist, a pointing posture, and an open hand. The raw data from the gloves is usually given in the form of joint angle measurements, and software is used to detect postures based on these measurements.

Many data gloves have been developed over the years using various kinds of sensors. For example, light-based sensors use flexible tubes with a light source at one end and a photocell at the other (Defanti and Sandin 1977). As the fingers are bent, the amount of light that hits the photocells varies, producing a measurement. Another light-based approach uses optical goniometer sensors consisting of flexible tubes with a reflective interior wall, a light source at one end, and a photosensitive detector on

Figure 4.13 *A bend-sensing data glove. (Photograph courtesy of Fifth Dimension Technologies, www.5dt.com)*

the other, which detects both direct and reflected light rays (Zimmerman, Lanier et al. 1987). Depending on the bending of the tubes, the detector changes its electrical resistance as a function of light intensity. These types of light-based sensors were used in older, first-generation data gloves. Today, more sophisticated sensor technology is used, such as fiber-optic sensors, resistive ink sensors, and strain-gauge bend sensors (Kramer 1991). Regardless of the type of sensor used, they are usually embedded in the glove or placed on the outer surface of the glove.

Data gloves typically have anywhere between five and 22 sensors. For example, a glove that has five sensors will usually measure one joint in each finger, while a glove with 18 sensors could measure at least two joints in each finger, abduction between fingers, wrist roll and yaw, and others. An example of a 16-sensor glove that uses fiber-optic sensors is shown in Figure 4.13.

From a 3D UI perspective, data gloves are commonly used for hand gesture and posture recognition, which can be applied to a variety of different interaction techniques. For example, a flick of the wrist could indicate the user wants to delete an object. A pointing posture could indicate a travel technique such as steering (Chapter 6). Often, hand postures and gestures are used as system commands in the context of system control techniques (see Chapter 8 for more details). Note that to recognize gestures (a series of postures or a moving posture), a motion tracker must be attached to the data glove.

In some 3D UIs, a virtual representation of a user's hand or hands is required. Data gloves with an associated tracking system can provide such a representation. In general, these types of representations are useful when the real world is completely blocked from the user's view (e.g., when using an HMD), and the user needs to see her hands in the scene with other virtual objects. For instance, a user might wish to get an understanding of where her hands are in relation to various dials and controls in a virtual car's interior.

One of the major advantages of bend-sensing gloves is that they provide a large number of DOF, making it possible to recognize a variety of hand gestures and postures as well as providing a representation of the user's hand in the 3D application. However, the user does have to wear the device, and there will always be a significant portion of the population for which the glove does not fit well. In addition, bend-sensing gloves sometimes need calibration on a user-by-user basis.

Pinch Gloves

The Pinch Glove (see Figure 4.14) system is an input device that determines if a user is touching two or more fingertips together. These gloves have a conductive material at each of the fingertips so that when the user pinches two fingers together, an electrical contact is made. These devices are often used for performing grabbing and pinching gestures in the context of object selection, mode switching, and other techniques (see Bowman, Wingrave et al. 2002 for details).

Figure 4.14 *A Pinch Glove is a user-worn discrete input device. (Photograph courtesy of Joseph J. LaViola Jr.)*

An interesting characteristic of this device is that the conductive cloth is also found on the back of the glove along the user's fingers and thumb, as shown in Figure 4.14. If a tracker is attached to the user's fingertip, for example, the user can make gestures that can be interpreted as simple sliders by simply making a cloth contact with the tracked fingertip and one of the cloth strips on the back of the other glove. When contact is made, the system can determine the location of the tracked fingertip as the user slides up and down the cloth strip. If another tracker is attached to the other glove, it is trivial to determine if the user's finger is moving toward the wrist or away from the wrist when making the gesture. This simple technique has been used to adjust object size or increase and decrease parameter values (LaViola 2000b).

Pinch Gloves are extremely light, reducing user fatigue in comparison to handheld input devices. They also provide for two-handed interaction (see Chapter 10). There are literally thousands of different pinching combinations that can be made using this device, which allows for a large design space of input-device-to-task mappings. However, only a handful of them are useful and ergonomic. Additionally, with extended use, the cloth contacts often deteriorate, making the gloves unusable.

Combining Bend-Sensing Data and Pinch Input

Both the Pinch Gloves and bend-sensing gloves have limitations. Although it is possible to determine if there is finger contact (e.g., index finger to thumb) with a bend-sensing glove, some form of hand gesture recognition is required, which will not be as accurate as the Pinch Glove (which has essentially 100% accuracy assuming the device is functioning properly). Conversely, one can get an idea of how the fingers are bent when using Pinch Gloves, but they provide only very rough estimates. Ideally, a data glove should have the functionality of both bend-sensing gloves and Pinch Gloves.

The Flex and Pinch input system is an example of an input device that combines the functionality of the Pinch Glove system with the bend-sensing technology of a data glove (see Figure 4.15). The pinch buttons, which are connected to a microcontroller, are made from conductive cloth and can be placed anywhere on the bend-sensing glove. This combination of hand-generated discrete and continuous events can make certain interaction techniques easier to perform, such as locking during a scaling operation or starting and stopping an object selection operation (LaViola 1999b).

Figure 4.15 *The Flex and Pinch input system combines bend-sensing input with discrete buttons. (Photograph courtesy of Joseph J. LaViola Jr.)*

4.4. 3D Mice

In the last section, we described tracking devices that are used to monitor the user or physical objects in 3D space. In many cases, specifically with motion trackers (see section 4.3.1), these tracking devices are combined with other physical device components such as buttons, sliders, knobs, and dials to create more functionally powerful input devices. We call these devices *3D mice* and define them broadly as handheld or worn input devices that combine motion tracking with a set of physical device components.

The distinguishing characteristic of 3D mice, as opposed to regular 2D mice, is that the user physically moves them in 3D space to obtain position and/or orientation information instead of just moving the device along a flat surface. Therefore, users can hold the device or, in some cases, wear it. Additionally, with orientation information present, it is trivial to determine where the device is pointing (the device's direction vector), a function used in many fundamental 3D interaction techniques (see Chapters 5 and 6). Because of their generality, they can be mapped to many different interaction techniques, and in one form or another, they are often the primary means of communicating user intention in 3D UIs for VE applications.

4.4.1. Handheld 3D Mice

A common design approach for 3D mice is to place a motion tracker inside a structure that is fitted with different physical interface widgets. Actually, one of the first 3D mice to be developed used no housing at all. The "bat" (Ware and Jessome 1988), so named because it is a mouse that flies, was developed by Colin Ware in the late 1980s. It was simply a 6-DOF tracking device with three buttons attached to it. Such a device is rather easy to build with a few electrical components (provided you have the tracking device). A more sophisticated and elegant version of the bat is shown in Figure 4.16. This device houses a motion tracker in a structure that looks like a simple remote control. It is commonly used in conjunction with surround-screen displays for both navigation and selection of 3D objects.

The physical structure that houses the motion tracker is often a replication of an input device used in the real world. For example, the 3D mouse shown in Figure 4.17 is modeled after an Air Force pilot's flight stick. Some 3D mice have also been developed to look like their 2D counterparts. For example, the Fly Mouse (see Figure 4.6) looks similar to a conventional 2D mouse, but it uses acoustic tracking, has five buttons instead of two, and can also be used as a microphone for speech input.

The Cubic Mouse (shown in Figure 4.18), originally developed at Fraunhofer GMD, is a 3D mouse designed primarily as an interactive prop for handling 3D objects. It is ideally suited for examining volumetric data because of its ability to intuitively map to the volume's coordinates and

Figure 4.16 *The Wanda input device. (Photograph courtesy of Ascension Technology Corporation)*

Figure 4.17 *A joystick modeled after a flight stick used in 3D VR applications. (Built by VP Integrated Solutions as a large-scale visualization device)*

act as a physical proxy for manipulating it (Fröhlich and Plate 2000). The device consists of a box with three perpendicular rods passing through the center, an embedded tracker, and buttons for additional input. The Cubic Mouse does have a disadvantage in that the three orthogonal rods can get in the way when the user is holding the device in certain configurations.

Figure 4.18 *The Cubic Mouse. (Photograph courtesy of Fakespace Systems)*

Aside from the Fly Mouse (shown in Figure 4.6), the 3D mice we have shown thus far have all been tethered. Many current 3D mice, however, are completely wireless, often making use of 6-DOF optical tracking. For example, the Bug (Stefani et al. 2003) is an ergonomically designed wireless device with two buttons and a slider. It looks similar to a desktop mouse, but features three spherical markers (used by the optical tracking system to measure position and orientation) protruding from the device.

4.4.2. User-Worn 3D Mice

Another approach to the design of 3D mice is to have the user wear them instead of hold them. Assuming the device is light enough, having the device worn on the user's finger, for example, makes the device an extension of the hand. Figure 4.19 shows the Ring Mouse, an example of such a device. It is a small, two-button, ringlike device that uses ultrasonic tracking that generates only position information. One of the issues with this device is that it has a limited number of buttons because of its small form factor.

The FingerSleeve, shown in Figure 4.20, is a finger-worn 3D mouse that is similar to the Ring Mouse in that it is small and lightweight, but it adds more button functionality in the same physical space by using pop-through buttons (Zeleznik et al. 2002). Pop-through buttons have two clearly distinguished activation states corresponding to light and firm finger pressure.

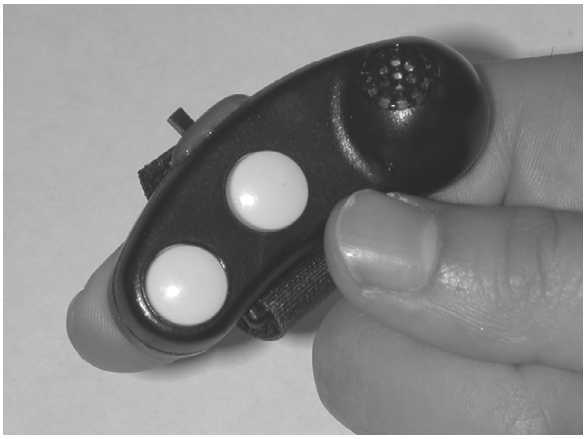

Figure 4.19 *The Ring Mouse input device uses acoustic tracking and is worn on a user's index finger. (Photograph courtesy of Joseph J. LaViola Jr.)*

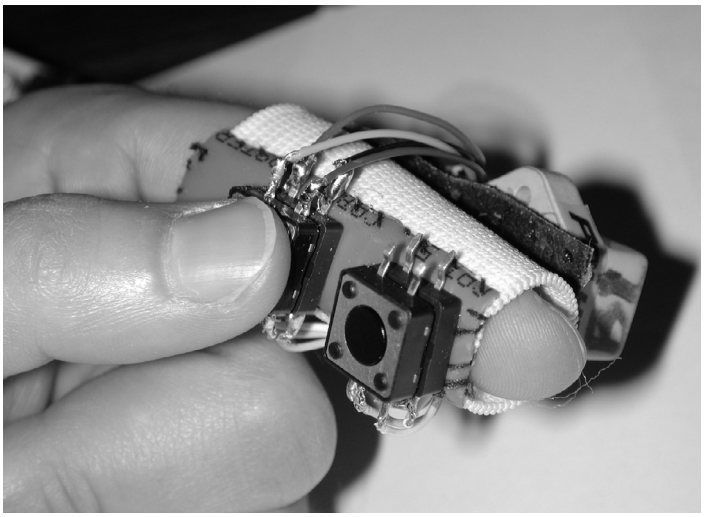

Figure 4.20 *A user pressing one of the multilevel buttons on the FingerSleeve. (Photograph reprinted from Zeleznik et al. 2002, © 2002 IEEE Press)*

The device can be worn on the index finger of either the left or right hand and is made of an elastic fabric and a small piece of flexible plastic that can be found at any arts and crafts store. The fabric is sewn into a sleeve with a varying diameter that fits snugly for most users. The plastic is sewn onto the front of the sleeve to provide a solid mount for pop-through buttons, and the buttons are glued into place a few millimeters apart on top of the plastic. A 6-DOF tracker is secured to the back of the sleeve using Velcro. See Zeleznik and colleagues (2002) for more details.

4.5. Special-Purpose Input Devices

Many other types of devices are used in 3D interfaces. These devices are often designed for specific applications or used in specific interfaces. In this section, we present some examples of these special-purpose devices.

ShapeTape (shown in Figure 4.21) is a flexible, ribbonlike tape of fiber-optic curvature sensors that comes in various lengths and sensor spacing. Because the sensors provide bend and twist information along the tape's length, it can be easily flexed and twisted in the hand, making it an ideal input device for creating, editing, and manipulating 3D curves (Grossman et al. 2003). In addition, the device can be used in system control (see Chapter 8) by recognizing different gestures made by the tape

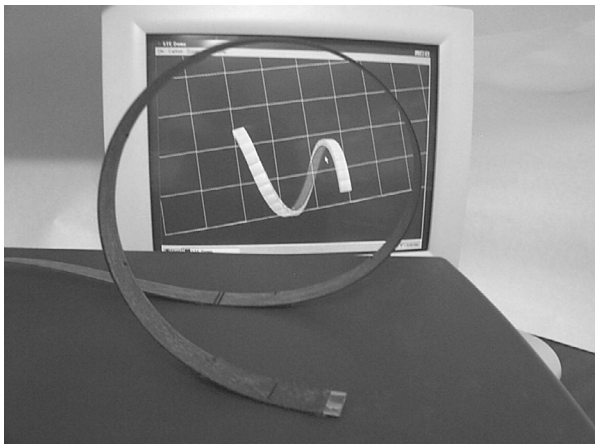

Figure 4.21 *ShapeTape used in manipulating 3D curves. (Photograph courtesy of Measurand, Inc., www.measurand.com)*

(e.g., quickly moving the endpoints of the tape together or apart). Note that in some cases, other input devices (the bat, for example) can be attached to the tape to increase functionality (Balakrishnan, Fitzmaurice et al. 1999).

In many surround-screen display configurations where the floor is actually a display surface, users must wear slippers when they enter the device to avoid making scuff marks and tracking in dirt. An interesting input device takes advantage of the need for slippers in these environments: the Interaction Slippers (see Figure 4.22).

Figure 4.22 *A user wearing Interaction Slippers. (Photograph reprinted from LaViola et al. 2001, © 2001 ACM Press)*

The Interaction Slippers (LaViola et al. 2001) embed a wireless trackball device (the Trackman) into a pair of common house slippers. The slippers use wireless radio technology to communicate to the host computer. The Trackman is inserted into a hand-made pouch on the right slipper and rewired. Two of the Trackman's three buttons are connected to a pair of conductive cloth patches on the instep of the right slipper. On the instep of the left slipper, two more conductive cloth patches are attached (as shown in Figure 4.22). Touching a cloth patch on the left slipper to a cloth patch on the right slipper completes the button press circuit. This design enables two gestures corresponding to heel and toe contacts respectively. The slippers were designed for interacting with the Step WIM navigation technique, in which a miniature version of the world is placed on the ground under the user's feet, allowing him to quickly travel to any place in the VE. LaViola and colleagues (2001) describes the slippers in more detail.

An example of an input device that was specifically developed for a particular 3D application is the CavePainting Table (see Figure 4.23) used in CavePainting (Keefe et al. 2001), a system for painting 3D scenes in a VE. The CavePainting Table uses a prop-based design that relies upon multiple cups of paint and a single tracked paintbrush. These paint cup props stay on a physical table that slides into the surround-screen device and also houses knobs and buttons used for various interaction tasks. In conjunction with the table, a real paintbrush is augmented with a single

Figure 4.23 *The CavePainting Table used in the CavePainting application. (Photograph reprinted from Keefe et al. 2000, © 2001 ACM; reprinted with permission)*

button that turns the "paint" on and off. The bristles of the brush are covered with conductive cloth, and users can dip the brush into the paint cups (which are lined with conductive cloth as well) to change brush strokes. A tracked bucket is used to throw paint around the virtual canvas.

In some cases, making a simple addition to an existing input device can create a powerful tool for interacting in 3D applications. For example, when interacting with 3D applications that utilize workbench-style displays, attaching a motion tracker to a piece of Plexiglas can create a useful tool for interacting in 2D and 3D. In addition, these devices can also have touch-sensitive screens (see Figure 4.24). Such a device allows the user to perform 2D interaction techniques, such as writing and selection of objects and commands from 2D palettes, as well as 3D interaction techniques, such as volumetric selection by sweeping the device through the virtual world (see Schmalstieg, Encarnação et al.1999, Coquillart and Wesche 1999, and Williams et al. 1999 for more details). This "pen-and-tablet" metaphor has been used extensively in 3D UIs and is discussed in detail in Chapter 10.

The last input device we discuss in this section is the Control Action Table (CAT), which was designed for use in surround-screen display environments (Hachet et al. 2003). This freestanding device (shown in Figure 4.25) looks like a circular tabletop. The CAT uses angular sensors to detect orientation information using three nested orientation axes. The device also has an isometric component; the tabletop is equipped with a potentiometer that detects forces in any 3D direction. Thus, the user can push or pull on the device for translational movement. Additionally, the CAT has a tablet for 2D interaction mounted on the tabletop, which

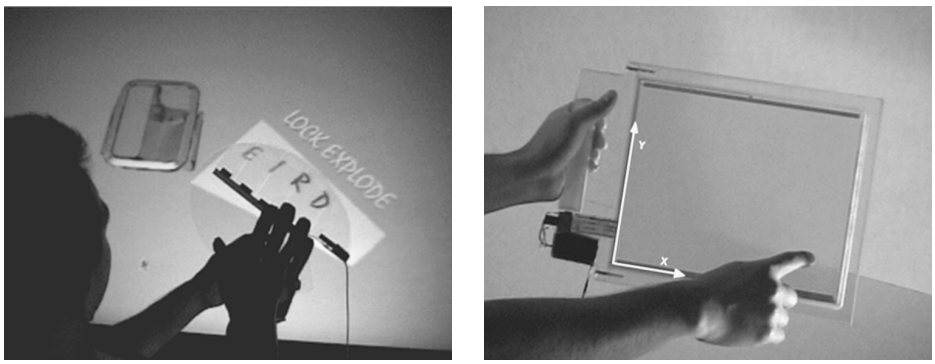

Figure 4.24 *Transparent palettes used for both 2D and 3D interaction (Williams et al. 1999; photographs courtesy of Fakespace Labs, Mountain View, California)*

Figure 4.25 *The CAT is designed for surround-screen display environments. It combines 6-DOF input with 2D tablet interaction. (Photograph courtesy of the Iparla Team [LaBRI–INRIA])*

makes it unique because it supports both 6-DOF and 2D input in the same device. Other advantages of the CAT include the ability to control each DOF individually and its *location persistence* (meaning that its physical state does not change when released). The CAT does have some inherent limitations because the nature of the nested orientation axes can make some orientations hard to specify, and in certain configurations (e.g., when the tabletop is vertical), translational movement can be difficult to perform as well.

4.6. Direct Human Input

A powerful approach to interacting with 3D applications is to obtain data directly from signals generated by the human body. With this approach, the user actually becomes the input device. For example, a user could

stand in front of a camera and perform different movements, which the computer would interpret as commands (Lucente et al. 1998). In this section, we specifically discuss speech, bioelectric, and brain computer input and how they can be used in 3D UIs.

4.6.1. Speech Input

Speech input provides a nice complement to other input devices. It is a natural way to combine different modes of input (e.g., multimodal interaction) to form a more cohesive and natural interface. In general, when functioning properly, speech input can be a valuable tool in 3D UIs, especially when both of the user's hands are occupied. Beyond choosing a good speech recognition engine, there are many other important issues to consider when using speech for a 3D interface (LaViola 1999a).

There are tradeoffs that must be made when dealing with speech input. One important issue is where the microphone is to be placed. Ideally, a wide-area microphone is used so that the user need not wear a headset. Placing such a microphone in the physical environment could be problematic, since it might pick up noise from other people or machines in the room. One of the big problems with using speech input is having the computer know when to and when not to listen to the user's voice. Often, a user is conversing with a collaborator with no intention of issuing voice commands, but the applications "thinks" the user is speaking to it. This misinterpretation can be very troublesome.

One of the best ways to avoid this problem is to use an implicit or invisible push-to-talk scheme. A traditional push-to-talk scheme lets the user tell the application when he or she is speaking to it, usually by pushing a button. In order to maintain the naturalness of the speech interface, we do not want to add to the user's cognitive load. The goal of implicit push-to-talk is to embed the "push" into existing interaction techniques so the user does not have the burden of remembering to signal the application that a voice command is about to be issued. As an example, consider a furniture layout application in which a user wants to place different pieces of furniture into a room or other architectural structure. The user wishes to put a table into a kitchen. To accomplish this task, the user must create the object and then place it in the room. The user shows where the table should be placed using a laser pointer and then says, "Give me a table, please." The act of picking up the laser pointer signals the application that the user is about to ask for an object. This action "piggybacks" the voice command onto the placement task, making the

push-to-talk part of the technique implicit. More information on speech input can be found in Chapters 8 and 9.

4.6.2. Bioelectric Input

NASA Ames Research Center has developed a bioelectric input device that reads muscle nerve signals emanating from the forearm (see Figure 4.26). These nerve signals are captured by a dry electrode array on the arm. The nerve signals are analyzed using pattern recognition software and then routed through a computer to issue relevant interface commands. Figure 4.19 shows a user controlling a virtual 757 aircraft (Jorgensen et al. 2000). This type of device could also be used to mimic a real keyboard in a VE.

4.6.3. Brain Input

The goal of brain–computer interfaces is to have a user directly input commands to the computer using signals generated by the brain (Millán 2003). A brain–computer interface can use a simple, noninvasive approach

Figure 4.26 *A bioelectric control device attached to a user's wrist. (Photograph courtesy of NASA Ames Research Center)*

by monitoring brainwave activity through electroencephalogram (EEG) signals. The user simply wears a headband or a cap with integrated electrodes. A future, more invasive approach would be to surgically implant microelectrodes in the motor cortex. Of course, this approach is still not practical for common use but might be appropriate for severely disabled people who cannot interact with a computer in any other way. Research has shown that a monkey with microelectrodes implanted in its motor cortex can move a mouse cursor to desired targets (Serruya et al. 2002). These types of interfaces are still in their infancy, but the underlying potential of such an interface is immense, not only for 3D interfaces but for all other types of computer interaction as well.

An example of a brain–computer interface device is shown in Figure 4.27. This device uses a brain–body actuated control technology that combines eye movement, facial muscle, and brainwave biopotentials to generate input signals. It has three sensors in a headband, and its interface unit amplifies and translates the brainwave, facial muscle, and eye-movement data into separate frequencies and transmits them to a PC serial port. Software processes and displays these frequencies and 10 continuous command signals, which can then be sent to applications for navigating through a VE, interacting with a computer game, or a variety of other tasks. Although this device represents only the beginning of brain–computer input devices, it shows the potential of this technology for taking computer interaction to a whole new level. More information on brain–computer interfaces can be found in Millán (2003).

Figure 4.27 *The user wears a headband that reads brainwave biopotentials to generate input. (Photograph courtesy of Brain Actuated Technologies, Inc.)*

4.7. Home-Brewed Input Devices

In some situations, commercial, off-the-shelf input devices may not be adequate for a given 3D application or task. For example, the 3D UI designer may want to better exploit a user's feet in navigating through a VE or to have discrete button functionality on a bend-sensing data glove. Such situations provide an opportunity for exploring the inhouse development of novel devices that are specifically suited to given interaction techniques. Creating home-brewed input devices often aids the 3D UI designer in developing new interaction techniques and improving upon existing ones to provide the user with more expressive power in specific 3D applications and with new methods of expression that existing input devices do not afford (LaViola et al. 2004).

The 3D UI researcher and practitioner can go a long way toward developing these augmented and novel devices using simple electronic components and household items. In fact, this home-brewed style of input device design and prototyping has produced many of the commercial input devices sold today. We have already seen devices such as the bat, the Cubic Mouse, the FingerSleeve, and the CAT, all of which were designed and built by researchers in small academic research labs. In this section, we briefly discuss some strategies for making custom input devices by first presenting some ideas on building the physical devices and then discussing methods for creating the interface between devices and the computer.

4.7.1. Strategies for Building Input Devices

There are a variety of strategies for constructing home-brewed input devices. One of the first things to consider is the device's intended functionality, because doing so helps to determine what types of physical device components will be required. For example, the device might need to sense forces, motion, or simply button presses. Based on the intended device functionality, the device developer can choose appropriate sensors, whether they be digital (output of 0 or 1) or analog (output of a range of values). These sensors can easily be found in electronics stores and over the Internet. Examples include pressure sensors, bend sensors, potentiometers, thermistors (for sensing temperature), photocells (for sensing light), simple switches, and many others. These sensors come in a variety of styles and configurations, and the appropriate choice is often based on trial and error. This trial-and-error approach is especially important with buttons, since buttons and switches come in many different shapes, sizes,

Figure 4.28 *Many different buttons and switches can be used to make homemade input devices. Top row, left to right: a lever switch, a mercury switch, and conductive cloth patches used to make flexible buttons. Bottom row: buttons of different sizes and shapes. (Photograph courtesy of Joseph J. LaViola Jr.)*

and force thresholds—the amount of force the user needs to activate the button or switch (see Figure 4.28).

In many cases, building the sensors is a feasible option, especially if they are switches or buttons. One powerful approach for building simple switches is to use conductive cloth. Conductive cloth is just fabric with conductive material sewn into it, and it has many advantages for building custom input devices. In fact, the input devices shown in figures 4.14, 4.15, 4.22, and 4.23 all use conductive cloth as part of their designs. Conductive cloth is inexpensive and a fairly robust material. Because it is cloth, it is flexible, so it can be used just about anywhere, in many different sizes and geometric configurations. Additionally, conductive cloth is easily sewn onto other fabrics so that devices can be constructed on clothing.

Another important consideration when making home-brewed input devices is how the sensors are housed in the physical device. Positioning sensors in the device housing is especially important when they are active components such as buttons, dials, and sliders, because the user must be able to interact with them comfortably to manipulate the device. For example, if a homemade 3D mouse is being constructed with several buttons, these buttons should be placed so that the user does not have to endure any undue strain in order to press any of the buttons at any given time. Sensor placement in homemade input devices is also affected by the geometry of the device itself.

One of the reasons many 3D UI designers do not build homemade devices is that they do not have the ability or equipment to construct the physical housing the sensors are placed in and on. Ideally, a milling machine, vacuform device (a device that heats plastic and stretches it over a mold), or 3D printer would be used to construct the device housing based on a model developed in 3D modeling software. However, these tools are not necessarily household items. One novel approach for constructing device housings is to use Lego bricks. The Lego Interface Toolkit (Ayers and Zeleznik 1996), a rapid prototyping system for creating physical interaction devices, uses this approach. It utilizes Lego bricks along with push buttons and rotational and linear sensors that support a variety of physical configurations. This component-based approach enables the input device developer to quickly snap together different device components. Another approach is to use modeling clay to create input device housings. The advantage of using modeling clay is that it can be molded into any shape the designer wants and can be quickly changed to try out different geometries. Once an appropriate design or designs are found, the clay can be oven fired and used as the device housing.

4.7.2. Connecting the Home-Brewed Input Device to the Computer

The other important part of constructing home-brewed input devices is choosing how to connect them to the computer. In the majority of cases, homemade input devices require some type of logic that the user needs to specify in order for the computer to understand the data the input device produces. The one exception is when existing devices are taken apart so that the sensors in them can be used in different physical configurations. An example of this is the Interaction Slippers shown in Figure 4.22. Because the device uses the rewired components of a wireless mouse, it can use the standard mouse port to transmit information to the computer, thus requiring no additional electronics.

There are two primary approaches for connecting a homemade input device to the computer so it can be used in 3D interfaces. The first approach is to use a microcontroller. A microcontroller is just a small computer that can interface with other electronic components through its pins. There are many different varieties to choose from depending on price, power, ease of programming, and so on. The designer can connect an input device to a microcontroller on a circuit board, which in turn

communicates to the computer through a serial or USB port. Typically, the designer first builds the electronics on a prototyping board (breadboard), which is an easy way to establish electrical connections between the device and the microcontroller without the need to solder. Using any of the many software packages (many of them are free and use Basic or C) for writing microcontroller code, the developer can write a program for controlling the input device and download it to the microcontroller. After the prototyping and testing stage, the microcontroller and any associated electronics can be attached to an appropriate circuit board. The homemade input device then has its own electronics unit for sending information from the device to the computer, and with appropriate software such as device drivers, to the 3D UI. Using microcontrollers does require some effort and has a slight learning curve, but the approach gives the input device developer a lot of freedom in choosing how the input device/computer interface is made. A nice introduction to using microcontrollers to build input devices can be found in Forman and Lawson (2003).

One way to still use microcontrollers but sidestep many of the issues involved in using them is to use Phidgets (Greenberg and Fitchett 2001). Phidgets are physical input device widgets that can be combined through a toolkit, which essentially encapsulates the implementation and construction details of connecting input devices to computers. In effect, they provide the ability to create homemade input devices without the need to program microcontrollers and design circuit boards. More details on how Phidgets are used can be found in Greenberg and Boyle (2002).

A second approach to connecting homemade input devices to a computer is to use the Musical Instrument Device Interface (MIDI). MIDI is a protocol that was developed to allow electrical musical instruments to communicate with computers (Rumsey 1994). The important characteristic of MIDI is that it is a protocol for communicating control information, such as if a button was pressed, how hard it was pressed, or how long it was pressed, which means it can be used for connecting input devices to computers. Figure 4.29 shows an example of a MIDI controller and some of the sensors used in developing input devices with it (Figure 4.20 shows an input device that uses this controller). Similar to Phidgets, using MIDI gives the input device developer the advantage of not having to deal with microcontroller programming and circuit design. However, in most cases, the developer still needs to write the device drivers to use the custom-built input devices in 3D applications.

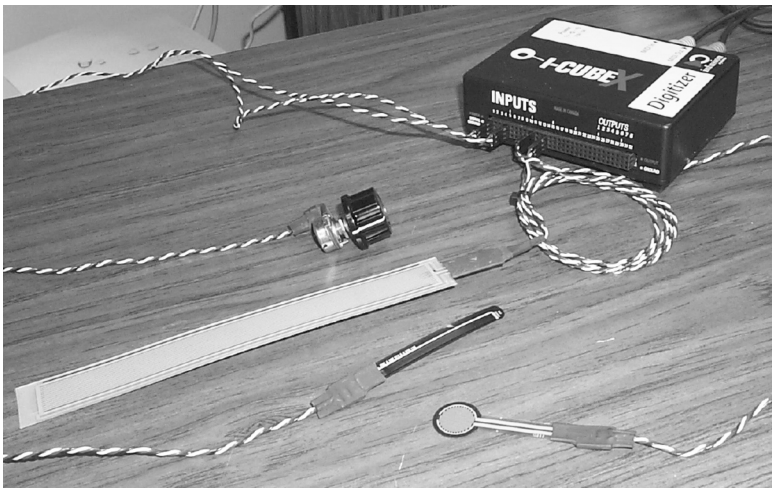

Figure 4.29 *A MIDI controller can be used as the connection between input devices and the computer. The MIDI-compatible sensors attached to the device are, from top to bottom, a turn sensor, a slide sensor, a bend sensor, and a small touch sensor. (Photograph courtesy of Joseph J. LaViola Jr.)*

4.8. Choosing Input Devices for 3D User Interfaces

A key issue in 3D interface design is to choose the appropriate input devices that are best suited to the needs of a particular application. The designer needs to examine the various tasks the 3D UI needs to support, find or develop the appropriate interaction techniques, and ensure that the chosen input devices are mapped to these techniques appropriately. In this section, we first examine some important factors to consider when choosing input devices. We then discuss two important tools, input device taxonomies and empirical evaluations, which can aid the designer in choosing input devices for 3D applications.

4.8.1. Important Considerations

Many factors must be considered when choosing an appropriate input device for a particular 3D UI. Device ergonomics, the number and type of input modes, the available technique to device-mapping strategies, and the types of tasks the user will be performing all play a role in choosing suitable input devices, making these choices a challenging task. The problem is amplified due to the variety of possible operations the user might perform within the context of a given 3D application. A particular

device might be perfect for one task in an application but completely inappropriate for another.

Device ergonomics is clearly an important consideration when choosing an appropriate input device for a 3D application. In general, we do not want to put undue strain on the user's body. Such strain can lead to repetitive stress injuries and make it difficult for the user to perform common tasks. Devices should be lightweight, require little training, and provide a significant transfer of information to the computer with minimal effort.

A particular device's input modes must be considered when choosing an input device for a 3D application. The types of input required for a given application helps to reduce the possible device choices. For example, a conventional 3D modeling system uses a keyboard and other 2D devices such as a mouse or a large tablet. However, these devices in an immersive 3D modeler are not appropriate, since they are difficult to use while standing and they do not provide the appropriate DOF and continuous events needed to track the user's head and hands. In contrast, a desktop 3D computer game does not necessarily require a complicated 6-DOF tracking device, since, in most cases, the keyboard and a mouse or a joystick will suffice. To take another example, a simple immersive architectural walkthrough requires the user to be head-tracked and have a way to navigate through the environment. In such an application, although a bend-sensing glove could be used to navigate (using some collection of gestures), it would probably not be appropriate given the complexity of the device. A simpler device such as a Wanda, shown in Figure 4.15, is much easier to use, since the application does not need all of the extra DOF that a bend-sensing glove gives the user.

As stated earlier, an input device can handle a variety of interaction techniques depending on the logical mapping of the technique to the device. The major issue is whether that mapping makes the device and the subsequent interaction techniques *usable.* Therefore, an important consideration when choosing an input device in a 3D application is how the given device will map to the variety of interaction techniques required to perform application tasks. It is in these mappings where tradeoffs are usually made, since very often a device will have a natural mapping to one or two of the interaction techniques in the application but relatively poor mappings to the others. For example, in the context of an immersive scientific visualization application, a 3D tracker may be attached to the user's hand and used for selection and manipulation of a tool for examining the dataset. The user can simply move the tool within the space of the

dataset to explore it, and this represents a natural mapping from device to IT. However, using the 3D tracker to input parameter values to change the rendering style has a less natural mapping from device to IT. If the scientific visualization application was on the desktop, the keyboard would provide a much more natural mapping for changing rendering styles but would make object selection and manipulation much more difficult.

This example makes the point that there is often a tradeoff when choosing an input device for a 3D application. In many cases, input devices have been designed for general use, which means that although they can be used for a variety of interaction techniques, they may not provide the best mapping for any one of them. Thus, several specialized devices may provide better usability than a single general-purpose device.

> It is often better to have a series of specialized devices that work well for a specific group of interaction techniques rather than one or two generic input devices for all of the techniques in a 3D application.

Examples of this type of approach can be found in the work of Hinckley and colleagues (1994); Forsberg, LaViola, and Zeleznik (1998); and Keefe and colleagues (2001).

4.8.2. Input Device Taxonomies

Input device taxonomies can be a useful tool for determining which input devices can be substituted for each other, and they can also help in making decisions about what devices to use for particular tasks. In addition, they are an important part of 3D UI design because they provide a mechanism for understanding and discussing the similarities and differences among input devices. In this section, we briefly review some of these input device taxonomies from a historical perspective to show the evolution of these tools. Additionally, we discuss how they can be used to help make decisions about choosing appropriate devices in 3D UIs.

One of the first input device taxonomies was developed by Foley and Wallace (1974). Their approach was to separate the input device from the interaction technique. They created a set of four virtual devices, which at the time covered most input devices. These virtual devices are the pick, locator, button, and valuator. A *pick* device is used to designate a user-defined object. A *locator* is used to determine position and/or orientation.

A *button* is used to designate a system-defined object. Finally, a valuator is used to input a single value within a set of numbers. Two additional virtual devices, stroke and string, were added to this set by Enderle, Kansay, and Pfaff (1984). A *stroke* is a sequence of points, and a *string* is a sequence of characters.

This virtual device taxonomy proves useful in many different situations. For example, the 3D UI developer can use this taxonomy as a tool for quickly reducing the number of possible input devices to choose from by simply examining which virtual devices fit the application best and selecting the devices that fit in those categories. If a 3D locator is required in an application, then we can automatically eliminate all of the physical devices that do not map to this virtual device. However, this taxonomy does have a fundamental flaw, because devices that appear to be equivalent in the taxonomy can be dramatically different both physically and practically. For example, a mouse and a trackball are very different devices, yet are both considered to be 2D locators and stroke devices.

As a result of these limitations, new taxonomies were developed to take other characteristics of the input devices into account. For example, Foley, Wallace, and Chan (1984) improved upon the virtual device taxonomy by mapping elementary interaction tasks (e.g., select, position, orient,) to the devices that perform those tasks. Based on task requirements, only a limited set of devices could be used for any particular task. However, because the taxonomy is task-based, an input device can appear for more than one task. Although this taxonomy can help to reduce the set of possible input devices even further than the virtual device taxonomy, it still has the problem that the structure of the device space and any pragmatic differences that distinguish input devices from one another are hidden.

To compensate for the pragmatic deficiencies of earlier taxonomies, Buxton (1983) developed a taxonomy that organizes continuous input devices into a 2D space, the dimensions of which are DOF and properties sensed (i.e., motion, position, pressure). Additionally, a subclassification is used for devices that have a mechanical intermediary between the hand and the sensing mechanism and those that are touch-sensitive. An example of how this taxonomy classifies input devices is shown in Figure 4.30. Two major problems with this taxonomy are that it was not developed to handle discrete input devices and, although it can determine if it is incorrect to substitute one input device for another, it cannot tell us why that substitution is inappropriate.

Property Sensed		Number of Dimensions: 1		2		3	
Position		Bend Sensor	Linear Slider	Tablet and Stylus	Isotonic Joystick	Trackers (Position & Orientation)	M
				Touch Tablet			T
Motion			Treadmill	Mouse	TrackBall		M
Pressure		Torque Sensor			Isometric Joystick	SpaceBall & SpaceMouse	T

Figure 4.30 *Buxton's input device taxonomy, based on pragmatic attributes of devices, categorizes devices based on DOF, properties sensed and whether the device uses an intermediary between the user and sensing mechanism or is touch-sensitive. (Buxton 1983)*

Mackinlay, Card, and Robertson (1990b) extend Buxton's taxonomy in two ways. First, they distinguish between absolute and relative quantities. Second, they separate translational and rotational movement. In addition, they distinguish between the different translational and rotational axes instead of using DOF. For example, a 6-DOF position-tracking device would be labeled a device for sensing position in *x*, *y*, and *z* Cartesian coordinates and orientation in *rX*, *rY*, and *rZ* rotational coordinates with an infinite number of possible values sensed for each axis (see Figure 4.31). Their taxonomy allows for the description of both discrete and continuous devices as well as simple devices that can be combined into complex controls. This taxonomy was one of the first that recognized that human performance issues are important to understanding how devices work for given tasks. It could be used for choosing input devices in a similar manner to previous taxonomies—as a tool to help narrow down device choices by examining device pragmatics and mappings to interaction tasks.

Although the taxonomies we have discussed thus far provide frameworks for characterizing and understanding different input devices, they are limited in that they can reduce the number of possible input device choices but not determine which specific device is preferable. Jacob and

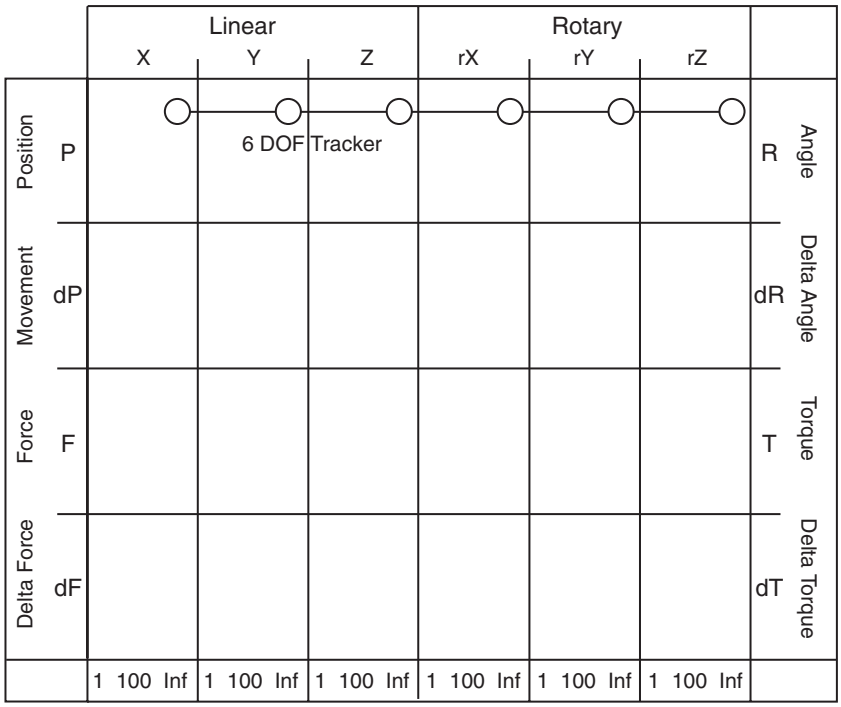

Figure 4.31 *Mackinlay, Card, and Robertson's (1990b) input device taxonomy categorizes a 6-DOF tracker. The circles indicate that the device senses one of the properties shown along the vertical axis. The placement of these circles within a column shows how many values are sensed.*

Sibert (1992) took these flaws into account by realizing that perceptual and cognitive issues were ignored. They developed a taxonomy (actually more of a theoretical model) to address these issues. Their approach is based on the idea that multidimensional spaces have different perceptual structures, and observers perceive multidimensional spaces based on these structures (Garner 1974). The taxonomy breaks these structures into two distinct components. Single composite perceptions are *integrally* related, while distinct perceptions are *separably* related. A multidimensional input device can be considered integral or separable based on whether it is a device that considers DOF together (a single composition) or that treats DOF individually (distinct compositions). The argument for this approach is that two 3D tasks may seem equivalent but have different perceptual spaces, implying that different input devices should be used. This model was designed not so much for characterizing devices as for

choosing when 3D input devices should be used instead of 2D devices for given tasks. It has been used in different input device evaluations, such as Hinckley, Tullio, and colleagues (1997) and Balakrishnan and colleagues (1997).

4.8.3. Empirical Evaluations

In general, the taxonomies we discussed in the last section are useful for narrowing down the choice of input device for a particular task or 3D UI. However, in order to get concrete information about which devices are appropriate for given tasks, empirical studies are often required. In contrast to the lack of empirical work done on choosing appropriate output devices for 3D applications (see Chapter 3), there has been a good amount of research evaluating input devices for interacting in 3D. Performing empirical analyses of input devices is somewhat easier than performing comparisons of output devices, because it is easier to obtain quantitative measurements about device performance. Characteristics such as speed, accuracy, and ease of learning are often used to measure how a device will perform in a certain task. In addition, tools such as Fitts's law (Fitts 1954, MacKenzie 1992), a formal relationship between speed and accuracy in aimed movements, and the steering law (Accot and Zhai, 1997), a formalization that describes the relationship between steering (i.e., drawing, navigation) time and the integral of the inverse of path width, have been developed based on extensive empirical evaluation.

Studies have been conducted to determine the effectiveness of 3D input devices compared to traditional desktop devices such as the mouse. For example, Hinckley, Tullio, and colleagues (1997) compared the mouse with a 6-DOF tracking device for performing 3D object rotation tasks. Their results showed that the tracking device performed 36% faster than the 2D mouse without any loss of accuracy. In another study, Ware and Jessome (1988) compared the mouse and the bat (see section 4.4) for manipulating 3D objects. These results indicated that 3D object manipulation was easier to perform with the bat than with the mouse. Although these are only two studies, they do suggest that 3D input devices with 3 DOF or more are better than a mouse for handling freeform 3D object manipulation.

Zhai conducted a number of interesting studies specifically on 3D input devices (Zhai 1998) and came up with a number of guidelines for choosing appropriate 3D input devices. For example, in Zhai and Milgram

(1998), 6-DOF flying mice were shown to be easy to learn and to outperform many other 3D input devices in terms of speed. However, flying mice, such as the ones described in section 4.4, do have disadvantages in that they cause fatigue easily and lack persistence in position when released. In contrast, desktop 6-DOF devices such as the SpaceBall 5000 and SpaceMouse Plus (see Figure 4.4) were shown to reduce user fatigue, increase device persistence, and provide smoother pointer movement, but with slower performance. This is a specific instance of the classic speed-accuracy tradeoff.

> When speed and a short learning curve is of primary concern (e.g., for video games or location-based entertainment), free moving 6-DOF devices are most suitable. When comfort, trajectory control, and coordination are most important (e.g., for 3D modeling or teleoperation), a desktop-based 6-DOF input device should be used.

Other interesting empirical studies of 3D input devices can be found in Zhai and Woltjer (2003); Balakrishnan and colleagues (1997); Kessler, Hodges, and Walker (1995); and Zhai (1995).

Empirical work for determining appropriate input devices for specific tasks in 3D UIs can be difficult due to the many variables that are involved, but it continues to be an important area of research as 3D UIs continue to evolve. As new devices are developed, scientific studies must be conducted to determine how these devices can be used most effectively.

Recommended Reading

For a more detailed discussion on motion-tracking technology, we recommend the following:

Foxlin, E. (2002). Motion Tracking Requirements and Technologies. *Handbook of Virtual Environments: Design, Implementation, and Applications.* K. Stanney (Ed.), Lawrence Erlbaum Associates, 163–210.

Welch, G., and E. Foxlin (2002). Motion Tracking: No Silver Bullet, but a Respectable Arsenal. *IEEE Computer Graphics & Applications,* special issue on "Tracking" 22(6): 24–38.

Allen, D., G. Bishop, and G. Welch (2001). Tracking: Beyond 15 Minutes of Thought. *SIGGRAPH Course #11.*

A comprehensive discussion on computer vision techniques used in optical tracking can be found in the following:

Forsyth, D., and J. Ponce (2002). *Computer Vision: A Modern Approach,* Prentice Hall.

A thorough discussion on eye tracking can be found in these texts:

Duchowski, A. T. (2003). *Eye Tracking Methodology: Theory and Practice,* Springer-Verlag.

Wilder, J., G. Hung, M. Tremaine, and M. Kaur (2002). Eye Tracking in Virtual Environments. In *Handbook of Virtual Environments.* K. Stanney (ed.), Lawrence Erlbaum Associates, 211–222.

Jacob, R. (1995). Eye Tracking in Advanced Interface Design. *Virtual Environments and Advanced Interface Design.* W. Barfield and T. Furness, (Eds.), Oxford University Press, 258–288.

A nice survey on data gloves and glove-based input can be found in the following:

LaViola, J. (1999a). *Whole-Hand and Speech Input in Virtual Environments.* Master's Thesis, Department of Computer Science, Brown University.

Details on two important laws, Fitts's law and the law of steering, used in evaluating input devices, can be found in the following:

Fitts, P. M. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology* 47: 381–391.

MacKenzie, I. S. (1992). Fitts' Law as a Research and Design Tool in Human Computer Interaction. *Human Computer Interaction* 7: 91–139.

Zhai, S., and R. Woltjer. (2003). Human Movement Performance in Relation to Path Constraint: The Law of Steering in Locomotion. *Proceedings of IEEE Virtual Reality 2003,* IEEE Press,149–158.

Accot, J., and S. Zhai (1997). Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. *Proceedings of the 1997 ACM Conference on Human Factors in Computing Systems (CHI '97),* ACM Press, 295–302.

Finally, an important starting point for anyone interested in 6-DOF device evaluation is the following:

Zhai, S. (1995). *Human Performance in Six Degree of Freedom Input Control.* PhD Dissertation, Dept. of Computer Science, University of Toronto.