Chapter 1:

AN INTRODUCTION TO HUMAN INPUT TO COMPUTERS

The human body has 200 joints. 56 of them are in the hands!
[Add Reference]

Introduction

In recent years there has been much talk about the "look and feel" of user interfaces. I often speculate that if typography had to reflect effort invested in input compared to output, this would be more properly written:

Look Feel

"Hands on" computing is a myth, and is more accurately stated as "finger on" (note the singular) except when typing, and in virtually no case do personal computers recognize that our fingers are capable of anything but binary on/off action.

A professional violinist will spend more on a *bow* than computing professionals spend on a workstation. This is simply in order to be able to capture the full subtlety of gestural nuance that they are capable of producing. Meanwhile I still can't buy a personal computer that lets me do what a 15¢ pencil can, namely draw a line whose thickness or strength varies depending on how hard I push, or what angle I hold it.

This is all by way of saying that input to computer systems is still in its infancy. It is a neglected area that is central to improving the quality of human computer interaction. This book is an attempt to help bring about a change in this situation, and stimulate a new wave of activity in research, design and practice.

The type of interaction that we deal with is known as *haptic*. The term comes from a Greek word having to do with contact.

Haptic interaction with computers has primarily to do with input. While we have physical contact with transducers such as mice, their design does not afford us to feel the edges of things that we are pointing at, for example. While rare, there are, however, some haptic output devices. One example would be a device capable of producing output in braille. Another class of device that delivers haptic output is what is known as *force feedback*. These are devices which can both sense users action and which can be controlled by the computer. At the CERN Accelerator in Geneva, for example, a knob has been implemented whose feel can be changed by the computer (Beck and Stumpe, 1973). A number of other force feedback devices are described, albeit briefly, by Minsky, Brooks, Behensky, Milliken, Russo and Druin (1989). Three examples which we will discuss more fully in a later chapter are Cadoz, Luciani and Florens (1984), Iwata (1990) and Brooks, Ouh-Young, Batter & Kilpatrick (1990).

While not necessarily controlled by a computer, *every* haptic input device can be considered to provide some output by way of the tactile or kinesthetic feedback that it provides. In practice, the quality and appropriateness of this "feel" is often extremely important in determining a device's effectiveness and acceptance in a particular context.

The bulk of the discussion will concern manual input, since that is the most common usage, and the richest source from which we can draw examples. However, it is important to remember that the concepts that arise in the use of the hands generally apply to control through other parts of the body. For example, foot pedals provide an alternative approach (Pearson and Weiser, 1988; Sellen, Kurtenbach and Buxton, 1990). Taking into account users with physical disabilities, tongue-activated joysticks, provide another important example.

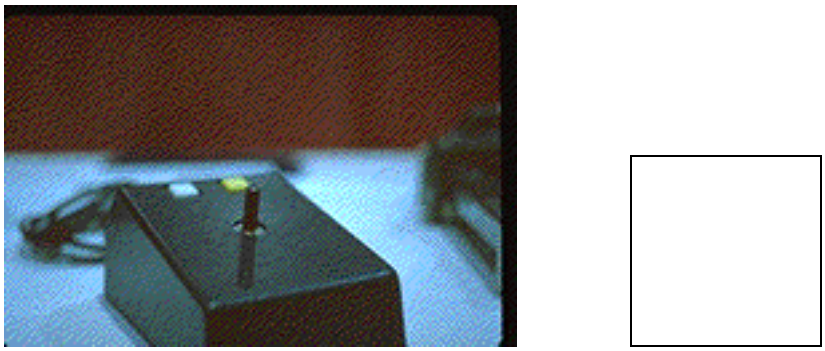


Figure 1: Two Isometric Joysticks

The Choice of Technology Makes a Difference

Each input device has its own strengths and weaknesses, just as each application has its own unique demands. With the wide range of input devices available, one of the problems that confront the designer is to obtain a match among application, input technology and user. Part of the problem has to do with recognizing the relevant dimensions along which the application's demands should be characterized. Another is knowing how each technology being considered performs along those dimensions. Finally, a key consideration is the end user. These are topics addressed below and in Buxton (1986a).

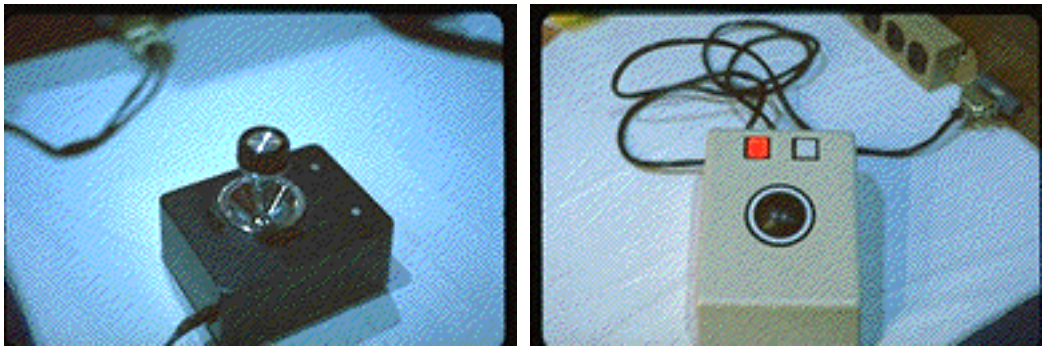
Example 1: The Isometric Joystick

An *isometric joystick* is a joystick whose handle does not move when it is pushed. Rather, its shaft senses how hard you are pushing it, and in what direction. It is, therefore, a pressure-sensitive device. Two isometric joysticks are shown in Figure 1. They are both made by the same manufacturer. They cost about the same, and are electronically identical. In fact, they are plug compatible. How they differ is in their size, the muscle groups that they consequently employ, and the amount of force required to get a given output.

> *Remember, people generally discuss joysticks vs mice or trackballs. Here we are not only comparing joysticks against joysticks, we are comparing one isometric joystick to another. When should one be used rather than the other? The answer obviously depends on the context. What can be said is that their differences may often be more significant than their similarities. In the absence of one of the pair, it may be better to utilize a completely different type of transducer (such as a mouse) than to use the other isometric joystick.*

Example 2: Joystick vs. Trackball

Let's take an example in which subtle idiosyncratic differences have a strong effect on the appropriateness of the device for a particular transaction. In this example we will look at two different devices. One is the joystick shown in Figure 2(a).



(a) (b)

Figure 2: A 3-D Joystick (a) and a 3-D Trackball (b).

In many ways, it is very similar to the isometric joysticks seen in the previous example. It is made by the same manufacturer, and it is plug-compatible with respect to the X/Y values that it transmits. However, this new joystick moves when it is pushed, and (as a result of spring action) returns to the center position when released. In addition, it has a third dimension of control accessible by manipulating the self-returning spring-loaded rotary pot mounted on the top of the shaft.

Rather than contrasting this to the joysticks of the previous example (which would, in fact, be a useful exercise), let us compare it to the 3-D trackball shown in Figure 2(b). (A 3-D trackball is a trackball constructed so as to enable us to sense clockwise and counter-clockwise "twisting" of the ball as well as the amount that it has been "rolled" in the horizontal and vertical directions.)

This trackball is plug compatible with the 3-D joystick, costs about the same, has the same "footprint" (consumes the same amount of desk space), and utilizes the same major muscle groups. It has a great deal in common with the 3-D joystick of Figure 2(a).

In many ways the trackball has more in common with the joystick in Figure 2(a) than do the joysticks shown in Figure 1!

If you are starting to wonder about the appropriateness of always characterizing input devices by names such as "joystick" or "mouse", then the point of this section is getting across. It is starting to seem that we should lump devices together according to some "dimension of maximum significance", rather than by some (perhaps irrelevant) similarity in their mechanical construction (such as being a mouse or joystick). The prime issue arising from this recognition is the problem of determining which dimension is of maximum significance in a given context. Another is the weakness of our current vocabulary to express such dimensions. Despite their similarities, these two devices differ in a very subtle, but significant, way. Namely, it is much easier to simultaneously control all three dimensions when using the joystick than when using the trackball. In some applications this will make no difference. But for the moment, we care about instances where it does. We will look at two scenarios.

Scenario 1: CAD

We are working on a graphics program for doing VLSI layout. The chip on which we are working is quite complex. The only way that the entire mask can be viewed at one time is at a very small scale. To examine a specific area in detail, therefore, we must "pan" over it, and "zoom in". With the joystick, we can pan over the surface of the circuit by adjusting the stick position. Panning direction is determined by the direction in which the spring-loaded stick is off-center, and speed is determined by its distance off-center. Zooming is controlled by twisting the shaft of the joystick.

Which way should one twist to zoom in and which way to zoom out? Why?

With the trackball, we exercise control by rolling the ball in the direction and at the speed that we want to pan. Panning is easier with trackball than the spring-loaded joystick. This is because of the strong correlation (or compatibility) between stimulus (direction, speed and amount of roll) and response (direction, speed and amount of panning) in this example. With the spring-loaded joystick, there was a position-to-motion mapping rather than the motion-to-motion mapping seen with the trackball. Such cross-modality mappings require learning and impede achieving optimal human performance. However, if our application demands that we be able to zoom and pan simultaneously, then we have to reconsider our evaluation. With the joystick, it is easy to zoom in and out of regions of interest while panning. One need only twist the shaft-mounted pot while moving the stick. However, with the trackball, it is nearly impossible to twist the ball at the same time that it is being rolled. The 3D trackball is, in fact, better described as a 2+ device.

Scenario 2: Process Control

We are using the computer to control an oil refinery. The pipes and valves of a complex part of the system are shown graphically on the CRT, along with critical status information. My job is to monitor the status information and when conditions dictate, modify the system by adjusting the settings of specific valves. I do this by means of *direct manipulation*. That is, valves are adjusted by adjusting their graphical representation on the screen. Using the joystick, this is accomplished by pointing at the desired valve, then twisting the pot mounted on the stick. However, it is difficult to twist the joystick-pot without also causing some change in the X and Y values. This causes problems, since graphics pots may be in close proximity on the display. Using the trackball, however, the problem does not occur. In order to twist the trackball, it can be (and is best) gripped so that the finger tips rest against the bezel of the housing. The finger tips thus prevent any rolling of the ball. Hence, twisting is orthogonal to motion in X and Y. The trackball is the better transducer in this example *precisely because of its idiosyncratic 2+1 property*.

Thus, we have seen how the very properties that gave the joystick the advantage in the first scenario were a liability in the second. Conversely, with the trackball, we have seen how the liability became an advantage. What is to be learned here is that if such cases exist between these two devices, then it is most likely that comparable (but different) cases exist among all devices. What we are most lacking is some reasonable methodology for exploiting such characteristics via an appropriate matching of device idiosyncrasies with structures of the dialogue.

Strength vs Generality

In the previous example we saw how the idiosyncratic properties of an input device could have a strong affect on its appropriateness for a specific task. It would be nice if the world was simple, and we could consequently figure out what a system was for, find the optimal device for the task to be performed on it, and be done. But such is seldom the case. Computer systems are more often used by a number of people for a number of tasks, each with their own demands and characteristics. One approach to dealing with the resulting diversity of demands is to supply a number of input devices, one optimized for each type of transaction. However, the benefits of the approach would generally break down as the number of devices increased. Usually, a more realistic solution is to attempt to get as much generality as possible from a smaller number of devices. Devices, then, are chosen for their range of applicability. This is, for example, a major attraction of graphics tablets. They can emulate the behavior of a mouse. But unlike the mouse, they can also be used for tracing artwork to digitize it into the machine.

Having raised the issue, we will continue to discuss devices in such a way as to focus on their idiosyncratic properties. Why? Because by doing so, we will hopefully identify the type of properties that one might try to emulate, should emulation be required.

Appropriate Gestures can Simplify Syntax

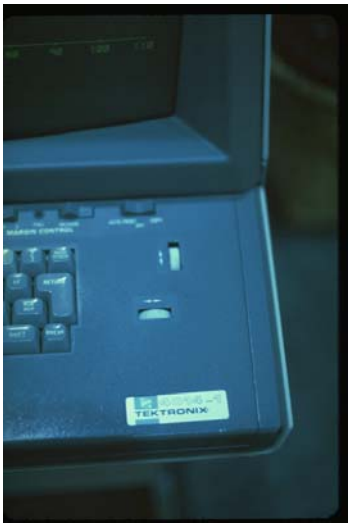
It is often useful to consider the user interface of a system as being made up of a number of horizontal layers. Most commonly, syntax is considered separately from semantics, and lexical issues independent from syntax. Much of this way of analysis is an outgrowth of the theories practiced in the design and parsing of artificial languages, such as in the design of compilers for computer languages. Thinking of the world in this way has many benefits, not the least of which is helping to avoid "apples-and-bananas" type comparisons. There is a problem, however, in that it makes it too easy to fall into the belief that each of these layers is independent. A major objective of this section is to point out how false an assumption this is. In particular, we will illustrate how decisions at the lowest level, the choice of input devices, can have a pronounced effect on the complexity of the system and on the user's mental model of it.

Two Children's Toys

The *Etch-a-Sketch* (shown in Figure 3(a)) is a children’s drawing toy that has had a remarkably long life in the marketplace. One draws by manipulating the controls so as to cause a stylus on the back of the drawing surface to trace out the desired image. There are only two controls: both are rotary pots. One controls left-right motion of the stylus and the other controls its up-down motion. For the purpose of the examples that follow, any similarities between the controls of the Etch-a-Sketch and those of the classic *Tektronix 4014 Graphics Terminal* (shown in Figure 3(b)) are purely intentional.

(a) Etch-a-Sketch

A popular children's toy which uses two 1 dimensional controls for drawing. The left knob controls horizontal movement of the drawing "stylus" and the right knob vertical movement.



(b) Tektronix 4014 Graphics Terminal

For those of us of a certain generation, this is the device on which we did some of our first computer graphics in the mid '70s. If you need a push to bridge the gap between the Etch-a-Sketch toy and "serious" computer design, note the similar use of two 1 dimensional rotary potentiometers for graphics input

Figure 3: Computers as Toys / Toys as Computers

The *Skedoodle* (shown in Figure 4) is another toy based on very similar principles. In computerese, we could even say that the two toys are semantically identical. They draw using a similar stylus mechanism and even have the same "erase" operator (turn the toy upside down and shake it). However, there is one big difference. Whereas the Etch-a-Sketch has a separate control for each of the two dimensions of control, the Skedoodle has integrated both dimensions into a single transducer: a joystick.

Figure 4: The *Skedoodle*
Like the Etch-s-Sketch, the Skedoodle is a children's drawing toy. Other than the oval (rather than rectangular "screen", the main difference between the two is how one draws. The computer terminal on the right is to the Skedoodle what the Tektronix 4014 is to the Etch-a-Sketch.

Since both toys are inexpensive and widely available, they offer an excellent opportunity to conduct some field research. Find a friend and demonstrate each of the two toys. Then ask him or her to select the toy felt to be the best for drawing. What all this is leading to is a drawing competition between you and your friend. However, this is a competition that you will always win. The catch is that since your friend got to choose toys, you get to choose what is drawn. If your friend chose the Skedoodle (as do the majority of people), then make the required drawing be of a horizontally-aligned rectangle, as in Figure 5a. If they chose the Etch-a-Sketch, then have the task be to write your first name, as in Figure 5b. This test has two benefits. First, if you make the competition a bet, you can win back the money that you spent on the toys (an unusual opportunity in research). Secondly, you can do so while raising the world's enlightenment about the sensitivity of the quality of input devices to the task to which they are applied.

(a) Geometric Figure (b) Cursive Script

Figure 5: Two Drawing Tasks

What is true with these two toys (as illustrated by the example) is equally true for any and all computer input devices: they all shine for some tasks and are woefully inadequate for others.

> *If you understand the importance of the points being made here, you are hereby requested to go out and apply this test on every person that you know who is prone to making unilateral and dogmatic statements of the variety "mice (tablets, joysticks, trackballs, ...) are best".*
>
> *A good working premise is to assume that every device is best for something and worst for something else. The trick is understanding what those "somethings" are, and finding the right match among device, task, context and user.*

We can build upon what we have seen thus far. What if we asked how we can make the Skedoodle do well at the same class of drawings as the Etch-a-Sketch? An approximation to a solution actually comes with the toy in the form of a set of templates that fit over the joystick (Figure 6).

If we have a general-purpose input device (analogous to the joystick of the Skedoodle), then we can provide tools to fit on top of it to customize it for a specific application. (An example would be the use of "sticky" grids in graphics layout programs.) However, this additional level *generally comes at the expense of increased cost in the complexity of the control structure*. If we don't need the generality, then we can often avoid this complexity by choosing a transducer whose operational characteristics implicitly channel user behavior in the desired way (in a way analogous to how the Etch-a-Sketch controls place constraints on what can be easily drawn).

Figure 6: Skedoodle with Templates

Summary

In this introduction we have introduced the main themes that pervade this book. We have seen that there is an intimate, but poorly understood, relationship among the user's motor-sensory capabilities and intent, the technique used to achieve that intent, and the transducer.

In what follows, we explore this relationship more deeply, and attempt to establish some theory and models that can be applied by designers in making decisions about input.

Chapter 4:

TAXONOMIES OF INPUT

INTRODUCTION

Traditionally, input devices have been discussed in terms of their mechanical and electrical properties (Foley & Van Dam, 1982; Sherr, 1988). Discussions centre on "joysticks," "trackballs," and "mice," for example.

Several studies have attempted to evaluate the technologies from the perspective of human performance. Many of these are summarized in Greenstein and Arnaut (1988) and Milner (1988). A common problem with such studies, however, is that they are often overly device-specific. While they may say something about a particular device in a particular task, many do not contribute significantly to the development of a general model of human performance. (There are exceptions, of course, such as Card, English and Burr, 1978.)

With the objective of isolating more fundamental issues, some researchers have attempted to categorize input technologies and/or techniques along dimensions more meaningful than simply "joystick" or "trackball." The underlying assumption in such efforts is that better abstractions can lead us from phenomenological descriptions to more general models, and hence better analogies.

THE STANDARDS EFFORT AND LOGICAL DEVICES

Just as machine independent compilers facilitated porting code from one computer to another, device independent programming constructs have been developed for I/O. With input, the principle idea was to recognize that all devices more-or-less reduced to a small number of generic or *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 input. Similarly, the application need not require any information about what specific device is providing location information (in pointing, for example). All that it needs to know is what the current location is.

**** fix **** This idea of device independent, “logical”, or “virtual” input devices was first done for the GPGS graphics system in the early 1970’s and published[1] by Caruthers, Van den Bos & Van Dam, A. (1977) and has been discussed, by among others, discussed by Foley and Wallace (1974), Wallace (1976), Newman (1968), and Rosenthal, Michener, Pfaff, Kessener and Sabin (1982). It was refined and integrated into the standardized graphics systems (GSPC, 1977; GSPC, 1979; ISO, 1983).

Within the 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 number of the type *real*.
- *Pick:* the name of a segment.
- *String:* produces a string of characters.
- *Choice:* returns a non-negative 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 normally 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 parameter returned.*

Because there is one logical device for each generic class of input, the beginnings of a taxonomy based on use is introduced.

Device independent graphics has had an important impact on our ability to rapidly prototype user interfaces. This is a subject discussed in more detail in Chapter 11 of Baecker and Buxton (1987), and is largely motivated by the iterative design methodologies discussed in Chapter 10 of Baecker and Buxton (1987).

While device independence has been a real benefit, it has also led to some problems. The reason is that 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. Further discussion of issues relating to device-independent graphics can be found in Baecker (1980).

GENERIC TRANSACTIONS: FOLEY, WALLACE & CHAN

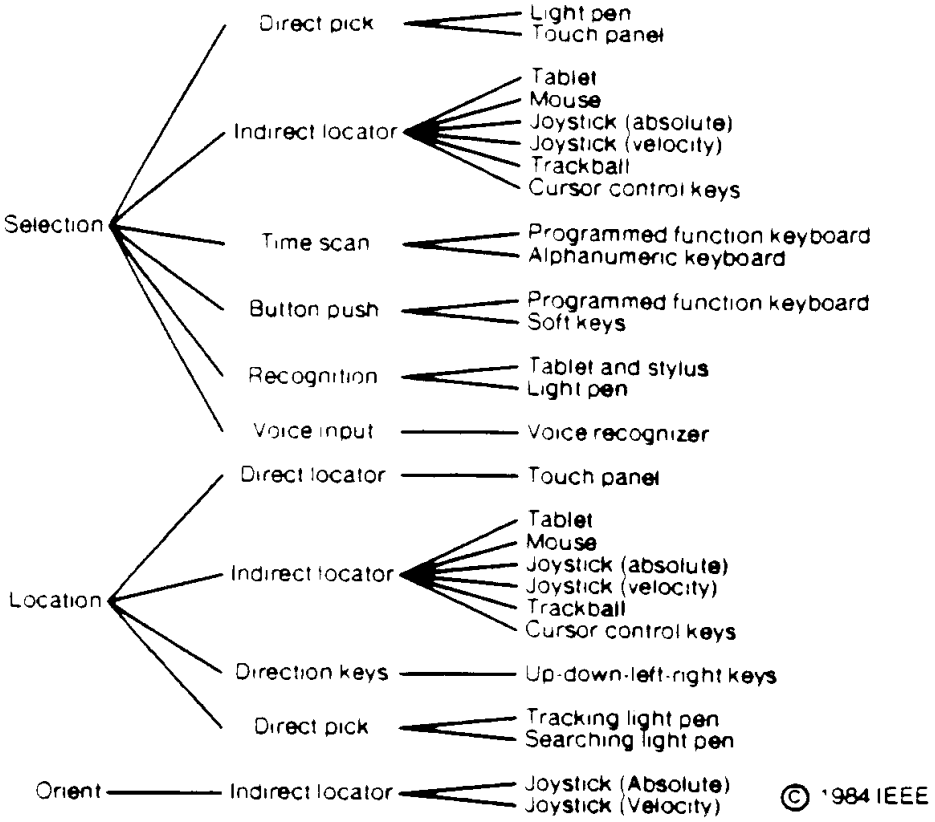
Foley, Wallace, and Chan (1984) took the notion of logical devices, and cast them more in the human perspective than that of the application software. They identified six generic transactions (which were more-or-less the counterparts of the GSPC logical devices) that reflected the user's intentions:

- *Select* an object

[1] Andy at Nimeghen, Peter Veeman at Delpht and Charles Lang , Cambridge University CAD Centre. See Dan Bergeron at New Hampshire for info on team in Delft who helped.

- *Position* an object in 1, 2, 3 or more dimensions;
- *Orient* an object in 1, 2, 3 or more dimensions;
- *Ink,* i.e., draw a line;
- *Text,* i.e., enter text;
- *Value*, i.e., specify a scalar value.

They then proceeded to enumerate a relatively comprehensive set of techniques and technologies capable of articulating each of these basic primitives.



PRAGMATIC CONSIDERATIONS: BUXTON

Buxton (1983) introduced a taxonomy of input devices that was more rooted in the human motor/sensory system. The concern in this case was the ability of various transducers to capture the human gesture appropriate for articulating particular intentions. Consequently, input devices were categorized by things such as the property sensed (position, motion, or pressure), the number of dimensions sensed, and the muscle groups required to use them.

The examples of the previous section help us explore how different devices lend themselves to different tasks. In this section, we want to develop a categorization of input devices which is based on the properties that cause these differences. Our approach to this is through the tableau, shown in Figure 1. There is a hierarchy of criteria according to which devices are

organized in this table. The tableau is limited, and only considers continuous, manually operated devices. Hence, the first two (implicit) organizational criteria are:

- Continuous *vs.* discrete: continuous
- Agent of control (hand, foot, voice,): hand.
- The table is divided into a matrix whose primary partitioning into rows and columns delimit
- What is being sensed (position, motion or pressure), and
- The number of dimensions being sensed (1, 2 or 3)

These primary partitions are delimited by solid lines. Hence, for example, both the rotary and sliding potentiometer fall into the box associated with one-dimensional position-sensitive devices (top left-hand corner).

These primary rows and columns are sub-divided by thin lines into secondary regions. These group:

- Devices that are operated using similar motor skills (sub columns)
- Devices that are operated by touch *vs* those that require a mechanical intermediary between the hand and the sensing mechanism (sub-rows)

Grouping by motor action can be seen in examining the two-dimensional devices. Since they are in the same sub-column, the tableau 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 sub-column.

The use of sub-rows to differentiate between devices that are touch activated and those that are not can be seen in comparing the light-pen and the touch screen. While each utilizes the same basic motor control, the light-pen requires the use of a stylus. Hence, the two appear in different sub-rows.

Property Sensed		Number of Dimensions							
		1		2				3	
Position		Rotary Pot	Sliding Pot	Tablet & Puck	Tablet & Stylus	Light Pen	Floating Joystick	3D Joystick	M
				Touch Tablet		Touch Screen			T
Motion		Continuous Rotary Pot	Treadmill	Mouse			Trackball	3D Trackball	M
			Ferinstat				X/Y Pad		T
Pressure		Torque Sensor					Isometric Joystick		T

Figure 1: Taxonomy of Input Devices.

Continuous manual input devices are categorized. The first order characterization is property sensed (rows) and number of dimensions (columns) Sub-rows distinguish between devices that have a mechanical intermediary (such as a stylus) between the hand and the sensing mechanism, and those which are touch-sensitive. Sub-columns distinguish among devices that use comparable motor control for their operation (From Buxton, 1983).

The tableau is useful for many purposes by virtue of the structure which it imposes on the domain of input devices. First, it helps in finding appropriate equivalences. This is important in terms of dealing with some of the problems which arose in our discussion of device independence.

The tableau makes it easy to relate different devices in terms of metaphor. For example, a tablet is to a mouse what a joystick is to a trackball. Furthermore, if the taxonomy defined by the tableau can suggest new transducers in a manner analogous to the periodic table of Mendeleev predicting new elements, then we can have more confidence in its underlying premises. 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 Weinzaphel (1978).

Generality and Extensibility

Choosing the input technologies to be used with a workstation generally involves making a trade-off between two conflicting demands. On the one hand, each task has specialized needs that can be best addressed by a specialized technology. On the other hand, each workstation is generally used for a multitude of tasks. Supplying the optimum device for each task is generally out of the question. 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 in this regard.

Stated differently, how many squares in Figure 1 can a particular device fill? Graphics tablets are important in this regard, for example, since they can emulate many of the other transducers. This is demonstrated in detail by Evans, Tanner and Wein (1981). The tablet is what could be called an "extensible" device. This property of extensibility is, in our opinion, an important but seldom considered criterion to be considered in device selection.

Relative vs Absolute Controllers

One of the most important characteristics 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. Another example taken from process control is discussed in the reading by Buxton (1986a). This is the case of what is known as the nulling problem which is introduced when absolute transducers are used in designs where one controller is used for different tasks at different times.

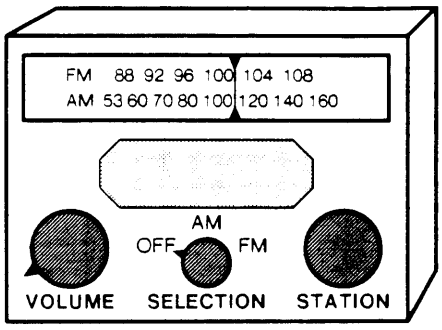
What Our Taxonomy Doesn't Show

Perhaps the main weakness of the taxonomy presented above is the fact that it only considers the continuous aspect of devices. As the sample tasks discussed earlier pointed out, other factors, such as the integration of button devices with continuous controllers has a strong impact on a device's performance. This is clear, for example, in 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 with 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.

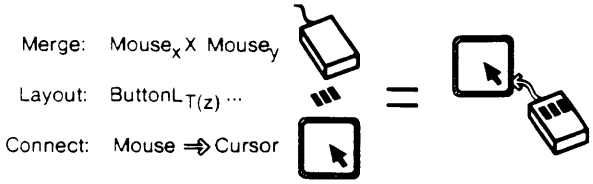
An approach to capturing this aspect of devices is found in the reading by Buxton, Hill and Rowley (1985). Here, a state-transition representation is developed which brings to light some very important, yet subtle, differences among devices. As an exercise, the reader is encouraged to compare a mouse, tablet, and trackball, touch-tablet, and touch-screen in terms of their ability to make the transitions through the three states (0, 1, & 2) described in the reading. As a further exercise, try to characterize each of the sample tasks discussed at the beginning of this section in terms of which of these three states a device must be able to capture before the transaction can be successfully performed.

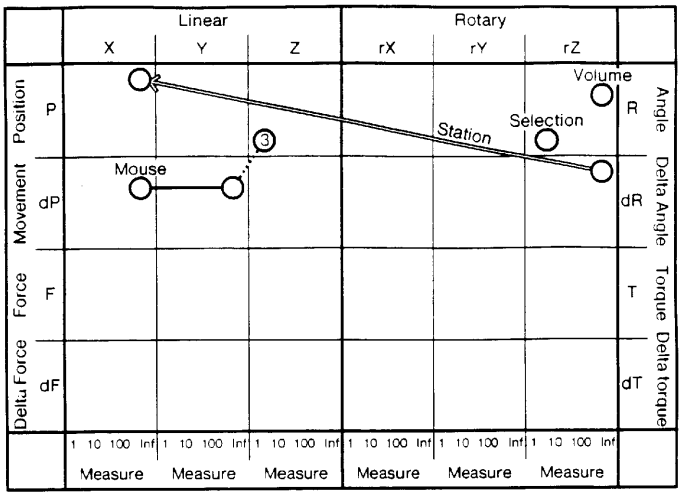
REFINING THE DESIGN SPACE: MACKINLAY, CARD & ROBERTSON

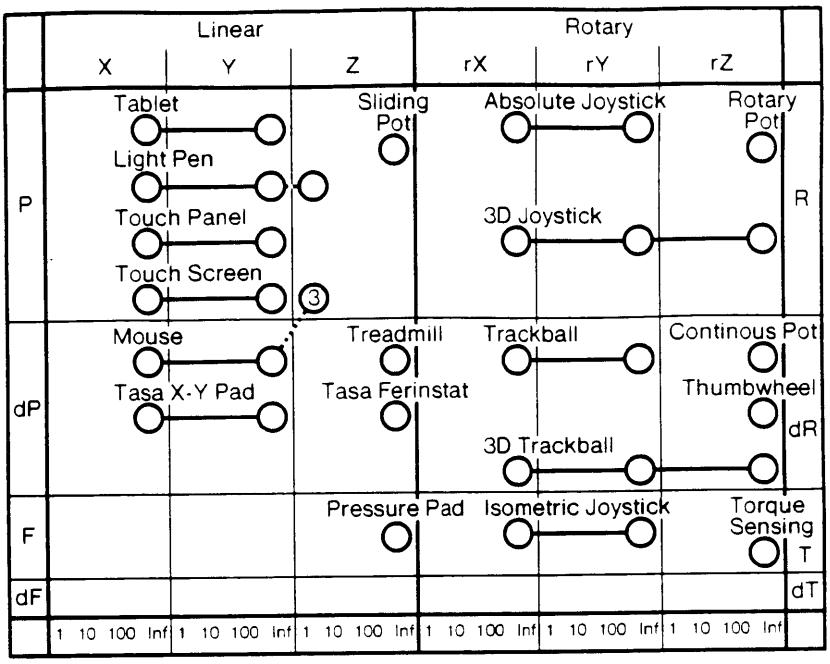
Recent research at Xerox PARC has built on this work (Card, Mackinlay and Robertson,1990; Mackinlay, Card and Robertson, 1990). Their taxonomy captures a broad part of the design space of input devices. The model captures both the discrete and continuous properties of devices, (unlike that of Buxton, which could only deal with the latter).



	Linear	Rotary
Position		
Absolute	Position **P**	Rotation **R**
Relative	Movement **dP**	DeltaRotation **dR**
Force		
Absolute	Force **F**	Torque **T**
Relative	DeltaForce **dF**	DeltaTorque **dT**







THE 3-STATE MODEL

Introduction

The next approach to categorizing devices uses state-transition diagrams. The use of these diagrams to describe interaction with input devices is not new. It was seen as early as 1968 in the work of William Newman (Newman, 1968b), and the work that follows is deeply indebted to Newman's work. The approach to the model that we present here was initially formulated in Buxton, Hill and Rowley (1985). It was subsequently refined in Buxton (1990). Its strength lies in the use of a common 3-state model to represent both the requirements of interactive techniques and the capabilities of specific input devices. Hence, it facilitates the ease with which one can match device to technique, as well as a vocabulary of comparing devices or techniques to one another.

The model is consistent with notion of device state introduced in the PARC model of Card, Mackinlay and Robertson (1990) and Mackinlay, Card and Robertson (1990). Hence it is complimentary to that work. And unlike the tabular model of Buxton, discussed above, this one deals reasonably well with the hybrid discrete/continuous properties of input transducers.

The model can be introduced using common direct manipulation transactions as examples. Consider moving the mouse without depressing the select button. Here, we would describe the state of the system as *tracking.* If, however, we point at an icon, depress the mouse button and, while holding it down, move the mouse, we would be in a new state, *dragging.* These two simple states can be represented graphically, as illustrated in Figure 2.

Now consider now the situation if a touch tablet rather than a mouse was connected to the system. For the purpose of the example, let us assume that the touch tablet is capable of sensing only one bit of pressure, namely touch or no-touch.

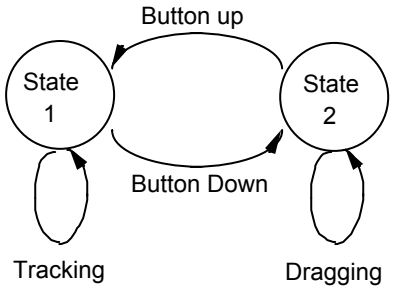


Figure 2: Simple 2-State Transaction

In State 1, moving the mouse causes the tracking symbol to move. Depressing the mouse button over an icon permits it to be dragged when the mouse is moved. This is State 2. Releasing the mouse button returns to the tracking state, State 1.

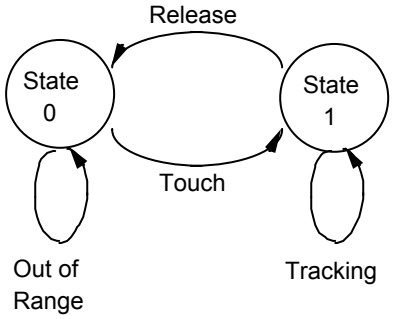


Figure 3: State 0-1 Transaction

Assume a touch tablet. In State 0, moving the finger is out of range (OOR), so has no effect. When the finger is in contact with the tablet, the tracking symbol follows the finger's motion (State 1: tracking). The system returns to State 0 when the finger releases contact from the tablet surface.

In this case we also get two states, but only one is common to the previous example. This is illustrated in Figure 3. The first state, (State 0), is what we will call *out of range*, (OOR). In this state, any movement of the finger has no effect on the system. It is only when the finger comes in contact with the touch tablet that we enter State 1, the tracking state seen in the previous example.

Each example has one state that the other cannot reach. Lifting a mouse off of one's desk is not sensed, and has no effect. No interactive technique can be built that depends on this action. Consequently, State 0 (the OOR condition) is undefined. Conversely, without some additional signal, the touch tablet is incapable of moving into the dragging state (State 2). To do so would require a signal from a supplementary key press or from a threshold crossing on a pressure-sensitive tablet (Buxton, Hill and Rowley, 1985).

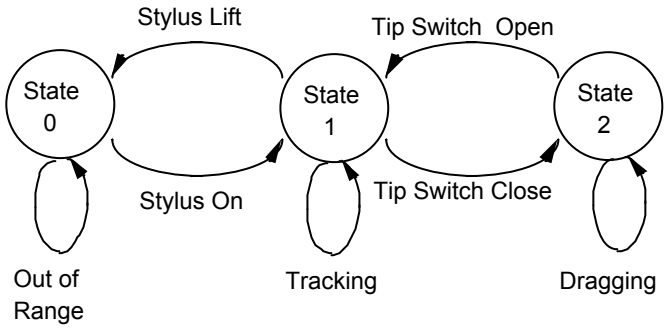


Figure 4: State 0-1-2 Transaction

Assume a graphics tablet with stylus. In State 0, the stylus is off of the tablet and the tip switch in its open state. Moving the stylus has no effect since it is out of range (OOR). When the stylus is in range, the tracking symbol follows the stylus' motion (State 1: tracking). Extra pressure on the stylus closes the tip switch, thereby moving the system into State 2.

There are, however, transducers that are capable of sensing all three states. A graphics tablet with a stylus would be one example.[1] This is illustrated in Figure 4.

The three states introduced in the above examples are the basic elements of the model. There can be some variations. For example, with a multi-button mouse (or the use of multiple clicks), State 2 becomes a set of states, indexed by button number, for example, as illustrated in Fig. 4.

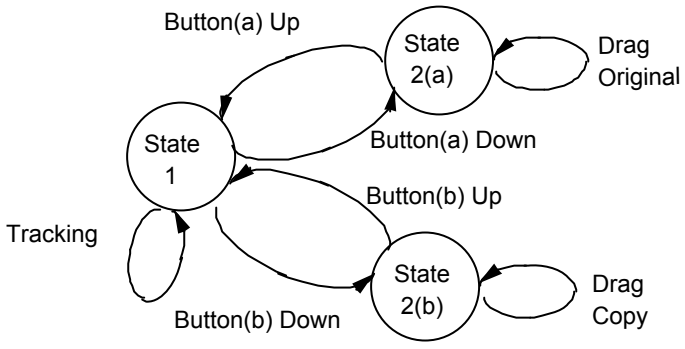


Figure 5: State 2 Set

With a multi-button mouse, for example, multiple State 2's are available. For example, selecting an object with button$_a$ may cause the selected object to be dragged, whereas selecting the object with button$_b$ may mean that a copy is dragged. Multiple State 2s can also be realized by multiple clicks on the mouse, as with the Apple Macintosh, where single clicks are used to select and double clicks are used to "open."

While the model is simple, it will be shown how important properties of devices and interactive techniques can be characterized in terms of these three states, and how this representation can be used in design.

The three states are not arbitrary. The usefulness of the representation depends on consistency in labeling of the states. For example, Figure 2 and Figure 3 are both two-state diagrams. However, in Figure 2 the first state is labeled “1”, while in Figure 3 it is labeled “0”. The reason for this is the desire for consistency. “State 1” is the same in both cases, even if it is the first state in one example and the second in the other. The diagrams look the same but the semantics are different. The significance of this will become clearer as we work through other examples.

Finally, I have chosen to refer to states by number (State 1, for example) rather than by description because the action performed in a particular state can vary. For example, State 2 could just as easily been "inking" or "rubber banding" as "dragging." The ordinal nomenclature is more neutral, and will be used.

Devices and Transactions: Tabulating Attributes

Two tables are presented. The first summarizes the demands of a number of transaction types expressed in terms of the states and state transitions that they require from a supporting transducer. The second summarizes the capabilities of a number of input devices, expressed in terms of this same type of state information. By comparing the two tables, a simple means is provided to evaluate the match between transducers and transactions.

Transaction	State 0	State 1	State 2	Notes
Point			√	
Pursuit Track			√	
Point/Select			√	√	
Drag			√	√	State 2 motion
Rubber Banding			√	√	State 2 motion
Sweep Region			√	√	State 2 motion
Pop/Pull Menu			√	√	State 2 motion
Ink			√	√	State 2 motion
Char Recognition		√	√	√	State 2 motion

Table 1: **State Characteristics of Several Classes of Transaction**

A number of representative types of transactions are listed showing their state and state transition requirements. This table is of use as a means to help verify if a particular transducer is well suited to that class of transaction.

State 1-0 Transitions and Gestures

Transitions from State 1 to State 0 are not significant in most direct manipulation systems. As stylus-driven interfaces using character and gesture recognition become more important, so will this class of state transition. The reason is that this signal is a prime cue to delimit characters.

You can think of it as the need for both the system and the user to be able to sense and agree when the pen has left the "paper."

If this cue is to be used in this or other types of transitions, it is important to note that input devices vary in how well they signal the transition. In particular, the majority of tablets (touch and otherwise) give no explicit signal at all. Rather, the onus is on the application to sense the absence of State 1 tracking information, rather than on the transducer to send an explicit signal that the pointing device has gone out of range.

Not only does this put an additional burden on the software implementation and execution, it imposes an inherent and unacceptable delay in responding to the user's action. Consequently, designers relying heavily on this signal should carefully evaluate the technologies under consideration if optimal performance and efficiency are desired.

Device	State 0	State 1	State 2	Notes
Joystick		√	4	
Joystick & Button		√	√3	
Trackball		√	4	
Mouse		√	√	
Tablet & Stylus	√	√	√	
Tablet & Puck	√1	√	√	
Touch Tablet	√	√	4, 5	
Touch Screen	√	2	√2	6
Light Pen	√	√	√	6

1. The puck can be lifted, but shape and weight discourages this.
2. If State 1 used, then State 2 not available.
3. Button may require second hand, or (on stick) inhibit motion while held.
4. Has no built in button. May require second hand. If same hand, result may be interference with motion while in State 2.
5. State 1-0 transition can be used for selection. See below.
6. Direct device. Interaction is directly on display screen. Special behaviour. See below.

Table 2: State Characteristics of Several Input Devices

A number of representative input devices are listed showing their state and state transition properties. This table is of use as a means to check if a transducer meets the state characteristics required by a particular type of transaction.

Point/Select and State Transitions

Point and select is an integral component of most direct manipulation interfaces. The transaction is compound: pointing, which is a continuous task, and selection, which is binary. In our vocabulary, the binary selection signal is represented as a state change.

As commonly implemented, the pointing task is undertaken in State 1, and the selection is articulated by a State 1-2-1 transition, with no motion in State 2. This can be easily supported with any of the devices in Table 2 that have plain check marks (√) in the State 1 and State 2 columns.

Some transducers, including trackballs, many joysticks, and touch tablets do not generally support State 2. For the most part this is due to their not having buttons tightly integrated into their design. Therefore, they warrant special mention.

One approach to dealing with this is to use a supplementary button. With joysticks and trackballs, these are often added to the base. With trackballs, such buttons can often be operated with the same hand as the trackball. With joysticks this is not the case, and another limb (hand, foot, etc.) must be employed[2]. As two-handed input becomes increasingly important, using two hands to do the work of one may be a waste. The second hand being used to push the joystick or touch tablet button could be used more profitably elsewhere.

An alternative method for generating the selection signal is by a State 1-0 transition (assuming a device that supports both of these states). An example would be a binary touch tablet, where lifting your finger off the tablet while pointing at an object could imply that the object is selected. Note, however, that this technique does not extend to support transactions that require motion in State 2 (see below). An alternative approach, suitable for the touch tablet, is to use a pressure threshold crossing to signal the state change (Buxton, Hill, Rowley, 1985). This, however, requires a pressure sensing transducer.

The selection signal can also be generated via a timeout cue. That is, if I point at something and remain in that position for some interval Δt, then that object is deemed selected. The problem with this technique, however, is that the speed of interaction is limited by the requisite Δt interval.

Continuous Motion in State 2

Unlike *point and select,* most of the transactions employing State 2 require continuous motion in that state. These include:

- D*ragging:* as with icons;
- R*ubber-banding:* as with lines, windows, or sweeping out regions on the screen;
- P*ull-down menus;*
- I*nking:* as with painting or drawing;
- C*haracter recognition:* which may or may not leave an ink trail.

Consequently, two prerequisites of any supporting technology are:

- State 1 to/from State-2 transitions
- Ease of motion while maintaining State 2.

The first is more obvious than the second, and has been discussed in the previous section.

It is this more obscure second point which presents the biggest potential impediment to performance. For example, this paper is being written on a Macintosh Portable which uses a trackball. While pointing and selecting work reasonably well, this class of transaction does not. Even though both requisite states are accessible, maintaining continuous motion in State 2 requires holding down a space-bar like button with the thumb, while operating the trackball with

the fingers of the same hand. Consequently the hand is under tension, and the acuity of motion is seriously affected, compared to State 1, where the hand is in a relaxed state.

Direct Input Devices are Special

Direct input devices are devices where input takes place directly on the display surface. The two primary examples of this class of device are *light pens* and *touch screens.*

In terms of the model under discussion, these devices have an important property: in some cases (especially with touch screens), *the pointing device itself (stylus or finger) is the tracking "symbol."* What this means is that they "track" when out of range. In this usage, we would describe these devices as making transitions directly between State 0 and State 2, as illustrated in Figure 6.

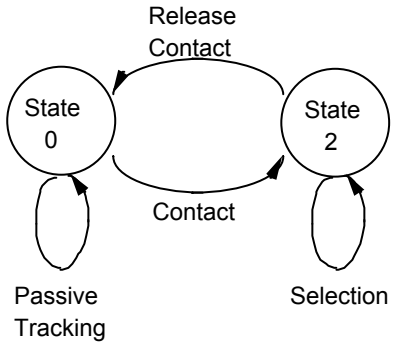


Figure 6: State 0-2 Transitions

With direct devices such as a light pen and touch screen, the pointing device (pen and finger, respectively, is the tracking mechanism. Hence, State 1 is bypassed. Since the tracking is passive (the system does not know what is being pointed at until contact), this tracking state should not be considered as State 1.

Another way that one might think of characterizing this would be as a simple State 1-2 transition, as shown in Figure 2. There are at least two reasons that this is not done, however. First, in this case the tracking is passive. The system has no sense of what the finger is pointing at until it comes into contact. This is a significant difference.

Second, there are examples where these same direct devices are used with an explicit State 1. For example, light pens generally employ an explicit tracking symbol. Touch screens can also be used in this way, as been shown by Potter, Shneiderman, and Weldon (1988), and Sears and Shneiderman (1989), among others. In these touch screen examples, the purpose was to improve pointing accuracy. Without going into the effectiveness of the technique, what is important is that this type of usage *converts the direct technology into a State 0-1 device*.

Consider the case of the touch screen for a moment. Choosing this approach means that the price paid for the increased accuracy is direct access to State-2 dependent transactions (such as selection and dragging). Anything beyond pointing (however accurately) requires special new procedures (as discussed above).

Aside Concerning the Human Factors Literature

We can gain a certain insight if we try to relate Table 1 to the human-factors literature that applies to input (as summarized in Milner, 1988 and Greenstein and Arnaut, 1988). Note that by far the two most common experimental tasks are *pursuit tracking* and *target acquisition.*

Both of these tasks are essentially state-1 motion tasks. When we look at the table, or consider where we actually spend our time in direct manipulation systems, these two tasks are not at all representative. Pursuit tracking is virtually never used except in games, aeronautics and military applications. It's relevance to mainstream usage is secondary at best.

Target acquisition is certainly relevant, but so are dragging, sweeping out regions and character recognition, for example.

What the model and the tables make clear is that there is a very large gap in the literature where State-2 motion tasks are concerned.

Summary and Conclusions

A state-transition model has been presented that captures many important aspects of input devices and techniques. As such, it provides a means of aiding the designer in evaluating the match between the two. While discussed in the context of well-known devices, the model can be applied to newer classes of transducers such as the VPL dataglove (Zimmerman, Lanier, Blanchard, Bryson & Harvill, 1987).

The model goes beyond that previously introduced by Buxton (1983) in that it deals with the continuous and discrete components of transducers in an integrated manner. However, it has some weaknesses. In particular, in its current form it does not cope well with representing transducers capable of pressure sensing on their surface or their buttons (for example, a stylus with a pressure sensitive tip switch used to control line thickness in a drawing program).

Despite these limitations, the model provides a useful conceptualization of some of the basic properties of input devices and interactive techniques. Further research is required to expand and refine it.

TO INTEGRATE

Lipscomb, J. & Pique, M. (1986). Analog input device physical characteristics. *SIGCHI Bulletin,* 25(3), 40-45.

Bleser, T. W. (1991) *An input device model of interactive systems design*. Doctor of Science Dissertation, Department of Electrical Engineering and Computer Science, The George Washington University.