multidisciplinary

SECOND EDITION

user-centered design

evaluation and observation

design in a work context

development tools

READINGS IN

GUIs

Human-Computer Interaction:

touch and gesture

Toward the Year 2000

speech and language

human information processing

fit to human capabilities

groupware

WRITTEN AND EDITED BY

intelligent agents

Ronald M. Baecker
Jonathan Grudin
William A. S. Buxton
Saul Greenberg

hypertext

cyberspace

Chapter 7

Touch, Gesture, and Marking

This chapter investigates input to computer systems. Although there has been much progress since the pioneering Sketchpad system of the early 1960s (Sutherland, 1983 video), this is still an area of human-computer interaction in which there is great potential for further improvement.

Our discussion will focus on what we call *haptic* input. This term haptic comes from a Greek word having to do with *contact*. Hence, haptic input is input that involves physical contact between the computer and the user. This contact may be via the hands using a mouse, feet using a pedal, or even the tongue using a special joystick.

There are also nonhaptic means of input. The main one, speech, is covered separately in Chapter 8. Eye-tracking is another possibility. Since it is still more of a future technology than something that can be used today in mainstream applications, we shall merely refer the reader to Hutchinson et al. (1989), Jacob (1991), Koons, Sparrell, and Thorisson (1993), and Thorisson, Koons, and Bolt (1992 video).

We can't discuss haptic input without at least also mentioning output. *Every* haptic input device also provides output through the tactile or kinesthetic feedback it gives to users. The quality and appropriateness of this "feel" may be very important in determining a device's effectiveness and acceptance. Some devices, such as certain joysticks, actually provide force feedback (Iwata, 1990). Others, such as devices producing output in Braille, even use the haptic channel solely for output.

What we are *not* going to do in this chapter is provide a complete catalogue of input devices and how they work. This information is available in sources such as Chapanis and Kinkade (1972), Seibel (1972), Sherr (1988), Greenstein and Arnaut (1988), and Potasnak (1988). A condensed presentation can be found in Chapter 8 of Foley et al. (1990). The focus of our discussion will be on the human use rather than on the mechanics of the technology. We shall begin by looking at sample tasks by which devices can be compared. We shall also present a taxonomy of input that helps differentiate devices along dimensions meaningful to the designer.

RELATING TASK AND TECHNOLOGY

Each input device has its own strengths and weaknesses, just as each application has its own unique demands (Gaver, 1991). With the wide range of available input devices, a problem confronting the designer is to obtain a match between application and input technology. The designer must recognize the relevant dimensions along which an application's demands can be characterized, and must know how each technology being considered performs along those dimensions. These topics are addressed below and in Buxton (1986a).

One way to try and understand these issues is to experiment with a diverse set of representative tasks. Each task has its own idiosyncratic demands. We can determine which properties are relevant to a particular application, and then test the effectiveness of various technologies in their performance. This allows us to match technology to the application. Furthermore, the set of representative tasks provides a reminder of what dimensions should be considered in the selection process.

We shall now describe a basic set of generic tasks. Like any such list, this one is not complete. It was chosen to reflect the types of 2D tasks typically found in graphical user interfaces, such as those illustrated in the excellent collection of widgets compiled in Myers (1990 video). Text entry and 3D input are not emphasized. We leave it as an exercise for the reader to expand the list.

Pursuit Tracking

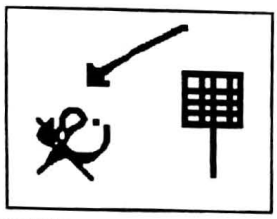
In this test, a target (a fly) moves over the screen under computer control (Figure 7.1). The operator uses the control device to track the fly's motion. Feedback about the operator's performance is given by a tracking symbol in the form of a fly swatter. The idea is to see how many times the fly can be killed by positioning the swatter over the fly and pushing a button device.

The main statistic in this test is how many times the fly can be swatted in a given time interval. A number of parameters should be variable in order to develop an understanding of their influence on task performance. One of these is the speed that the target moves. Another is the *control:display (C:D) ratio*. The C:D ratio is the ratio between the distance the controller must be moved in order to

cause the tracker to move a given distance on the display. For example, if the C:D ratio is 2:1, two centimeters of motion by the controller result in only one centimeter of motion by the tracker.

FIGURE 7.1

Pursuit tracking. A moving target (a fly) moves over the screen. The tracking symbol is a fly swatter. When the swatter is over the fly, the user pushes a button device to swat the fly.



A high C:D ratio is useful for fine cursor positioning, and for users who do not have well developed motor coordination (such as the aged or physically disabled). A low C:D ratio permits one to traverse the display with relatively little motion of the controller.

Notice that with devices that use the same surface for both control and display, such as touch screens and light pens, the C:D ratio is almost always 1:1. Such devices therefore have a directness not shared by most other technologies (Nelson, 1983 video; Minsky, 1985; Pickering, 1986; Potter, Shneiderman, and Weldon, 1988; Potter, Berman, and Shneiderman, 1989; Plaisant and Shneiderman, 1991 video; Sears and Shneiderman, 1991; Shneiderman, 1991; Plaisant and Wallace, 1992 video), which makes them particularly appropriate for systems to be used by anyone who walks up to them, such as information and museum kiosks.

With appropriate technology, the C:D ratio may be changed. Sometimes this is done by the user, and in other cases it is changed automatically. The C:D ratio also need not be linear. On the Macintosh computer, for example, the C:D ratio varies, based on the speed that the controller (the mouse) is moved. In effect, the C:D ratio is governed by an "automatic transmission."

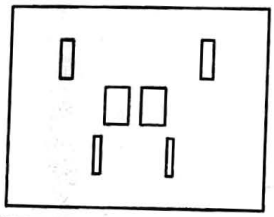
One other parameter to consider in this test is the button push that is required to swat the fly. For example, can this be activated with the same hand that is providing the spatial control? This may be difficult if a trackball or a touch tablet (Buxton, Hill, and Rowley, 1985) is used.

Target Acquisition

In this task, the user must select each of a number of rectangles displayed on the screen (Figure 7.2). A rectangle is selected by positioning the tracking symbol over it and signaling the system with a button event. Rectangles should be selected left to right, top to bottom.

FIGURE 7.2

Target acquisition. Select each rectangle in turn, left to right, top to bottom. Notice how selection time is related to target size.



The most interesting statistic is how long it takes to select the full set of targets. One should also examine how target size affects the speed of target acquisition. The smaller the target, the longer it will take to acquire, because of the fine motor control required. This is generalized as *Fitts' Law* (Fitts and Peterson, 1964), and was first used to study HCI in Card, English, and Burr (1978) (see also English, Engelbart, and Berman, 1967).

Fitts's Law states that the movement time, *MT*, to acquire a target with a continuous linear controller is a function of the distance over the target size. More precisely, the time *MT* to move the hand to a target of width *W*, which lies at distance *D* away is

$$MT = a + b \log_2 (D/W + 1)$$

where *a* is a constant, and (according to Card, Moran, and Newell, 1983, p. 241) $b = 100[70\text{–}120]$ msec/bit.

The most current and thorough discussion of Fitts' Law is MacKenzie (1992a). The reading by MacKenzie (1992b) summarizes important points for the practitioner and presents examples of how this law can be applied.

As in the pursuit-tracking task, issues such as C:D ratio and what button device is used will affect performance. There are a number of variations on this basic task. Each brings to light important aspects of the input technology:

- *Homing time:* If an application involves frequently moving between a text entry device (usually a QWERTY keyboard) and a pointing device, this movement affects user performance. This is the homing time discussed in the keystroke model (Card, Moran, and Newell, 1980). To get a feeling for the effect of homing time, have the user push a key on the keyboard (the spacebar, for example), after each square is selected. Also, using the same tablet, what is the difference in performing this task between using a puck and a stylus?
- *Number of dimensions:* Fitts' Law deals with movement in one dimension. Some studies have used it to predict and analyze performance in higher dimensional tasks. The two main questions are, (a) What is the effective target width in two or more dimensions? and (b) What is the effect of approach angle? The former question can be understood by considering selecting a word, which is graphically short in the vertical dimension but wide horizontally. Should the width or height be used? The second question has to do with whether or not we can move more effectively horizontally or vertically. These are points discussed in MacKenzie (1992b), and more fully in MacKenzie and Buxton (1992).
- *Dragging and rubber-band lines:* In this variation, an object is dragged (or a rubber-band line is stretched) from square to square. As shown by MacKenzie, Sellen, and Buxton (1991), dragging between targets is also a Fitts' Law task, and the act of maintaining the dragging state (by holding down a button on the mouse, for example) can negatively affect the acquisition of the target. In particular, different devices are affected to different degrees by state maintenance. Hence, if dragging is an important part of the interaction style, devices should also be evaluated in this mode.
- *Left hand vs right hand:* It is interesting to see how devices perform in Fitts' Law tasks when carried out using the left or the right hand. Kabbash, MacKenzie, and Buxton (1993)

found that the degradation in moving from the dominant to nondominant hand varied across devices. The lesson is to be sure that devices are evaluated in the context in which they are to be used.

Free-Hand Inking

Attempting to input a facsimile of your handwritten signature places yet another set of demands on the input technology. To get a feeling for the degree to which various devices lend themselves to this type of task, present the user with a screen ruled with lines of decreasing spacing. Have users sign their names once in each space (Figure 7.3). Use a simple subjective evaluation of the quality of signatures as a means of comparing devices.

FIGURE 7.3

Free-hand inking. A device's ability to capture a good facsimile of your signature is a good measure of its effectiveness for inking and drawing tasks. Comparisons should be made for different sizes.

The attributes of this handwriting exercise are relevant to several other common tasks, such as those seen in drawing programs, marking interfaces, and systems that utilize character recognition.

Tracing and Digitizing

In applications such as cartography, computer-aided design, and the graphic arts, it is important to be able to trace material previously drawn on paper, or digitize points from a map. *Relative* devices such as mice and trackballs are almost useless in this regard. *Absolute* devices such as tablets vary widely in how well they can perform such a task. Demands on resolution and linearity vary greatly across applications. In cartography, for example, the accuracy of digitization required is often far beyond what is needed to digitize a sketch.

Constrained Motion

In some applications it is important to be able to move the tracker rapidly along a straight-line path. One example is in using the scrollbar mechanism of some systems. Another is where you want to use the motion of a mouse in one dimension, say Y, to control one parameter, without changing the value of another parameter being controlled by motion in the X dimension. Different devices vary in the ease with which this can be done. X/Y thumbwheels, for example, would outperform a mouse in this task if the motion were along the primary axes.

In the example task of Figure 7.4, the tracking symbol is a ball that is dragged along a linear path defined by two parallel lines. The object is to move along the path without crossing the parallel lines. How is speed affected by the input device used and the path width? Are similar results obtained if the path is vertical or diagonal?

A variation of the previous example is to see how well the user can specify a circular motion within a constrained region (Figure 7.5). This type of control is useful in manipulating 3D objects, for example, and is described in Evans, Tanner, and Wein (1981). As in the previous example, different results will be obtained from different devices. Results will also vary according to C:D ratio and the width of the path.

FIGURE 7.4

Constrained linear motion. How quickly can the ball be moved along the straight path without crossing the lines?

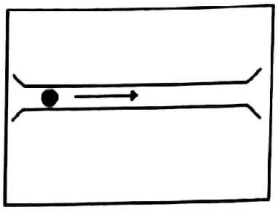
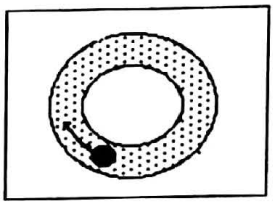


FIGURE 7.5

Constrained circular motion. How fast can the tracking symbol be moved around the circular shaded path without crossing the borders?



Three-Dimensional Input and Interaction

All of the tasks discussed thus far involve 2D control. As the power of graphics displays increases, 3D interfaces are becoming increasingly important to the user interface designer. Issues that used to concern only computer graphics specialists are now relevant to designers of general applications.

One consequence of this is that designers need to devise the 3D equivalents of the representative 2D tasks discussed in the previous section (see, for example, Myers and Zeller, 1990 video). Another is that the designer must decide what devices to use in supporting applications of higher dimensionality. At one extreme, does one need special devices such as those described in Brooks et al. (1983 video) and Brooks et al. (1990), or the instrumented gloves of virtual reality systems (Zimmerman et al., 1987; Zacharey, 1987 video; Weiner and Ganapathy, 1990 video)? Or can one support 3D interaction using the same, relatively nonintrusive devices found in conventional 2D interfaces?

A number of important studies show that the latter is often possible. One of the more compelling examples is the Information Visualizer described in Mackinlay, Card, and Robertson (1990), Mackinlay, Robertson, and Card (1990), and Mackinlay, Card, and Robertson (1990 video) and discussed more fully in Chapter 6.

Chen, Mountford, and Sellen (1988) is a good introduction to the problem of controlling 3D worlds with 2D devices. It introduces a novel new technique, the *virtual 3D trackball*. It also contains an excellent comparative evaluation of a number of techniques, many of them deriving from earlier work by Evans, Tanner, and Wein (1981; see also Evans, Tanner, and Wein, 1981 video; and Shoemake, 1992, for a variation of the virtual 3D trackball.)

Other publications dealing with novel 3D input devices, interaction techniques, and experimental studies include Schmandt (1983), Ware and Jessome (1988), Ware and Osborne (1990), Hill and Hollan (1992 video), MacKenzie and Ware (1993), and Zhai, Buxton, and Milgram (1994).

Finally, the reader should compare two different approaches to a 3D drawing package. Gargoyle 3D (Bier and Pier, 1987 video; Bier, 1989 video; Bier, 1990) and 3-Draw (Sachs, Stoop, and Roberts, 1989, 1990 video) are similar in that they use *constraints* and "snapping" techniques, but they differ greatly in how input is handled.

A TAXONOMY OF INPUT DEVICES

The examples of the previous section highlight how different devices lend themselves to different tasks. In this section, we develop a categorization of input devices that is based on the properties that cause these differences. Our approach is shown in Figure 7.6.

The table uses a hierarchy of criteria to organize the devices. It is limited, considering only continuous, manually operated devices. Hence, the first two (implicit) organizational criteria are

- continuous versus discrete
- agent of control (hand, foot, voice, etc.)

The table is divided into a matrix whose primary partitioning into rows and columns delimit

- what is being sensed (position, motion, or pressure)
- the number of dimensions being sensed (1, 2, or 3)

These primary partitions are delimited by solid lines. For example, both the rotary and sliding potentiometers fall into the box associated with one-dimensional position-sensitive devices (top left-hand corner). These primary rows and columns are then further subdivided by dotted lines into secondary regions that group

- devices that are operated using similar motor skills (subcolumns)
- devices that are operated by touch versus those that require a mechanical intermediary between the hand and the sensing mechanism (subrows)

Grouping by motor action can be seen in examining the two-dimensional devices. Since they are in the same subcolumn, Figure 7.6 implies that tablets and mice utilize similar types of hand control and that this motor action is different from that used by joysticks and trackballs, which appear in a different subcolumn.

The use of different subrows to differentiate between devices that are and are not touch-activated can be seen in comparing the light pen and the touch screen. While each utilizes similar motor control, the light pen requires the use of a stylus.

FIGURE 7.6

Taxonomy of input devices. Continuous manual input devices are categorized. The first-order categorization is property sensed (rows) and number of dimensions (columns). Subrows distinguish between devices that have a mechanical intermediary (such as a stylus) between the hand and the sensing mechanism (indicated by "M"), and those which are touch sensitive (indicated by "T"). Subcolumns distinguish devices that use comparable motor control for their operation (Buxton, 1983).

Property Sensed		Number of Dimensions: 1		2				3	
		rotary	linear	puck	stylus finger horiz.	stylus finger vertical	small fixed location	small fixed with twist	
Position		Rotary Pot	Sliding Pot	Tablet & Puck	Tablet & Stylus	Light Pen	Isotonic Joystick	3D Joystick	M
					Touch Tablet	Touch Screen			T
Motion		Continuous Rotary Pot	Treadmill	Mouse			Sprung Joystick Trackball	3D Trackball	M
			Ferinstat				X/Y Pad		T
Pressure		Torque Sensor					Isometric Joystick		T

Thus we see that the table models the structure that exists in the domain of input devices. It helps in finding appropriate equivalencies, which is useful in dealing with issues of device independence (to be discussed below). It also helps us relate devices to one another. For example, a tablet is to a mouse what a joystick is to a trackball.

Furthermore, if the taxonomy can suggest new transducers in a manner analogous to the way the periodic table of Mendelev has predicted new elements, then we can have even more confidence in it. We make this claim and cite the "torque sensing" one-dimensional pressure-sensitive transducer as an example. To our knowledge, no such device exists commercially. Nevertheless it is a potentially useful device, an approximation of which has been demonstrated by Herot and Weinzapfel (1978).

Three novel input devices demonstrated on video are the bicycle as workstation (Roberts, 1989 video), the pointing stick (Rutledge and Selker, 1990 video), and the cue ball (Theil, 1991 video).

Generality and Extensibility

Choosing the input technologies to be used with a workstation often involves a trade-off between two conflicting demands. Every task has specialized needs that can be best addressed by a specialized technology, yet each workstation is used for multiple tasks. Supplying the optimum device for each task is generally impossible, so a trade-off must be made.

Devices must be chosen to give the best coverage of the demands of the range of tasks. An important criterion in comparing devices, therefore, is how broad their coverage is. Stated differently, how many squares in Figure 7.6 can a particular device be used to fill? For example, graphics tablets can emulate many other transducers (Evans, Tanner, and Wein, 1981). The tablet is what could be called an *extensible* device. This property of extensibility is an important but seldom considered criterion that should be used in device selection.

Relative versus Absolute Controllers

Another important characteristic of input devices is whether they sense absolute or relative values. This has a very strong effect on the nature of the dialogues that the system can support with any degree of fluency. As we have seen, a mouse cannot be used to digitize map coordinates or trace a drawing because it does not sense absolute position. An example taken from process control, the *nulling problem*, occurs when absolute transducers are used in designs where one controller must be used for different tasks at different times (Buxton, 1986a).

What Our Taxonomy Doesn't Show

Perhaps the main weakness of the taxonomy presented above is that it considers only the continuous aspect of devices. As the sample tasks discussed earlier in this chapter illustrated, other factors, such as the integration of button devices with continuous controllers, has a strong impact on a device's performance. An example is the case of trying to "pick up" and drag an object with a mouse (where the button is integrated) compared to performing the same transaction using a trackball (where it is difficult to hold down the button, which is not integrated, with the same hand that is controlling the dragging motion).

An approach to capturing this aspect of devices is found in Buxton (1990a). A three-state model is developed that can be used to characterize both input *devices* and *tasks*. By providing a common vocabulary to describe both, a means of arriving at an appropriate match between the two is provided.

The reader is also referred to Foley, Wallace, and Chan (1984), an important early approach to characterizing input; Card, Mackinlay, and Robertson (1990, 1991), an excellent taxonomy of input devices that extends the model developed above; and Lipscomb and Pique (1993), a complementary means of categorizing input devices.

CHUNKING AND PHRASING

Much of the rest of this chapter deals with alternative ways of articulating commands to the computer. The reading by Buxton (1986b) is intended to lay a theoretical foundation for this. The main thesis of the paper is that human-machine dialogues can benefit by appropriate phrasing similar to that used in written and spoken language, and in music.

Phrasing not only groups together things that are associated in meaning or purpose, but also makes clear points of *closure*, that is, points at which one can be interrupted, or take a break. Most human-machine dialogues are *compound*, for example, selecting and positioning, positioning and scaling, navigating and selecting (Buxton, 1982; Buxton, 1984 video). The structure that emerges from appropriate phrasing can accelerate the process whereby novice computer users "chunk" together concepts, thereby building cognitive skill. Relevant to this issue is Mantei (1990 video), a delightful and thought-provoking collection of short clips documenting a variety of mouse use behaviors.

The reading discusses the nature of skill acquisition and the use of phrasing in its acquisition. In so doing, it lays the foundation for how some of the literature on cognitive modeling can be extended to apply to the *pragmatic* and *device* levels of the interface. Finally, it prepares the reader for the sections that follow—those that deal with marking, gesture, and two-handed input.

Modes and Mode Errors

As originally defined by Norman (1981), a *mode error* is the misclassification of a situation resulting in actions that are inappropriate for the true situation. Whenever a particular action has different consequences depending upon the state of the system, mode errors may occur. The classic example of this is in text editors with command-line interfaces. Here, for example, typing the word "add" may be interpreted as a *command*, indicating that you are about to add text, or as just another word that you want to enter into the document.

Reducing mode errors is one of the main attractions of direct manipulation interfaces (Tesler, 1981). Yet mode errors still occur. The best way to prevent them is to provide continuous and meaningful feedback to the user. For example, pressure and movement feedback has been shown to be effective in reducing mode errors (Sellen, Kurtenbach, and Buxton, 1992).

MARKING

There is increasing interest in a style of interaction that has been variously called "paperlike," "pencentric," "pen based," "character recognition," or "gesture driven." Yet many of these are not like paper, and many do not use a pen. What all have in common is that the user's input is in the form of a stream of x,y coordinates

that could be called *digital ink*. Hence, we will refer to these as *marking* interfaces, applications, or systems.

One of the main claims made for such systems is that they are more natural and easier to use. In many ways, we agree. Our discussion on chunking and phrasing emphasizes their potential. However, it is important to realize that *knowing how to use a pen is no more a ticket to mastering marking systems than knowing how to type is a ticket to mastering Unix or MS-DOS*. Most marking systems are extremely difficult to learn and are very prone to *mode errors*. This is true even of the best of the currently available systems, such as Carr (1991) and Carr and Shafer (1991). A good discussion of problems with marking interfaces is Briggs et al. (1993).

Who Does the Recognition?

Recognition lies at the heart of marking systems, whether they deal with block characters, cursive script, proofreader's symbols, or other annotations. Computer recognition is far from perfect, even when user's have training and are careful. This is not likely to change in the near future. In many ways, computer recognition is a "black hole" that has diverted energy and attention away from other aspects of marking-based systems that in the long run may be far more important than recognition.

This is because there are many benefits of electronic documents that do not require the computer to recognize their contents—particularly when marking-based systems are used in computer-mediated human-human interaction (see Chapter 11), rather than just in human-computer interaction.

We should therefore pay more attention to applications where recognition of the marks is done by a human rather than by a computer. As with notebook applications, this may be where the "recognizer" is also the author, or annotation systems, where the marked-up document is read by someone else. The markings might be recognized as they are written, as with electronic whiteboards (Elrod et al., 1992), or at a later time, as in fax or e-mail type applications.

The Freestyle system discussed in Case C (Levine and Ehrlich, 1991, included as a reading; Wang Laboratories, 1989 video) is a good example of how computer recognition can be sidestepped. Electronic mail and voice mail are integrated with markings to provide a creative system for annotating electronic documents.

While both Hardock, Kurtenbach, and Buxton (1993) and the reading by Goldberg and Goodisman (1991) show how Freestyle-like systems can be augmented with recognition, we want to emphasize that the application does not depend on it.

What Is Recognized?

If the computer does do mark recognition, this typically means character or script recognition. An early system is shown in Ward (1985 video). Good discussions of this topic can be found in Pittman (1991), Plamondon, Suen, and Simner (1989), and Suen (1990). The reading by Goldberg and Goodisman (1991) gives a brief introduction to character recognition, and a detailed analysis of how to handle recognition errors.

Recognizing characters and script is typically expensive both in terms of computational effort and the user's investment in training the system. Powerful applications can result when the recognition focuses on higher-level marks such as proofreader symbols. Such marks are frequently easier to recognize and can be user-independent. This is illustrated in the Tivoli electronic whiteboard application presented in the reading by Pedersen et al. (1993). Apte, Vo, and Kimura (1993) show how recognition can be applied to geometric shapes.

User-definable graphical marks have also been demonstrated by some researchers (Rubine, 1991; Wolf, Rhyne and Ellozy, 1989, 1989 video).

Self-Revelation and Marking Menus

A "paperlike" interface has the potential to be almost indistinguishable from Unix or MS-DOS. Just consider the similarity between a blank piece of paper and a screen that is blank other than for a "%" or "A>" prompt.

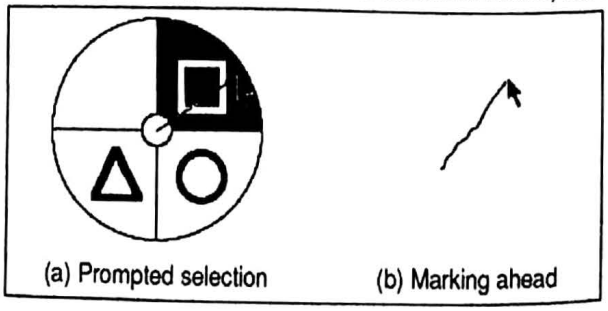
Without some help, the user has no means of knowing what state the system is in, or what options are available. So marking-based systems often assume a *form-filling* style (see the introduction to Part III), or a style similar to classical GUIs. Yet such interface styles may not be appropriate. Consider, for example, applications for very small displays, such as personal digital assistants (PDAs), where screen real estate is not available.

Recently, another design option has been developed, that of *marking menus* (Kurtenbach and Buxton, 1991a, 1991b, 1993, 1994). Marking menus are an extension of "pie menus" (Callahan et al., 1988). The novice user presses down on a stylus and waits for a short interval of time (approximately 1/2 second). A pie menu of the available commands then appears directly under the cursor (Figure 7.7a). The user may then select a command from the pie menu by keeping the stylus tip depressed and making a stroke through the desired sector or slice of the pie. The slice is highlighted and the selection is confirmed when the pressure on the stylus is released.

The other option is to "mark ahead" by making the mark without waiting for the pie menu to pop up (Figure 7.7b). The physical movement involved in selecting a command is identical to the physical movement required to make the mark corresponding to that command. For example, a command that requires movement up and to the right for pie menu selection requires marking up and to the right in order to invoke that command. The concept is similar to that of accelerator keys in many of today's applications. A user is reminded of the keystrokes associated with particular menu items every time a menu is displayed since the name of the corresponding keys may appear next to the commands. With marking menus, the user is not only reminded, but actually rehearses the physical movement involved in making the mark every time a selection from the menu is made. We believe that this further enhances the association between mark and command.

FIGURE 7.7

Marking menus. The transition from novice to expert reflected in two ways of invoking commands (Kurtenbach and Buxton, 1991b).



Supporting markings with pie menus in this way helps users make a smooth transition from novice to expert. Novices in effect perform menu selection. Users almost always wait for the pop-up menu and then select the desired sector when they first encounter a new menu. Yet waiting for the menu takes time. Thus, as users begin to memorize the layout (as they become expert), they begin to mark ahead to invoke the commands. We have also observed an intermediate stage where users may begin to make a mark, and then wait for the menu to pop-up in order to verify their choice of action.

Why not just use pie menus rather than marks? Marks are faster after one has memorized the layout of the menu. Even if a user did not have to pause to signal for the menu to be popped up, one would still have to wait for the menu to be displayed before making a selection. In many systems, displaying the menu can be annoyingly slow and visually disturbing. There is anecdotal evidence that expert users avoid these problems by "mousing ahead" in pie menu systems (Hopkins, 1991).

One disadvantage of this technique is that "reselection" is not possible using marks, in other words, one lacks the ability to change the item being selected before actually executing an item. For example, one can pop-up the menu and highlight a series of items before releasing the mouse button while on the desired item. Typically this behavior is exhibited by novices who are unfamiliar with the menu items. However, once familiar with a menu, users rarely use reselection. Hence an expert using marks has little need for reselection.

Working Within the Marking Idiom

In music, composers choose instruments that best match their musical ideas, and then write in the appropriate *idiom*. Problems with current marking interfaces are sometimes caused by designers not adequately conceptualizing the idiom of marking systems.

Let us assume, for example, that character and cursive script recognition were 100% accurate. The systems might still not be successful. Even cursive writing is much slower than three-fingered typing. Why then write? Reasons include the ability to do free-form layout, include sketches, and incorporate graphical aspects of text layout. For example, it is faster to write "$2^2 \neq 22$" than to type it on a keyboard. Yet virtually none of the properties of these benefits are captured by current recognizers, which force the user to write rectilinear text, more often than not in little bounding boxes.

If one merely needs to enter linear text and a keyboard is not available, it is faster to enter text with a stylus on a graphical keyboard than it is to print it or to write it cursively. The text thus entered will be more accurate and more legible than that which is written. (For example, there will be no more confusion between the letter *O* and the digit *0*.)

A graphical keyboard may not be appropriate or desirable. On a palmtop computer, for example, it would not fit unless the "keys" were inordinately small. If one is to enter text using marks, yet is concerned with speed, then an alternative is to use *shorthand*. While recognizing traditional shorthand has been investigated (Leedham et al., 1984), it is technically problematic. There are relatively few people who have the skill, and the skill is not easy to acquire. In some applications, however, the designer can invent an effective shorthand notation that does not have these disadvantages. The music shorthand illustrated in Buxton (1986b) is one such example. A more generally applicable example is the Unistrokes shorthand developed by Goldberg and Richardson (1993, 1993 video).

The Unistrokes alphabet is shown in Figure 7.8. In keeping with the name, each character is represented by a single stroke mark. The alphabet must be learned, but this only takes about an hour, aided by the use of mnemonics (Figure 7.9).

FIGURE 7.8

The Unistrokes alphabet.

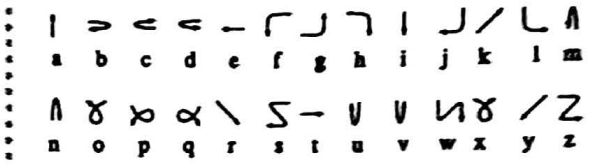
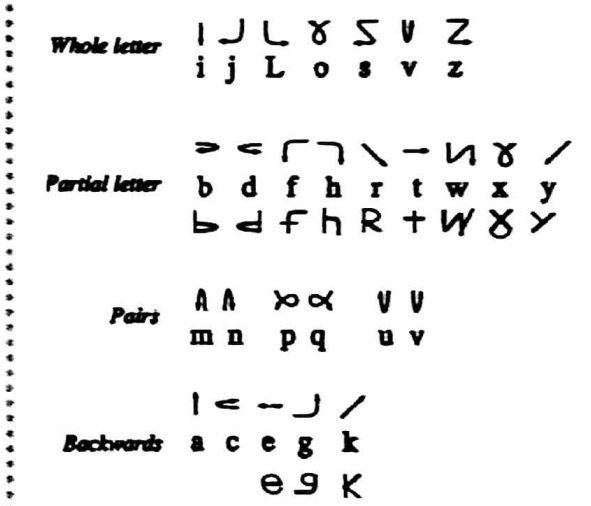


FIGURE 7.9

Unistroke mnemonics.



Unistrokes are interesting for at least two reasons beyond the potential for faster input. Since each character is fully specified by a single stroke, there is no *segmentation problem*. That is, the system need not invest any energy trying to figure out which stroke belongs to which character. Hence there is no need to print characters within the confines of bounding boxes. Characters can even be recognized when the Unistrokes are written one on top of the other. This would be useful in entering data into a wristwatch computer by writing on a touch-sensitive watch face, which may be superior to using the microkeyboards now on some wristwatches.

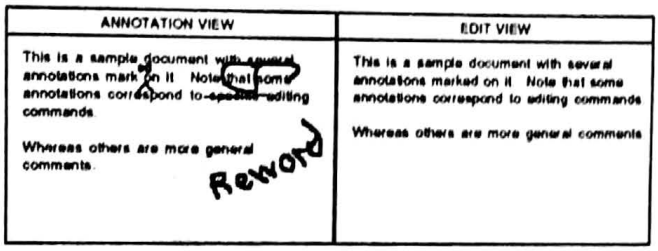
Second, because of the lack of segmentation problems, one need not look at the "page," or display, when writing Unistrokes, so "heads up" writing is possible. Unlike paper and pen technologies, one can visually attend to the whiteboard in a lecture, or a document which one is transcribing, and still take legible notes. So we obtain one of the key benefits of touchtyping in a marking interface.

Another aspect of the idiom is that, unlike direct manipulation GUIs, it is natural for marking systems *to leave an explicit*

audit trail of the user's actions, as shown in Hardock, Kurtenbach, and Buxton (1993). In their MATE system, users mark annotations on electronic documents and then return them to the author. The author sees two views of the document: one with the annotated original text, the other with the modified, or edited text (Figure 7.10).

FIGURE 7.10

MATE in incorporation mode. In incorporation mode, a user can view the annotated document and select which annotations to incorporate. Annotations (seen in the left window) that have been "executed" appear as thin lines (e.g., "ed"). Annotations that have not been executed appear as thick lines. Annotations are color coded based on who made them. Annotations that represent commands can be executed by selecting them with the stylus. Annotations that have been executed can be "undone" by reselecting them. The document's current state appears in the right window. The user can navigate (scroll) independently or concurrently in each window (Hardock, Kurtenbach, and Buxton, 1993).



Unlike Freestyle, the annotations can be invoked by the author to effect the indicated change. This is done by a point/select operation on the annotation. When invoked, the annotation remains visible, but "grayed out" so as to be distinguished from annotations that have not yet been acted upon. These grayed out marks represent the audit trail of actions taken. The benefit comes when one wants to *undo* an action. This is effected simply by reselecting a grayed-out mark. This differs from conventional direct manipulation systems in that it largely doesn't matter when the undone command was originally invoked. Since all past operations are visible (as grayed-out marks), one has an effective undo capability without needing to be too concerned about the order in which things were done.

Finally, one of the most important aspects of the marking idiom is the *figure/ground* relationship that exists between computer-printed vs. human-printed text. Even from a distance, it is absolutely clear which marks were made from the computer and which by hand. This property makes marking very powerful in annotation applications.

Situated Design

Even if character and cursive script recognition were perfect, this would not solve the problems of marking systems. No matter how good the recognition, it is still usually faster to find an address in your address book than in your personal digital assistant (PDA). The same is true for checking appointments in a calendar, looking up a word in the dictionary, or starting to take notes in a meeting.

When we specify systems, performance of the display, bus, and disk drive are rigorously defined. It is strange that we do not do the same in specifying applications. As an exercise, take any of the transactions described above (and others for which PDAs are used) and time them with a stopwatch. The times obtained from the status quo paper and pen condition should set the *minimum* specification for the electronic version, or there had better be a very good reason why not. It is not enough to have the "best" calendar program on the market. It must also be there when I need it and where I need it, and be superior to existing technologies.

Designers must, therefore, pay more attention to the context in which tools are to be used and the constraints these contexts imply.

Summary

Marking interfaces have great potential. For this potential to be realized, we have to abandon the preoccupation with character and script recognition, and also pay attention to other important issues.

The weaknesses of current systems is largely due to a lack of user-centered design. We have already shown examples where use and context should help define performance specifications. Too much design is still carried out without real user involvement and user testing (see Chapter 2). A notable exception is the work by Wolf and her colleagues (Wolf, 1986, 1988, 1992; Wolf and Morrel-Samuels, 1987).

A good way to monitor recent developments in mark-based computing is to subscribe to the bimonthly newsletter *Pen-Based Computing* (1995).

GESTURES

Many people use the term "gesture" to refer to marking interfaces. While every mark is based on a gesture, it is the resulting mark and not the gesture that is used as input to the system. There is a distinct class of system in that it is truly the gesture itself which is recognized. Typically, such systems leave no marks and produce more dimensions of input than the x,y point stream of marking input.

One common way to capture manual gestures, particularly in virtual reality applications (Chapter 14), is using the instrumented glove mentioned above. Fels and Hinton (1990) describe a novel alternative, a prototype neural network system called Glovetalk, which recognizes manual sign language as input and produces continuous speech as output.

One of the pioneers in gesture-based input is Myron Krueger (1991, 1985 video, 1988 video). What is novel about his work is that it does not require any intrusive technology such as gloves. The input is acquired by the system via a video camera. By coupling the video signal with real-time image processing, the computer is able to "see" and recognize the manual gestures.

Gesture is even more powerful when combined with other modalities, such as direct manipulation (Rubine, 1992 video), and especially voice (Bolt, 1984; Schmandt et al., 1984 video; Thorisson, Koons, and Bolt, 1992 video; Koons, Sparrell, and Thorisson, 1993; Koons and Sparrell, 1994 video). This is a topic that we explore further in Chapter 9.

Gestures in Collaborative Work

Computer-supported collaborative work (Chapter 11) is an area that puts special demands on input. Effective meetings require rapid and fluid interaction (Wolf et al., 1991; Elrod et al., 1992).

In meetings where people are at different sites, however, the remote awareness problem arises. Consider distributed audio/graphic conferencing using shared view software such as the

Tivoli system discussed in the reading by Pedersen et al. (1993). If the people at the remote site want to point or indicate something, their only means is via a remote cursor, or telepointer. What this does is reduce their gestural vocabulary to little more than that of a fruit fly. Although a great improvement over no pointing and gesturing, it falls short of what one can do in face-to-face meetings.

What is missing is the superimposition of the hands' and arms gestures over the work surface. This shortcoming has been addressed in innovative papers by Tang and Minneman (1991a, 1991b). What they do is capture the image of the hands over the work surface and transmit it to the remote site, where it appears as a shadow. Hence far richer gestural interaction is possible. Further advances in enriching machine-mediated gestural communications may be seen in the elegant work of Hiroshi Ishii described in Chapter 11 (see Ishii, Kobayashi, and Arita, 1994; Ishii, Arita, and Kobayashi, 1992 video). What remains is to add Krueger's recognition capabilities to these approaches.

TWO-HANDED INPUT

Consider how people use their hands in performing tasks in the everyday world, such as painting, threading a needle, taking notes, opening a book, and driving a car. Each is an example of people's ability to coordinate the action of their two hands. Each contrasts with how the two hands are typically used today in interacting with computers.

Both hands are certainly used in typing, but this is a single task made up of discrete events. We also see two-handed usage in activating function and/or accelerator keys with one hand, while pointing with a mouse with the other. In this case, the button pushes are discrete and the pointing continuous. The class of interaction rarely seen in interacting with computers, and illustrated by the real-world examples given above, has both hands performing continuous tasks in concert. Incorporating such *asymmetrical bimanual action* can greatly improve the quality of direct manipulation interaction.

Guiard (1987) claims that bimanual asymmetric actions in the everyday world can be characterized by three properties:

- *The nondominant hand determines the frame of action of the dominant hand.* Two examples are holding a nail that is to be hammered, and holding a needle that is to be threaded.
- *The sequence of action is nondominant hand, then dominant.* This is seen in the nail and the needle examples, as well as that of a painter moving his or her palette to a convenient location, then dipping his or her brush into the desired paint pot.
- *The action of the nondominant is coarse relative to the fine action of the dominant hand.* This is seen in the example of the painter's palette. The positioning of the palette is not as demanding as the accuracy required in dipping the brush into the appropriate paint pot.

If we are to design our user interfaces to exploit everyday skills, then these three characteristics suggest how the two hands can be used. Buxton (1990b) works through a MacPaint example, where one is "painting" and wants to continue to paint on a part of the "page" that is not visible in the window. It contrasts the complexity of scrolling the page and resuming painting using two techniques: the standard "hand" dragging tool, versus using a trackball in the nondominant hand. The former takes eight steps, the latter, one. The former interrupts the flow of action (painting). In the latter, as in the physical world, one does not have to lay the "paint brush" down in order to reposition the paper.

The ideas underlying the previous example were tested formally in an experiment by Buxton and Myers (1986). People spontaneously used two hands in performing compound tasks when there was a good match between task and action. This study refutes the often heard complaint about two-handed action—"It is difficult to rub your stomach while tapping your head." Although there are cases where two-handed action is very difficult and requires a high degree of skill, there are many bimanual tasks in which users are already skilled. Buxton and Myers demonstrated that these skills could be applied in performing *compound tasks*, such as *positioning and scaling*, and *navigation and selection*.

More recently, Bier et al. (1993) have developed a new two-handed paradigm of interaction that is consistent with Guiard. The paradigm is called the *see-through interface*, and it uses two new classes of widgets called *tool glass* and *magic lenses*. It is best thought of as a 2 ½ D interface that functions on three planes:

- The *desktop* on which icons sit. This is consistent with conventional GUIs.
- The *cursor* that floats above the desktop and its icons. This is typically manipulated by the dominant hand, with a device such as a mouse. This is consistent with conventional GUIs.
- The *magic lens* and *toolglass sheets*, which lie between the cursor and the desktop. These sheets are much like the plastic protractors and rulers that one gets with drafting sets: you can see the tool and the markings on it, but also see what is on the "paper" below them. The magic lens and toolglass sheets are repositioned using the nondominant hand, using a trackball or small touch tablet, for example.

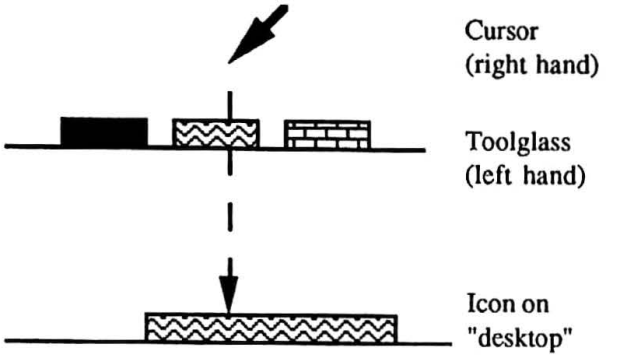
The relationship among these three levels is illustrated in Figure 7.11. The example shows a toolglass sheet with three *click-through buttons* on it. Each represents a different texture that might be assigned to the icon on the desktop. A particular texture is assigned by aligning the cursor and the desired toolglass button over the icon (as represented by the vertical broken line in the figure) and clicking the mouse button.

Magic lenses are also manipulated using the nondominant hand. They are visualization widgets that can be thought of as analogous to magnifying glasses with a diverse range of optical properties. For example, they enable one to highlight vertices, provide X-ray views of scenes, or filter out all objects except those of a particular color.

Bier et al. (1993), Stone, Fishkin, and Bier (1994), Bier et al. (1994), which is included as a reading, and Bier et al. (1994 video) provide an expanded view of the design space. Kabbash, Buxton, and Sellen (1994) presents an experiment that illustrates the usability and utility of toolglass widgets.

FIGURE 7.11

Click-through buttons. Three click-through buttons are shown on a sheet of toolglass. Each button represents a different texture with which icons can be coloured. The button with the desired texture is aligned with the cursor over the icon. Clicking "through" the button over the icon causes it to acquire that button's texture.



Asymmetric bimanual input will be supported by future GUIs. It is perhaps the best way to improve both the "directness" and the "manipulation" of so-called *direct manipulation* interfaces. However, two hands is *not* always better than one. Opportunites for bad design are everywhere, and this includes two-handed interfaces. Using two cursors, one for each hand, for example, should be avoided, as is shown in Kabbash, Buxton, and Sellen (1994) and in Dillon, Edey, and Tombaugh (1990). As with any interaction technique, we need appropriate human-centered design.

REALIZING INPUT'S FULL POTENTIAL

Unfortunately, we are hard-pressed to use the haptic channel to its full potential. We need more experience before this situation can be altered. In many cases this experience is hard to obtain. If, for example, we want to compare two devices systematically, we will very often find that they are incompatible physically, electronically, or logically. Hence, what should be a simple comparison turns into a logistical nightmare.

Yet things are starting to improve, because of the introduction of the *Apple Desktop Bus (ADB)*. This is a standard bus for connecting input devices that was introduced by Apple (but available on some other computers, such as from Silicon Graphics). Because of its design, one can easily switch from one input device to another, thereby enabling one to study their respective properties.

The ease with which devices can be exchanged in an interface should be a prime consideration when choosing a platform for studying HCI.

Transparent Access and the Physically Disabled

For most users, the problems of connecting different input devices to a system, as outlined in the previous section, are an annoyance. However, for users with physical disabilities, these problems can determine whether or not they are able to use a computer at all, and thus can have a major impact on their quality of life. To put it in extreme terms, our discussion of two-handed input assumes that users have the use of two hands. Clearly, designs have to accommodate the special needs of those who do not.

For most common input devices there exist special-purpose transducers that permit people with different physical disabilities to supply comparable signals. A mouse may be replaced by a tongue-activated joystick, or a button replaced by a blow-suck tube. It is reasonable to expect disabled persons to acquire such special-purpose devices. However, it is economically unreasonable and socially unacceptable to expect them to be dependent upon custom applications in order to interact with their systems.

What is required is *transparent access* to standard applications. That is, existing applications should be able to be used by simply plugging in the specialized replacement transducer. The difficulties in providing transparent access are exactly the same difficulties that we encountered above where we wanted to replace one input device with another for comparative purposes. In recognizing that this is a problem "handicapping" all of us, perhaps the achievement of generalized transparent access will become a greater priority than it has been up to now. These and related issues are raised in more detail in Chapter 10.

Device Independence and Virtual Devices

Recently there have been some efforts to overcome some of the problems that stand in the way of transparent access through development of the concept of device-independent graphics.

Just as machine-independent compilers facilitated porting code from one computer to another, device-independent programming constructs have been developed for input-output. With input, the principle idea is that all devices more or less reduce to a small number of generic *virtual devices*. For example, an application can be written in a device-independent way such that it need not know if the source of text input is via a keyboard or a speech-recognition system. All the application need know is that text is being inputted. Similarly, the application need not know what specific device is providing location information (in pointing, for example). All that it need know is what the current location is.

This idea of device independence has been discussed by Foley and Wallace (1974), Wallace (1976), Newman (1968), and Rosenthal et al. (1982). It was refined and integrated into the standardized graphics systems (GSPC, 1977, 1979; ISO, 1983).

Within the Graphical Kernel System (GKS) standard (ISO, 1983), the virtual devices are defined in terms of the values that they return to the application program. The virtual devices for input in GKS are

- locator: a pair of real values giving the coordinate of a point in world coordinates
- stroke: a sequence of x/y coordinates in world coordinates
- valuator: a single real number
- pick: the name of a segment
- string: a string of characters
- choice: an integer defining one of a set of alternatives

For the designer of user interfaces, the main advantage of device-independent graphics has been that one can experiment with different devices without usually having to modify the applications code. All that needs to be changed (from the software perspective) is the actual device driver. The application doesn't care what driver is being used for a particular device because the

standard is defined in terms of the calling protocol and the number and type of parameters returned.

Device-independent graphics has aided us in rapidly prototyping user interfaces (Chapter 2), yet it has also led to problems because some practitioners have confused technical interchangeability with functional interchangeability. Just because I can substitute a trackball for a mouse does not mean that the resulting user interface will still be satisfactory. As we have seen, devices have idiosyncratic properties that make them well suited for some tasks, and not for others (to consider this issue in terms of text input, for example, see Seibel, 1962; Conrad and Longman, 1965; Devoe, 1967; Klemmer, 1971; Kroemer, 1972; Owen, 1978; Rochester, Bequaert, and Sharp, 1978; Montgomery, 1982; Norman and Fisher, 1982; Noyes, 1983; Darragh, Witten, and James, 1990; and Matias, MacKenzie, and Buxton, 1993). Further discussion of problems with the concept of device independence appears in Baecker (1980).

CONCLUSION

Although the difficulties of physically and logically interfacing input devices to applications have impeded development, these logistical problems are diminishing. Perhaps more significant is that many people think about input only at the device level, as a means of obtaining improved time-motion efficiency through use of the sensorimotor system. The reading (Buxton, 1986b) makes clear that this is a big mistake. Effectively structuring the pragmatics of input can also have a significant impact on the cognitive level of the interface. This chapter has been directed towards exploiting that potential.

BIBLIOGRAPHY

Note: A ● before a reference indicates a reading.

Apte, A., Vo, V., and Kimura, T. (1993). Recognizing Multistroke Geometric Shapes: An Experimental Evaluation. *Proc. UIST '93*, ACM, 121–128.

Baecker, R. (1980). Towards an Effective Characterization of Graphical Interaction. In Guedj, R., ten Hagen, P., Hopgood, F., Tucker, H., and Duce, D. (Eds.), *Methodology of Interaction*, North Holland, 127–148. Reprinted in Baecker and Buxton (1987).

Baecker, R., and Buxton, W. (1987). *Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan Kaufmann.

Bier, E. (1990). Snap-Dragging in Three Dimensions. *Computer Graphics* 24(2), ACM, 249–262.

● Bier, E., Stone, M., Fishkin, K., Buxton, W., and Baudel, T. (1994). A Taxonomy of See-Through Tools. *Proc. CHI '94*, ACM, 358–364.

Bier, E., Stone, M., Pier, K., Buxton, W., and DeRose, T. (1993). Toolglasses and Magic Lenses: The See-Through Interface. *Computer Graphics* 27(3), ACM, 73–80.

Bolt, R. (1984). *The Human Interface: Where People and Computers Meet*. Lifetime Learning Publications.

Briggs, R., Dennis, A., Beck, B., and Nunamaker, J. (1993). Whither the Pen-Based Interface. *Journal of Management Information Systems* 9(3), 71–90.

Brooks, F., Ouh-Young, M., Batter, J., and Kilpatrick, P. (1990). Project GROPE—Haptic Displays for Scientific Visualization. *Computer Graphics* 24(3), ACM, 177–185.

Buxton, W. (1982). An Informal Study of Selection-Positioning Tasks. *Proc. Graphics Interface '82*, 323–328.

Buxton, W. (1983). Lexical and Pragmatic Considerations of Input Structures. *Computer Graphics* 17(1), ACM, 31–37.

Buxton, W. (1986a). There's More to Interaction Than Meets the Eye: Some Issues in Manual Input. In Norman, D., and Draper, S. (Eds.), *User Centered System Design*, Lawrence Erlbaum Associates, 319–337.

● Buxton, W. (1986b). Chunking and Phrasing and the Design of Human-Computer Dialogues. In Kugler, H. J. (Ed.), *Information Processing '86*, Proc. IFIP 10th World Computer Congress, North Holland, 475–480.

Buxton, W. (1990a). A Three State Model of Graphical Input. In *Proc. INTERACT '90*, Elsevier Science (North-Holland), 449–456.

Buxton, W. (1990b). The Natural Language of Interaction: A Perspective on Non-Verbal Dialogues. In Laurel, B. (Ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, 405–416.

Buxton, W., Hill, R., and Rowley, P. (1985). Issues and Techniques in Touch-Sensitive Tablet Input. *Computer Graphics* 19(3), ACM, 215–224.

Buxton, W., and Myers, B. (1986). A Study in Two-Handed Input. *Proc.CHI '86*, ACM, 321–326.

Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. (1988). An Empirical Comparison of Pie vs. Linear Menus. *Proc. CHI '88*, ACM, 95–100.

Card, S., English, W., and Burr, B. (1978). Evaluation of Mouse, Rate Controlled Isometric Joystick, Step Keys and Text Keys for Text Selection on a CRT. *Ergonomics* 21(8), 601–613.

Card, S., Mackinlay, J., and Robertson, G. (1990). The Design Space of Input Devices. *Proc. CHI '90*, ACM, 117–124.

Card, S., Mackinlay, J., and Robertson, G. (1991). A Morphological Analysis of the Design Space of Input Devices. *ACM Transactions on Information Systems* 9(2), 99–122.

Card, S., Moran, T., and Newell, A. (1980). The Keystroke Level Model for User Performance Time with Interactive Systems. *Communications of the ACM* 23(7), 396–410.

Card, S., Moran, T., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.

Carr, R. (1991). The Point of the Pen. *Byte* 16(2), 211–221.

Carr, R., and Shafer, D. (1991). *The Power of PenPoint*. Addison-Wesley.

Chapanis, A., and Kinkade, R. (1972). Design of Controls. In Van Cott, H., and Kinkade, R. (Eds.), *Human Engineering Guide to Equipment Design*, Revised Edition, U.S. Government Printing Office, 345–379.

Chen, M., Mountford, J., and Sellen, A. (1988). A Study in Interactive 3-D Rotation Using 2-D Control Devices. *Computer Graphics* 22(4), ACM, 121–129.

Conrad, R., and Longman, D. (1965). Standard Typewriter Versus Chord Keyboard: An Experimental Comparison. *Ergonomics* 8, 77–88.

Darragh, J., Witten, I., and James, M. (1990). The Reactive Keyboard: A Predictive Typing Aid. *IEEE Computer* 23(11), November, 41–49.

Devoe, D. B. (1967). Alternatives to Handprinting in the Manual Entry of Data. *IEEE Transactions on Human Factors in Electronics* 8(1), 21–32.

Dillon, R., Edey, J., and Tombaugh, J. (1990). Measuring the True Cost of Command Selection: Techniques and Results. *Proc. CHI '90*, ACM, 19–25.

Elrod, S., Bruce, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., Tang, J., and Welch, B. (1992). Liveboard: A Large Interactive Display Supporting Group Meetings, Presentations and Remote Collaborations. *Proc. CHI '92*, ACM, 599–607.

English, W., Engelbart, D., and Berman, M. (1967). Display Selection Techniques for Text Manipulation. *IEEE Transactions on Human-Factors in Electronics* 8(1), 5–15.

Evans, K., Tanner, P., and Wein, M. (1981). Tablet-Based Valuators That Provide One, Two, or Three Degrees of Freedom. *Computer Graphics* 15(3), ACM, 91–97.

Fels, S., and Hinton, G. (1990). Building Adaptive Interfaces with Neural Networks: The Glove-Talk Pilot Study. In *Proc. INTERACT '90*, Elsevier (North-Holland), 683–688.

Fitts, P., and Peterson, J. (1964). Information Capacity of Discrete Motor Responses. *Journal of Experimental Psychology* 67, 103–112.

Foley, J., and Wallace, V. (1974). The Art of Graphic Man-Machine Conversation. *Proceedings of IEEE* 62(4), 462–470.

Foley, J., Wallace, V., and Chan, P. (1984). The Human Factors of Computer Graphics Interaction Techniques. *IEEE Computer Graphics and Applications* 4(11), 13–48.

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990). *Computer Graphics: Principles and Practice*. Addison-Wesley.

Gaver, W. (1991). Technology Affordances. *Proc. CHI '91*, ACM, 79–84.

● Goldberg, D., and Goodisman, A. (1991). Stylus User Interfaces for Manipulating Text. *Proc. UIST '91, ACM*, 127–135.

Goldberg, D., and Richardson, C. (1993). Touch-Typing with a Stylus. *Proc. InterCHI '93*, ACM, 80–87.

Greenstein, J., and Arnaut, L. (1988). Input Devices. In Helander (1988), 495–520.

GSPC (1977). Status Report of the Graphics Standards Planning Committee. *Computer Graphics* 11(3), ACM.

GSPC (1979). Status Report of the Graphics Standards Planning Committee. *Computer Graphics* 13(3), ACM.

Guiard, Y. (1987). Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as Model. *Journal of Motor Behavior* 19(4), 486–517.

Hardock, G., Kurtenbach, G., and Buxton, W. (1993). A Marking Based Interface for Collaborative Writing. *Proc. UIST '93*, ACM, 259–266.

Helander, M. (Ed.) (1988). *Handbook of Human-Computer Interaction*. North-Holland.

Herot, C., and Weinzapfel, G. (1978). One-Point Touch Input of Vector Information from Computer Displays. *Computer Graphics* 12(3), 210–216.

Hopkins, D. (1991) The Design and Implementation of Pie Menus. *Dr. Dobb's Journal*, December 1991, 16–26.

Hutchinson, T., White, K., Martin, W., Reichert, K., and Frey, L. (1989). Human-Computer Interaction Using Eye-Gaze Input. *IEEE Transactions on Systems, Man, and Cybernetics* 19(6), 1527–1534.

Ighii, H., Kobayoshi, M., and Arita, K. (1994). Interactive Designof Seamless Collaboration Media: from Team Work Station to Clear Board. *Communications of the ACM* 37 (8), 83–97.

ISO (1983). *Information Processing-Graphical Kernel System (GKS) Functional Description*. International Standards Organization, ISO/DP 7942.

Iwata, H. (1990). Artificial Reality with Force-Feedback: Development of Desktop Virtual Space with Compact Master Manipulator. *Computer Graphics* 24(3), ACM, 165–170.

Jacob, R. (1991). The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At Is What You Get. *ACM Transactions on Information Systems* 9(3), 152–169.

Kabbash, P., Buxton, W., and Sellen, A. (1994). Two-Handed Input in a Compound Task. *Proc. CHI '94*, ACM, 417–423.

Kabbash, P., MacKenzie, I. S., and Buxton, W. (1993). Human Performance Using Computer Input Devices in the Preferred and Non-Preferred Hands. *Proc. InterCHI '93*, ACM, 474–481.

Klemmer, E. T. (1971). Keyboard Entry. *Applied Ergonomics* 2(1), 2–6.

Koons, D., Sparrell, C., and Thorlsson, K. (1993). Integrating Simultaneous Input from Speech, Gaze, and Hand Gestures. In Maybury, M. (Ed.), *Intelligent Multimedia Interfaces*, AAAI Press / MIT Press, 257–276.

Krueger, M. (1991). *Artificial Reality II*. Addison-Wesley.

Kroemer, K. H. (1972). Human Engineering the Keyboard. *Human Factors* 14(1), 51–63.

Kurtenbach, G., and Buxton, W. (1991a). GEdit: A Testbed for Editing by Contiguous Gesture. *SIGCHI Bulletin* 23(2), ACM, 22–26.

Kurtenbach, G., and Buxton, W. (1991b). Integrating Mark-Up and Direct Manipulation Techniques. *Proc. UIST '91*, ACM, 137–144.

Kurtenbach, G., and Buxton, W. (1993). The Limits of Expert Performance Using Hierarchic Marking Menus. *Proc. InterCHI '93*, ACM, 482–487.

Kurtenbach, G., and Buxton, W. (1994). User Learning and Performance with Marking Menus. *Proc. CHI '94*, ACM, 258–264.

Leedham, C., Downton, A., Brooks, C., and Newell, A. (1984). On-Line Acquisition of Pitman's Handwritten Shorthand as a Means of Rapid Data Entry. *Proc. Interact '84*, Vol. 2, Elsevier (North-Holland), 86–91.

Levine, S., and Ehrlich, S. (1991). The Freestyle System: A Design Perspective. In Klinger, A. (Ed.), *Human-Machine Interactive Systems*, Plenum Press, 3-21. Appears in Case C of this volume.

Lipscomb, J., and Pique, M. (1993). Analog Input Device Physical Characteristics. *SIGCHI Bulletin* 25(3), 40–45.

MacKenzie, I. S. (1992a). Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human Computer Interaction* 7(1), 91–139.

● MacKenzie, I. S. (1992b). Movement Time Prediction in Human-Computer Interfaces. *Proc. Graphics Interface '92*, Morgan Kaufmann, 140–150.

MacKenzie, I. S., and Buxton, W. (1992). Extending Fitts' Law to Two-Dimensional Tasks. *Proc. CHI '92*, ACM, 219–226.

MacKenzie, I. S., Sellen, A. and Buxton, W. (1991). A Comparison of Input Devices in Elemental Pointing and Dragging Tasks. *Proc. CHI '91*, ACM, 161–166.

MacKenzie, I. S., and Ware, C. (1993). Lag as a Determinant of Human Performance in Interactive Systems. *Proc. InterCHI '93*, ACM, 488–493.

Mackinlay, J., Card, S., and Robertson, G. (1990). Rapid Controlled Movement through a Virtual 3D Workspace. *Computer Graphics 24(3)*, ACM, 171–176.

Mackinlay, J., Robertson, G., and Card, S. (1990). Rapid Controlled Movement through Virtual 3D Workspaces. *Proc. CHI '91*, ACM, 455–456.

Matias, E., MacKenzie, I. S., and Buxton, W. (1993). Half-QWERTY: A One-Handed Keyboard Facilitating Skill Transfer from QWERTY. *Proc. InterCHI '93*, ACM, 88–94.

Minsky, M. (1985). Manipulating Simulated Objects with Real-World Gestures Using a Force and Position Sensitive Screen. *Computer Graphics* 18(3), ACM, 195–203.

Montgomery, E. (1982). Bringing Manual Input into the 20th Century. *IEEE Computer* 15(3), 11–18. See also follow-up letters in the May, June, and October 1982 issues.

Newman, W. M. (1968). A Graphical Technique for Numerical Input. *Computing Journal* 11, 63–64.

Norman, D., (1981). Categorization of Action Slips. *Psychology Review* 88 (1), 1–15.

Norman, D. and Fisher, D. (1982). Why Alphabetic Keyboards Are Not Easy to Use: Keyboard Layout Doesn't Much Matter. *Human Factors* 24(5), 509–519.

Noyes, J. (1983). The QWERTY Keyboard: A Review. *International Journal of Man-Machine Studies* 18, 265–281.

Owen, S. (1978). QWERTY Is Obsolete. *Interface Age*, January 1978, 56–59.

● Pedersen, E., McCall, K., Moran, T., and Halasz, F. (1993). Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. *Proc. InterCHI '93*, ACM, 391–398.

Pen-Based Computing: The Journal of Stylus Systems (1995). Stylus Publishing, P.O. Box 876, Sandpoint, Idaho 83864-0876. Tel: (208) 265-5286. e-mail: nickbarab@bix.com.

Pickering, J. (1986). Touch-Sensitive Screens: The Technologies and Their Applications. *International Journal of Man-Machine Studies* 25(3), 249–269.

Pittman, J. A. (1991). Recognizing Handwritten Text. *Proc. CHI '91*, ACM, 271–275.

Plamondon, R., Suen, C. Y., and Simner, M. (Eds.) (1989). *Computer Recognition and Human Production of Handwriting*. World Scientific Publishing.

Potasnak, K. (1988). Keys and Keyboards. In Helander (1988), 475–494.

Potter, R., Berman, M., and Shneiderman, B. (1989). An Experimental Evaluation of Three Touch Screen Strategies within a Hypertext Database. *International Journal of Human-Computer Interaction* 1(1), 41–52.

Potter, R., Shneiderman, B., and Weldon, L. (1988). Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies. *Proc. CHI '88*, ACM, 27–32.

Rochester, N., Bequaert, F., and Sharp, E. (1978). The Chord Keyboard. *IEEE Computer* 11(12), 57–63.

Rosenthal, D. S., Michener, J. C., Pfaff, G., Kessener, R., and Sabin, M. (1982). Detailed Semantics of Graphical Input Devices. *Computer Graphics* 16(3), ACM, 33–43.

Rubine, D. (1991). Specifying Gestures by Example. *Computer Graphics* 25(4), ACM, 327–337.

Sachs, E., Stoop, D. and Roberts, A. (1989). 3-Draw: A Three Dimensional Computer Aided Design Tool. *Proceedings of the 1989 IEEE International Conference on Systems, Man and Cybernetics*, Cambridge, MA, 1194–1196.

Schmandt, C. (1983). Spatial Input/Display Correspondence in a Stereoscopic Computer Graphic Work Station. *Computer Graphics* 17(3), ACM, 253–261.

Sears, A., and Shneiderman, B. (1991). High Precision Touchscreens: Design Strategies and Comparisons with a Mouse. *International Journal of Man-Machine Studies* 34, 593–613.

Seibel, R. (1962). A Feasibility Demonstration of the Rapid Type Data Entry Station. *Research Report No. RC 845*, IBM T. J. Watson Research Center.

Seibel, R. (1972). Data Entry Devices and Procedures. In Van Cott, H., and Kinkade, R. (Eds.), *Human Engineering Guide to Equipment Design*, Revised Edition, Washington: U.S. Govt. Printing Office, 311–344.

Sellen, A., Kurtenbach, G., and Buxton, W. (1992). The Prevention of Mode Errors through Sensory Feedback. *Human Computer Interaction* 7(2), 141–164.

Sherr, S. (Ed.) (1988). *Input Devices*. Academic Press.

Shneiderman, B. (1991). Touch Screens Now Offer Compelling Uses. *IEEE Software* 8(2), 93–107.

Shoemake, K. (1992). ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse. *Proc. Graphics Interface '92*, Morgan Kaufmann, 151–156.

Stone, M., Fishkin, K., and Bier, E. (1994). The Movable Filter as a User Interface Tool. *Proc. CHI '94*, ACM, 306–312.

Suen, C. (Ed.) (1990). *Frontiers in Handwriting Recognition: Proceedings of the International Workshop on Frontiers in Handwriting Recognition*. Centre for Pattern Recognition and Machine Intelligence, Concordia University, Montreal, Quebec, Canada H3G 1M8.

Tang, J., and Minneman, S. (1991a). VideoDraw: A Video Interface for Collaborative Drawing. *ACM Transactions on Information Systems* 9(3), 170–184.

Tang, J., and Minneman, S. (1991b). Videowhiteboard: Video Shadows to Support Remote Collaboration. *Proc. CHI '91*, ACM, 315–322.

Tesler, L. (1981). The Smalltalk Environment. *Byte*, August, 90–147.

Wallace, V. (1976). The Semantics of Graphical Input Devices. *Proceedings of the Siggraph/Sigplan Symposium on Graphical Languages*, 61–65.

Ware, C., and Jessome, D. (1988). Using the Bat: A Six-Dimensional Mouse for Object Placement. *IEEE Computer Graphics and Applications* 8(6), 65–70.

Ware, C., and Osborne, S. (1990). Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Computer Graphics* 24(2), ACM, 175–183.

Wolf, C. (1986). Can People Use Gesture Commands? *SIGCHI Bulletin* 18(2), ACM, 73–74.

Wolf, C. (1988). A Comparative Study of Gestural and Keyboard Interfacess. *Proc. 32nd Annual Meeting of the Human Factors Society*, 273–277.

Wolf, C. (1992). A Comparative Study of Gestural, Keyboard and Mouse Interfaces. *Behaviour and Information Technology* 11(1), 13–23.

Wolf, C., and Morrel-Samuels, P. (1987). The Use of Hand-Drawn Gestures for Text-Editing. *International Journal of Man-Machine Studies* 27, 91–102.

Wolf, C., Rhyne, J., and Ellozy, H. (1989). The Paper-Like Interface. In Salvendy, G., and Smith, M. J. (Eds.), *Designing and Using Human-Computer Interfaces and Knowledge-Based Systems*, Elsevier (North-Holland), 494–501.

Wolf, C., Rhyne, J., Zorman, L., and Ossher, H. (1991). WE-MET (Window Environment-Meeting Enhancement Tools). *Proc. of CHI '91*, ACM, 441–442.

Zhai, S., Buxton, W., and Milgram, P. (1994). The "Silk Cursor": Investigating Transparency for 3D Target Acquisition. *Proc. CHI '94*, ACM, 459–464.

Zimmerman, T., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y. (1987). A Hand Gesture Interface Device, *Proc. CHI+GI '87*, ACM, 189–192.

VIDEOS

Bier, E. (1989). Gargoyle 3D: Snap-Dragging in 3D. *ACM SIGGRAPH Video Review (SGVR)* 47.

Bier, E., and Pier, K. (1987). Snap-Dragging and the Gargoyle Illustration System. *ACM SGVR* 33.

Bier, E., Stone, M., Pier, K., Fishkin, K., Baudel, T., Conway, M., Buxton, W., and DeRose, T. (1994). Toolglasses and Magic Lenses: The See-Through Interface. *ACM SGVR* 97.

Brooks, F., et al. (1981). The Grip-75 Man-Machine Interface. *ACM SGVR* 4.

Buxton, W. (1984). Selection-Positioning Task Study. *ACM SGVR* 12.

Evans, K., Tanner, P., and Wein, M. (1981). Graphics Interaction at NRC. *ACM SGVR* 4.

Goldberg, D., and Richardson, C. (1993). Touch Typing With a Stylus. *ACM SGVR* 88.

Hill, W., and Hollan, J. (1992). Pointing and Visualization. *ACM SGVR* 77.

Ishii, H., Arita, K., and Kobayashi, M. (1992). Toward Seamless Collaboration Media: From Team Work Station to Clear Board. *ACM SGVR* 87.

Koons, D., and Sparrell, C. (1994). Iconic: Speech and Depictive Gestures at the Human-Machine Interface. *ACM SGVR* 97.

Krueger, M. (1985). Videoplace Sampler, *ACM SGVR* 20.

Krueger, M. (1988). Videoplace '88, *ACM SGVR* 40.

Mackinlay, J., Card, S., and Robertson, G. (1991). Rapid Controlled Movement through a Virtual 3D Workspace. *ACM SGVR* 65.

Mantei, M. (1990). The Strauss Mouse. *ACM SGVR* 56.

Myers, B. (Ed.) (1990). All the Widgets. *ACM SGVR* 57.

Myers, R., and Zeller, B. (1990). Silicon Graphics Workspace. *ACM SGVR* 56.

Nelson, D. (1987). Magnetic Fusion Experiment Control Center. *ACM SGVR* 13.

Plaisant, C., and Shneiderman, B. (1991). Scheduling Home Control Devices. *ACM SGVR* 63.

Plaisant, C., and Wallace, D. (1992). Touchscreen Toggle Design. *ACM SGVR* 77.

Roberts, S. (1989). 16,000 Miles on a Bicycle. *ACM SGVR* 47.

Rubine, D. (1992). Combining Gestures and Direct Manipulation. *ACM SGVR* 77.

Rutledge, J., and Selker, T. (1990). In-Keyboard Analog Pointing Stick: A Case for the Pointing Stick. *ACM SGVR* 55.

Sachs, E., Stoop, D. and Roberts, A. (1990). 3-Draw: A Tool for the Conceptual Design of 3D Shape. *ACM SGVR* 55.

Schmandt, C., et al. (1984). Put That There. *ACM SGVR* 13.

Sutherland, I. (1983). Sketchpad. *ACM SGVR* 13. Film originally produced in 1961.

Theil, D. (1991). The Cue Ball as Part of a Gestural Interface. *ACM SGVR* 63.

Thorisson, K., Koons, D., and Bolt, R. (1992). Multi-Modal Natural Dialogue. *ACM SGVR* 76.

Wang Laboratories (1989). Wang Freestyle. *ACM SGVR* 45.

Ward, G. (1985). Software Control at the Stroke of a Pen. ACM *SGVR* 18.

Weiner, D., and Ganapathy, S. (1990). Three-Dimensional Interfaces in Shared Environments. ACM *SGVR* 55.

Wolf, C., Rhyne, J. and Ellozy, H. (1989). The Paper-Like Interface. *ACM SGVR* 47.

Zacharey, G. (1987). The Dataglove. *ACM SGVR* 27.