Chapter 5:

CASE STUDY 1: TOUCH TABLETS

Introduction

Touch tablets are an interesting subject for a case study. On the one hand, they are simplicity personified. They are just a flat surface that can sense *that* it has been touched and communicate to the computer the location of *where* that touch occurred. On the other hand, they form the basis for an extremely broad and diverse set of physical and logical manifestations, as well as interaction techniques. Hence, they constitute a rich source for improving our understanding of input.

Figure 1: Integration of Touch Tablets into other devices

The two images show a touch tablet integrated into a keyboard and a mouse. (Photos: Cherry Electrical Products and Fujitsu Takamisawa America Inc., respectively)

In simplest terms, a touch tablet is typically mounted horizontally on a working surface and operated with a single finger. But from this basic configuration is a broad range of variations. They can range in size from an inch per side to several feet. They differ in how much pressure is required before a touch is registered. Some are capable of continuously reporting to the computer the amount of pressure being applied by the touch. Some are able to be operated with a stylus as well as a finger, while others are capable of independently sensing the location (and sometimes pressure) of multiple simultaneous points of touch. And, as is illustrated in Figure 1, they can be integrated into other devices such as a keyboards or mice.

The biggest problem in any discussion around touch tablets stems from confusing them with touch screens. The problem is legitimate since the differences between the two are not always a clear cut as one might first think. Both are controlled by touch. With touch screens, the touch technology is superimposed over a display. But what about the input device shown in Figure 2? It is technically a touch screen, since the touch sensor is over a display. On the other hand, it is more like a touch tablet, since it is not mounted on the primary visual display, and is horizontally mounted in a tablet-like fashion.

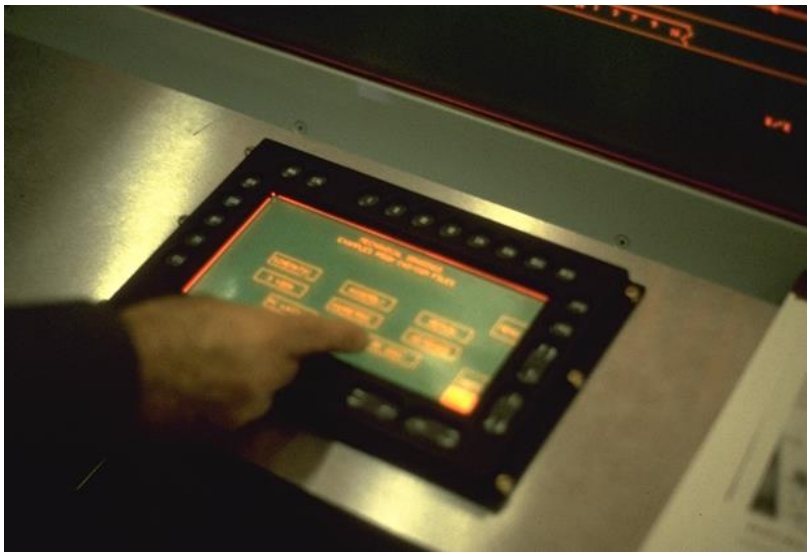


Figure 2: A Horizontally Mounted Touch Screen

When a touch screen is mounted over horizontally mounted LCD display, the distinction between touch tablet and touch screen starts to blur. In this context, most of the attributes of a touch tablet, as discussed below, apply. (Photo: by author)

As discussed already in *Chapter 3 Alternative Perspectives*, here is another example where categorizing devices according to technology is of limited value. In the rest of this chapter, we will dig into a number of issues, techniques and applications of touch tablets. Those interested in digging further into the literature are referred to Arnault & Greenstein (1986), Becker & Greenstein (1988), Buxton, Hill & Rowley (1985), and MacKenzie, & Oniszczak, (1998). Those interested in the touch screen literature are referred to: to Herot and Weinsapel (1978), Nakatani and Rohrlich (1983), Minsky (1984), Harrison and Hudson (2012), and Heo & Lee (2011).

Properties of Touch Sensitive Tablets

Asking "which input device is best?" is much like asking "how long is a piece of string?" It depends. The trick is knowing on what. With input devices, making informed choices depends on our understanding of the relationship between device properties and the demands of specific applications, users, and contexts. We will investigate touch tablets from the perspective of improving our understanding of these relationships. The objective is not just to shed light on touch tablets, *per se,* but to demonstrate the kind of analysis that should be undertaken with any technology under consideration.

Gettng to it, touch tablets have a number of properties that distinguish them from other devices:

- They have no mechanical intermediate device (such as a stylus or a puck) between the hand and the sensor. Hence they are useful in environments (such as classrooms and public access terminals) where such intermediate devices can get damaged, lost or stolen.
- Having no puck to slide or get bumped, the tracking symbol always stays put once placed, thus making them well suited for pointing tasks in environments subject to vibrations or motions (eg. factories and cockpits).
- Unlike joysticks and trackballs, touch tablets have a very low profile that can be integrated into other equipment such as desks and low profile keyboards, as illustrated in Figure 1. This has potential benefits in portable systems, and according to the Keystroke Model of Card, Newell and Moran (1980) reduces homing time from the keyboard to the pointing device.
- They can be molded into one-piece constructions thus cracks and grooves where dirt can collect. This makes touch tablets especially suitable for environments which must be kept very clean such as hospitals or environments which are very dirty such as factories. (See Figure 3.)
- Due to their simple construction which involves no moving parts, touch tablets generally have reliable and long-lived operation making them especially suitable to environments where they will be subjected to intense use or where reliability is critical.

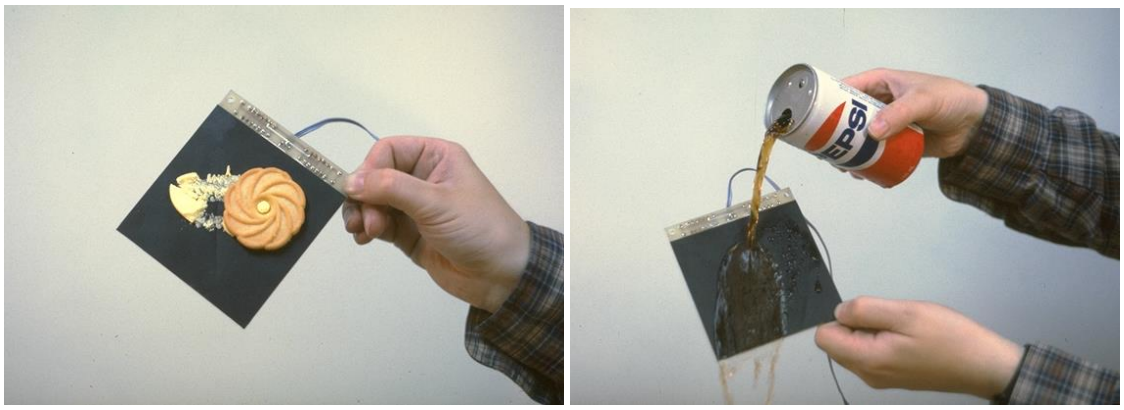


Figure 3: Suitability of Touch Tablets in Very Clean and Dirty Environments

As the two figures illustrate, the lack of moving parts and places for dirt to accumulate make touch tablets well suited for environments that are very dirty or which must be kept very clean.

- They afford the overlaying of physical templates, such as those shown in Figure 4. These can provide tactile feedback as to where one is touching, much like the cracks between the keys on a piano or the frets of a guitar. Hence, they have the potential to afford “heads up”, or “touch typing” on virtual devices defined on the tablet’s surface.

- Like all graphics tablets, they can be used in either relative or absolute mode. That is, they can report the absolute coordinates of where they are being touched, or relative motion, that is the direction and distance of change. If the surface is partitioned into regions, such as with a template, different regions may work in different ways.

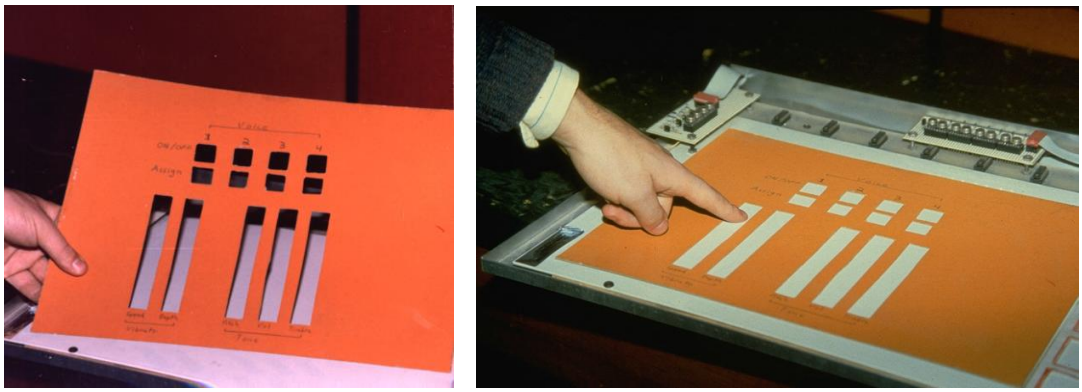


Figure 4: Templates Overlays as Guides Defining Virtual Devices

- They present no inherent mechanical or kinesthetic constraints on their ability to sense multiple simultaneous touches. That is, the tablet itself does not inherently prevent me from touching it with multiple fingers from one or both hands. Nor does it prevent me from applying different degrees of pressure at each location being touched, or touching it with different materials or things of different sizes or shape.

This last point brings up an important issue: the difference between what the tablet mechanically affords and what the human can do, *versus* what the tablet can actually sense. For example,

- **Multi-touch**: Just because a tablet lets me touch it with multiple fingers does not meant that it can sense the location of each point of contact.

- **Multi-material:** Just because it can sense my finger does not mean that it can sense my finger when wearing a glove, or sense a wooden stylus.

- **Pressure**: Just because it can sense the location of one or more points of contact does not mean that it can sense the pressure being applied at any such point.

- **Area**: Just because it is being touched at a location centred on a particular point on its surface does not mean that it can sense the area of contact.

- **Shape**: Just because it senses that it is being in a certain area, does not mean that it can sense the shape of that area adequately to support shape recognition.

The ability of a particular technology in terms of considerations such as these will have a large impact on the type of interaction that can be supported. For example, despite the

capacity for multi-touch, nevertheless, some of the most popular mobile phones are incapable of doing what a first generation PalmPilot could do, namely, let you use your finger or a stylus. That technology decision means that regardless of what software the device runs, it will be incapable of serving as a hand-held scratch pad on which you can make quick sketches or hand-written notes. Likewise, if a larger surface cannot sense the shape and/or size of the area of contact, it will be highly unlikely that any software will be able to do palm rejection, i.e., distinguish between the tip of the stylus, being used for writing/drawing, and the palm of the hand that is holding the stylus, when it inadvertently rests on the writing surface, as is typically the case with pencil and paper. (Note that this problem does not occur with tablet computers or graphics tablets, since they do not use touch-sensing to detect the location or pressure of the stylus. While they use a stylus and may be the same size as a stylus-driven touch screen, they are very different in how they work, and those differences have a significant impact on what types of interaction can be supported, and how easily they can be done – if at all.

Sidebar to come

Summarize key technologies (resistive, capactive, optical, …) and what they do and do not generally support.

In order to explore the practical implications of some of these considerations in more detail, and to demonstrate the use of touch tablets, we will now work through some examples based on a toy paint system. In the process, we make use of the *3-State Model* introduced in *Chapter 4.* Remember, however, that our purpose is not to show how to implement a paint program. A paint system is simple a common and easily understood application, and therefore a useful vehicle for discussing interaction techniques that use touch tablets within the context of an application. Much of this derives from Buxton, Hill and Rowley (1985).

Example 1: Painting Without Pressure Sensing

The example paint program allows the creation of simple finger paintings. The layout of the main display for the program is shown in Figure 5. On the left is a large drawing area where the user can draw simple free-hand figures. On the right is a set of menu items. Most are “paint pots” that are used to select paint colour. The lowest menu item selects a colour mixing tool, discussed in detail later.

Figure 5: Main display for paint program.

Input to the program is via the 8 cm x 8 cm touch shown in Figure 6.

Figure 6: The 8 cm x 8cm Touch Tablet used in the Example Paint Program

Most commercial touch tablets have only 1-bit (binary on/off touch) of pressure sensing. Consequently, they are a 2-state device, as characterized by the *3-State Model* described in Chapter 4.

Typically, selection and manipulation take place in State 1, since that is when the finger is in contact with the tablet. State 0-1 and 1-0 transitions are signaled by the finger coming into or out of contact with the touch tablet, respectively. But with touch tablets, in State-0 there is no accurate way to know where your finger is relative to the screen. So how can one know if their finger is over a particular menu item, or in the location where they want to paint? With a touch screen, the screen and touch sensor are superimposed, so the problem doesn't exist. (This is one of the main differences between a touch tablet and a touch screen.) In order to use our touch tablet to paint, we need to sort this out.

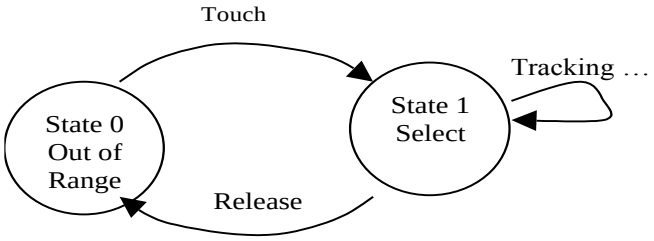


Figure 7: State diagram for drawing portion of simple paint program.

There are a few approaches to handling the situation:

- *Graphics on the tablet:* Can be used to delimit specific regions. Effectively, this is a degenerate case of a touch display, where the display on the tablet is static rather than dynamic. Since the tablet graphics are static, this is generally only of value if the regions are likewise unchanged throughout the application. This can help in selecting menu items and interacting with widgets. However, it still will not help in painting, where one wants to be precise in placing the brush.

- *Drag cursor in State 1 / Select item on release:* On contact, a cursor tracks the finger position on the tablet. When the finger releases from the tablet, the item currently under the cursor is selected. This approach can work well for selecting objects. However, it does not help in painting, or in dragging objects, unless some mechanism like a double tap is used. This is typically inaccurate and unsatisfactory.

- *Use a secondary mechanism:* A secondary device or mechanism, (such as a keyboard button, a separate button beside the touch tablet, dwell time over a position, or the double tap mentioned in the previous example) to signify the onset of painting, or that the object currently under the cursor is to be dragged. This is awkward and requires practice to develop the coordination needed to make small rapid strokes in the painting. It is also inefficient and either uses two hands where one could (and normally should) do, or, places more burden and stress on the hand which is doing the painting.

Virtually all laptop computers equipped with a touch tablet (such as the *Dell Latitude* shown in Figure 8) come with the kind of tablet described in this example. Hence, they provide a convenient object for study. Yes you can select objects and menu items, drag objects and paint. But in order to do so, you must use one of the mechanisms described above. See for yourself how it feels.

Picture of Painting on

Dell Laptop.

To come

Figure 8: Painting Using the Touch Tablet on a Laptop

Touching the tablet does not initiate painting; rather it allows me to position and move the cursor. Paint is laid down by moving the cursor while holding down one of the "mouse" buttons beside the touch pad. You can see how awkward this is by trying the same thing on virtually any laptop equipped with a touch tablet.

Example 2: Painting with Two levels of Pressure

This version of the program uses a tablet that reports two levels of contact pressure to provide a satisfactory solution to the signaling problem. A low pressure level (a light touch by the user) is used for general tracking. A heavier touch is used to make menu selections, or to enable painting (see Figure 9 for the tablet states used to control painting with this program). The two levels of contact pressure allow us to make a simple but practical one finger paint program.

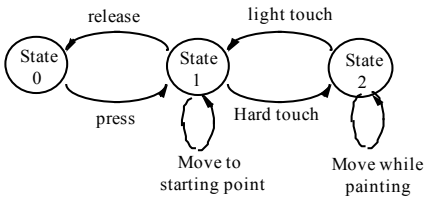


Figure 9: State diagram for painting portion of simple paint

program using pressure sensing touch tablet.

This version is very much like using the one button mouse on the Apple Macintosh with MacPaint (Williams, 1984). Thus, a simple touch tablet is not very useful, but one that reports two levels of pressure is similar in power (but not feel or applicability) to a one button mouse.[1]

[1] Also, there is the problem of friction, to be discussed below under "Inherent Problems".

Example 3: Painting with Continuous Pressure Sensing

In the previous demonstrations, we have only implemented interaction techniques that are common using existing technology. We now introduce a technique that provides functionality beyond that obtainable using most conventional input technologies.

In this technique, we utilize a tablet's capability to sensing a continuous range of touch pressure. With this additional signal, the user can control both the width of the paint trail and its path, using only one finger. The new signal, pressure, is used to control width. This is a technique that cannot be used with any mouse that we are aware of, and to our knowledge, is available on only one conventional tablet (the GTCO *Digipad* with pressure pen (GTCC 1982)).

We have found that using current pressure sensing tablets, the user can accurately supply two to three bits of pressure information, after about 15 minutes practice. This is sufficient for simple doodling and many other applications, but improved pressure resolution is required for high quality painting.

Touch, Pressure and Friction

In using touch technologies, we run into a problem when we want to go beyond simple pointing and selection tasks - especially if we are using our finger rather than a stylus. This comes up, for example, with tasks such as *dragging* or *inking*, which involve motion across the touch surface and which are common to direct manipulation systems.

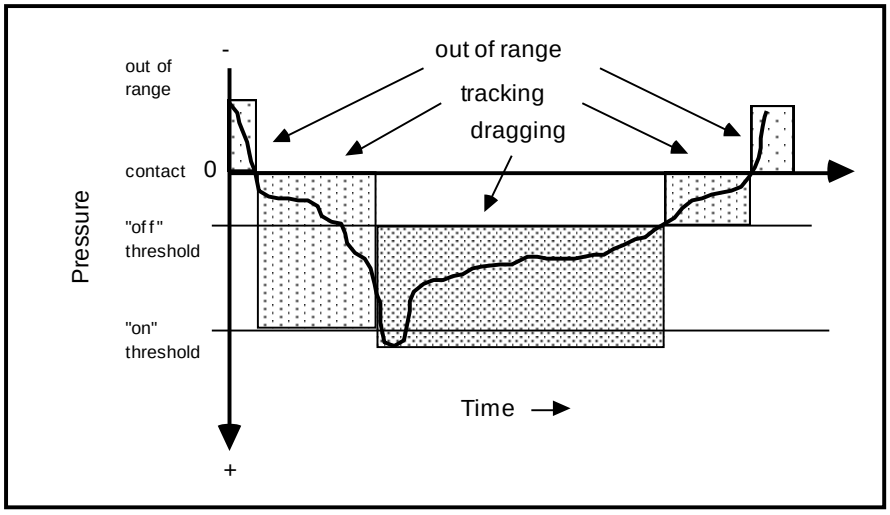


Figure 10: Dual threshold approach to reduce friction during dragging.

Dragging is initiated through relatively hard application of pressure. However, to avoid friction that would inhibit dragging (or inking), the "end of dragging" signal is initiated by crossing a much lighter pressure threshold. The approach illustrated assumes a technology

The problem in question has to do with the pressure thresholds at which events occur. In the simplest case, we have two conflicting demands to consider:

- To reliably sense contact implies having to cross a reasonably *high* pressure threshold in order to instigate an event.
- To reliably drag or ink implies a *low* amount of pressure so that the coefficient of friction does not interfere with the freedom of motion of the finger. The threshold signaling the end of the dragging or inking must be extremely light to avoid unintentional termination of the transaction.
- One approach to meeting both of these criteria is to utilize a different pressure threshold for each of the initiation and the termination of the transaction. This is illustrated in Figure 10, where these are labeled the *on* and *off* thresholds, respectively.
- The figure graphs pressure over time during a hypothetical dragging transaction. In the example illustrated, the user goes through three states:
  - *State 0: Out of range:* The initial and final state, where the finger is out of contact with the device.
  - *State 1: Tracking:* An intermediate state where the finger is in contact with the touch surface, its position is being sensed, but no event other than tracking is initiated. This is equivalent to the "normal" state of a mouse without any of its buttons depressed.
  - *Stare 2: Dragging:* The state in which the actual dragging event is undertaken.

The key point to notice in both previous figures is that the pressure threshold which initiates the dragging state (the "on" threshold) is higher pressure than the threshold which terminates the dragging (the "off" threshold).

The appropriate setting for these thresholds can only be determined through actual user testing. They depend on the technology used, the context, and the user population. It may be, for example, that it is best to set the "off" threshold to be the same as the "out of range" threshold, since using touch it is generally difficult to make the transition from the dragging state to the tracking state reliably. This is illustrated in Figure 11.

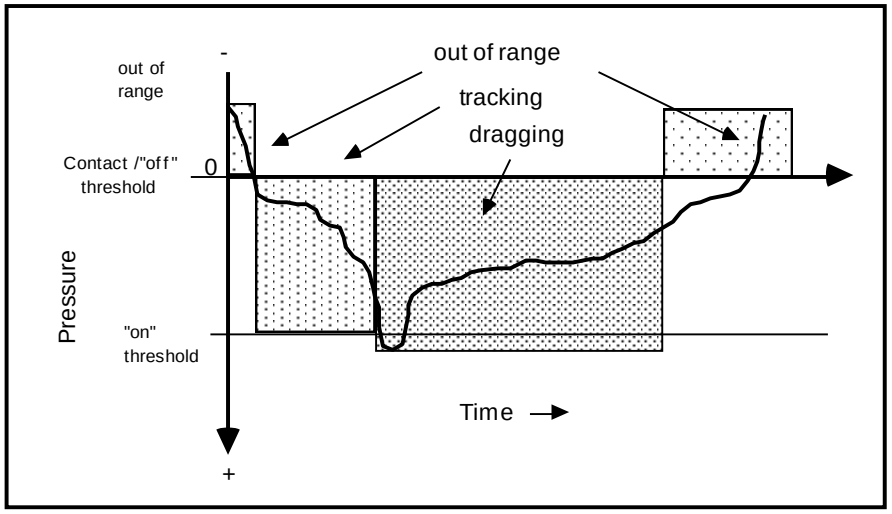


Figure 11: Eliminating the intermediate (tracking) state on termination

The "off" threshold and the "contact" threshold may be identical, thereby helping avoid ambiguities at the end of the event.

Using transitions across pressure to initiate state changes requires a touch technology that can sense more than one bit of pressure. Technology that can make more than simple touch/no touch discriminations must be used. On the order of 4 bits (sixteen levels) of continuous pressure should be able to be sensed to use these techniques effectively. This is important to be aware of, since most commercial touch technologies are binary only.

The limitations of binary touch sensing are more serious with touch tablets than touch screens. As described in Chapter 4 in the discussion of the *3-State Model,* with a touch screen, one can often combine the State-0 “out of Range” state with the State-1 "tracking" state. This is illustrated in Figure 12.

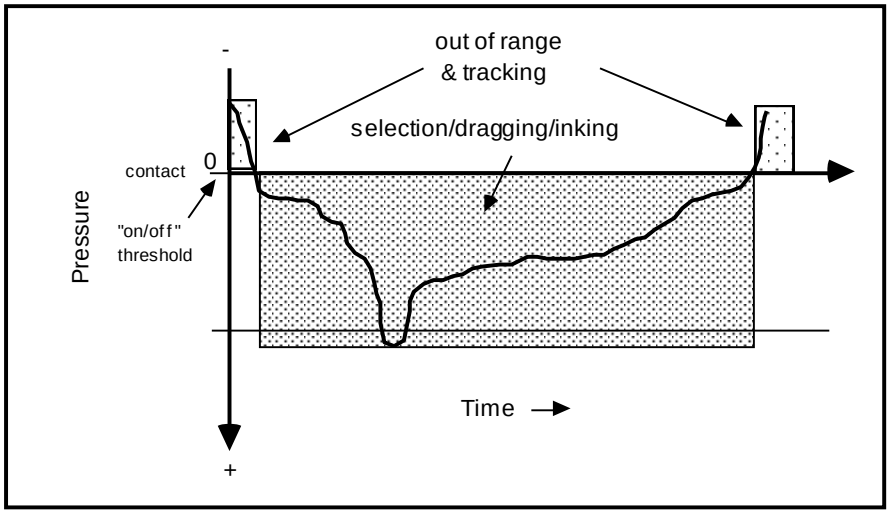


Figure 12: Touch screen with binary pressure sensing.

The "out of range" finger is the tracking mechanism. The system itself provides no feedback as to location until contact. Selection, dragging or inking tasks, for example, are all initiated immediately upon coming into contact with the screen. Note that the out of range finger provides the tracking information only if the control and display surface are one and the same. With normal touch tablets, for example, this will not work since the tablet is a different surface than the display, so no accurate feedback as to position is provided until the finger come into contact with the tablet surface.

The ability to track position while out of range is due to the fact that the display surface and the touch surface are the superimposed. Consequently, the out of range finger, itself, serves as the tracking symbol. (This is also true with touch tablets in the special case where, for example, a visual menu or some other target - such as a template - is mounted on the tablet surface.)

Note, however, that this two-state approach does not permit any intermediate feedback to confirm that the correct object is selected.

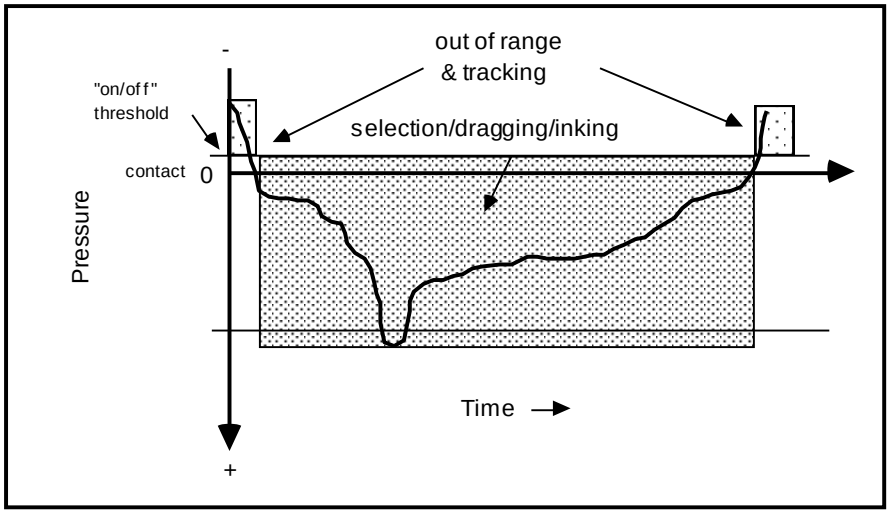


Figure 13: The "chess player's syndrome"

In this case, the system senses the finger before the user comes into physical contact with the touch surface. This is relatively common with infrared touch technologies when used with a CRT. Because of screen curvature there is a gap between the light beam and the screen surface. The finger is sensed when it breaks the light beam, rather than when it comes into contact with the surface. The effect is analogous to having one's chess partner insist that you make a particular move, when you claim that you didn't touch the piece in question. Notice that the problem exists for both event initiation and termination.

With normal touch tablets, the display surface and the control surface are different. As long as this is true, it is virtually impossible to provide any precise feedback as to position before coming into contact with the tablet. Hence, a binary touch tablet only supports the *out of range* and *tracking* states (in and out of contact, respectively). In order to support transactions such as inking or dragging, some other mechanism must be introduced.

Finally, illustrates what we call the "chess players syndrome." As shown in Figure 13, with some technologies, a "touch" is sensed by a mechanism other than contact with the display surface. (A common example is with some infrared touch screens mounted on CRTs, especially near the bezel of the display.)

In this case, one is committed to the action before coming into physical contact with the touch surface. The effect is similar to being committed to a chess move by one's partner, who claims that you touched a particular piece (and you know this not to be the case). This problem can affect all touch-activated transactions including selection, button pushes, dragging and inking.

Behavioural data:

MacKenzie and Oniszczak, A. (1998) conducted a study comparing three methods of implementing the select operation on touchpads:

- *physical button:* where one pointed at the target using the touchpad and then pushed a separate button to effect the selection

- *"lift-and-tap":* where, after pointing at the target using the touchpad, one tapped on the tablet surface with the pointing finger

- *finger pressure with tactile feedback:* where after pointing at the target with the finger on the touchpad, one pushed down on the pad. Crossing a predefined pressure threshold initiated the selection event.

In an empirical test with 12 participants, the tactile condition was 20% faster than lift-and-tap and 46% faster than using a button for selection. Error rates were higher with the tactile condition, however. These were attributed to limitations in the prototype, such as the use of a capacitive-sensing touchpad and poor mechanical design. In a questionnaire, participants indicated a preference for the tactile condition over the other two conditions.

In the study, the *finger pressure with tactile feedback* technique used was similar to the technique employed by Buxton, Hill & Rowley (1985), as described in the toy paint program above, and in our discussion of issues concerning pressure sensing. The only significant difference was that, as implemented by MacKenzie and Oniszcak, there was tactile (by virtue of a relay) and aural feedback (by way of an audible click) when crossing the pressure threshold. It remains for a future study to investigate how much of the performance improvement and preference was due to the feedback *vs* the time-motion and gestural aspects of the technique.

"Windows" on the Tablet: Colour Selection

We now demonstrate how the surface of the touch tablet can be dynamically partitioned into "windows" onto virtual input devices. We use the same basic techniques as discussed under templates (above), but show how to use them without templates. We do this in the context of a colour selection module for our paint program. This module introduces a new display, shown in Figure 14.

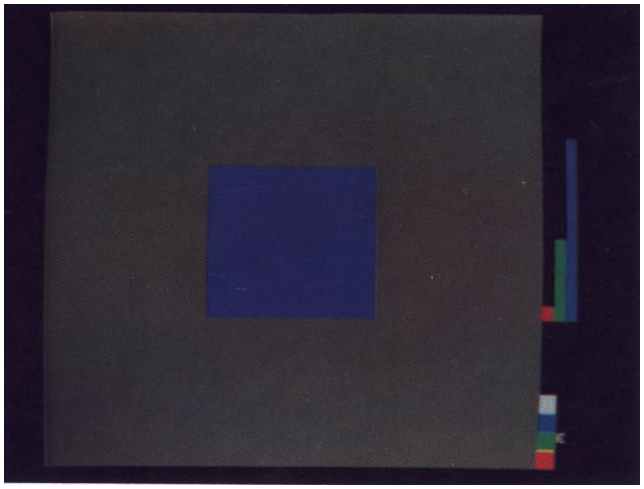


Figure 14: Colour mixing display.

In this display, the large left side consists of a colour patch surrounded by a neutral grey border. This is the patch of colour the user is working on. The right side of the display contains three bar graphs with two light buttons underneath. The primary function of the bar graphs is to provide feedback, representing relative proportions of red, green and blue

in the colour patch. Along with the light buttons below, they also serve to remind the user of the current layout of the touch tablet.

In this module, the touch tablet is used as a “virtual operating console”. Its layout is shown (to scale) in Figure 15. There are 3 valuators (corresponding to the bar graphs on the screen) used to control colour, and two buttons: one, on the right, to bring up a pop-up menu used to select the colour to be modified, and another, on the left, to exit.

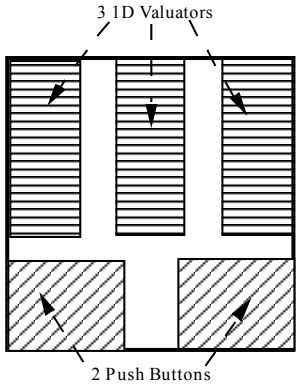


Figure 15: Layout of virtual devices on 8 cm x 8 cm touch tablet.

The single most important point to be made in this example is that a single physical device is being used to implement 5 virtual devices (3 valuators and 2 buttons). This is analogous to the use of a display window system, in its goals, and its implementation.

The second main point is that there is nothing on the tablet to delimit the regions. This differs from the use of physical templates as previously discussed, and shows how, in the absence of the need for a physical template, we can instantly change the “windows” on the tablet, without sacrificing the ability to touch type.

We have found that when the tablet surface is small, and the portioning of the surfaces is not too complex, the users very quickly (typically in one or two minutes) learn the positions of the virtual devices relative to the edges of the tablet. More importantly, they can use the virtual devices, practically error free, without diverting attention from the display. (We have repeatedly observed this behaviour in the use of an application that uses a 10 cm square tablet that is divided into 3 sliders with a single button across the top).

Because no template is needed, there is no need for the user to pause to change a template when entering the colour mixing module. Also, at no point is the user's attention diverted from the display. These advantages cannot be achieved with any other device we know of, without consuming display real estate.

The colour of the colour patch is manipulated by draggng the red, green and blue values up and down with the valuators on the touch tablet. The valuators are implemented in relative mode (i.e., they are sensitive to changes in position, not absolute position), and are manipulated like one dimensional mice. For example, to make the patch more red, the user presses near the left side of the tablet, about half way to the top, and slides the finger up (see Figure 16). For larger changes, the device can be repeatedly stroked (much like stroking a mouse). Feedback is provided by changing the level in the bar graph on the screen and the colour

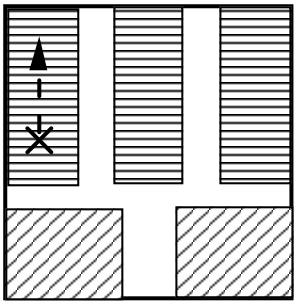
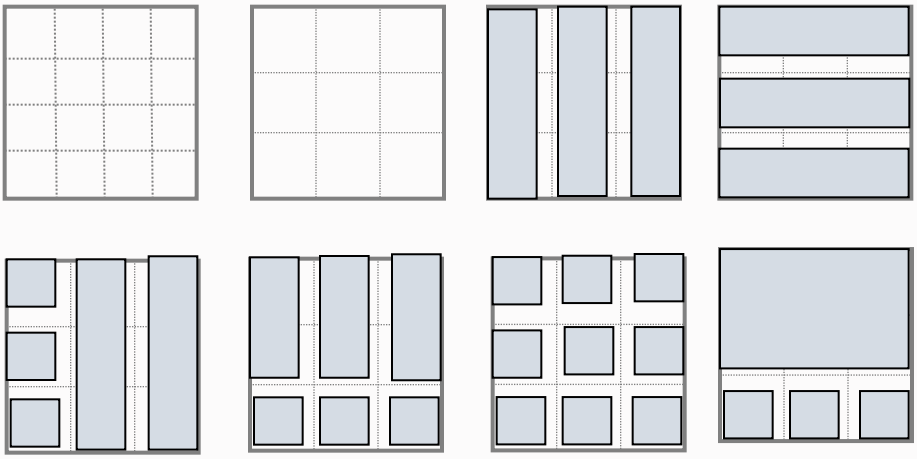


Figure 16: Increasing red content, by pressing on red valuator and sliding up.

In the lead-up to this work we did some informal studies which are easily duplicated. We tested what the upper bounds were in terms of one's ability to hit, eyes free, unmarked buttons or regions on a 10x10 cm touch tablet. We compared widgets that fit onto a uniform 4 x 4 and 3 x 3 grid, as per the two un-shaded grids below.



Not surprisingly, there were much better results on the 3x3 grid. Again, not surprisingly, that remained true when we merged adjacent tiles. Examples of mapping virtual widgets onto the 3x3 grid, much like we did in the touch-tablet paint study, are illustrated in the grids with the shaded areas. If you want to get a sense of your ability to operate such virtual controls, just think about playing X's & O's (Knots and Crosses) on a similarly sized grid, and see how well you can hit the desired tile when the palm has a good reference place to rest relative to the grid.

Furthermore, in an appendix of the first edition of their classic book on interactive graphics, Newman and Sproul (1973) describe in detail how a simple trainable printed character recognizer can be implemented using this same 3x3 grid. The recognizer (developed by H.W. Ledeen in 1967) codes characters in terms of what grid lines are crossed, and in which sequence, during the printing of the symbol. Hence, such a touch tablet can be used for alphanumeric text entry, as well as recognize symbolic commands for which it has been trained. Building this recognizer is an excellent exercise for the reader with basic programming skills.

Using a mouse, the above interaction could be approximated by placing the tracking symbol be displayed at all. They are only a convenience to the user. There are interfaces where, in the interests of maximizing available display area, there will be no items on the display analogous to these bars. That is, there would be nothing on the display to support an interaction technique that allows values to be manipulated by a mouse.

Finally, we can take the example one step further by introducing the use of a touch tablet that can sense multiple points of contact (e.g., Metha,1982; Lee, Buxton and Smith,1985). With this technology, all three colour values could be changed at the same time (for example, fading to black by drawing all three sliders down together with three fingers of one hand). This simultaneous adjustment of colours could not be supported by a mouse, nor any single commercially available input device we know of. Controlling several valuators with one hand is common in many operating consoles, for example: studio light control, audio mixers, and throttles for multi-engine vehicles (e.g., aircraft and boats). Hence, this example demonstrates a cost effective method for providing functionality that is currently unavailable (or available only at great cost, in the form of a custom fabricated console over the bars of colour, and dragging them up or down. However, if the bars are narrow, this takes visual acuity and concentration that distracts attention from the primary task - monitoring the colour of the patch. Furthermore, note that the touch tablet implementation does not need the bars to), but has wide applicability.

Summary of Examples

Through these simple examples, we have demonstrated several things:

- The ability to sense at least two levels of pressure is a virtual necessity for touch tablets, as without it, auxiliary devices must be used for signaling, and “direct manipulation” interfaces cannot be effectively supported.
- The extension to continuous pressure sensing opens up new possibilities in human-computer interaction.
- Touch tablets are superior to mice and tablets when many simple devices are to be simulated. This is because: (a) there is no need for a mechanical intermediary between the fingers and the tablet surface, (b) they allow the use of templates (including the edges of the tablet, which is a trivial but useful template), and (c) there is no need for positional feedback that would consume valuable display space.
- The ability to sense multiple points of contact radically changes the way in which users may interact with the system. The concept of multiple points of contact does not exist for, nor is it applicable to, current commercially available mice and tablets.

Physical Templates and Windows on Tablets

- flip keyboard

Inherent Problem with Touch Tablets:

A problem with touch tablets that is annoying in the long term is friction between the user's finger and the tablet surface. This can be a particularly severe problem if a pressure sensitive tablet is used, and the user must make long motions at high pressure. This problem can be alleviated by careful selection of materials and care in the fabrication and calibration of the tablet.[2] Also, the user interface can be designed to avoid extended periods of high pressure.

Perhaps the most difficult problem is providing good feedback to the user when using touch tablets. For example, if a set of push-on/push-off buttons are being simulated, the traditional forms of feedback (illuminated buttons or different button heights) cannot be used. Also, buttons and other controls implemented on touch tablets lack the kinesthetic feel associated with real switches and knobs. As a result, users must be more attentive to visual and audio feedback, and interface designers must be freer in providing this feedback. (As an example of how this might be encouraged, the input “window manager” could automatically provide audible clicks as feedback for button presses).

Potential Enhancements to Touch Tablets (and other devices)

The first problem that one notices when using touch tablets is “jitter” when the finger is removed from the tablet. That is, the last few locations reported by the tablet, before it senses loss of contact, tend to be very unreliable.

This problem can be eliminated by modifying the firmware of the touch tablet controller so that it keeps a short FIFO queue of the samples that have most recently be sent to the host. When the user releases pressure, the oldest sample is retransmitted, and the queue is emptied. The length of the queue depends on the properties of the touch tablet (e.g., sensitivity, sampling rate). We have found that determining a suitable value requires only a few minutes of experimentation.

A related problem with most current tablet controllers (not just touch tablets) is that they do not inform the host computer when the user has ceased pressing on the tablet (or moved the puck out of range). This information is essential to the development of certain types of interfaces. (As already mentioned, this signal is not available from mice). Currently, one is reduced to deducing this event by timing the interval between samples sent by the tablet. Since the tablet controller can easily determine when pressure is removed (and must if it is to apply a de-jittering algorithm as above), it should share this information with the host.

Clearly, pressure sensing is an area open to development. Two pressure sensitive tablets have been developed at the University of Toronto (Sasaki, et al. 1981; Lee, et al. 1905). One has been used to develop several experimental interfaces and was found to be a very powerful tool. They have recently become available from Elographics and Big Briar. Pressure sensing is not only for touch tablets. Mice, tablet pucks and styli could all benefit by augmenting switches with strain gauges, or other pressure sensing instruments. GTCO, for example, manufactures a stylus with a pressure sensing tip (GTCO 1982), and this, like our pressure sensing touch tablets, has proven very useful.

---

[2] As a bad example, one commercial “touch” tablet requires so much pressure for reliable sensing that the finger cannot be smoothly dragged across the surface. Instead, a wooden or plastic stylus must be used, thus loosing many of the advantages of touch sensing.

Conclusions

We have shown that there are environments for which some devices are better adapted than others. In particular, touch tablets have advantages in many hostile environments. For this reason, we suggest that there are environments and applications where touch tablets may be the most appropriate input technology.

This being the case, we have enumerated three major distinctions between touch tablets and one button mice (although similar distinctions exist for multi-button mice and conventional tablets). These assist in identifying environments and applications where touch tablets would be most appropriate. These distinctions concern:

- limitation in the ability to signal events,
- suitability for multiple point sensing, and
- the applicability of tactile templates.

These distinctions have been reinforced, and some suggestions on how touch tablets may be used have been given, by discussing a simple user interface. From this example, and the discussion of the distinctions, we have identified some enhancements that can be made to touch tablets and other input devices. The most important of these are pressure sensing and the ability to sense multiple points of contact.

We hope that this paper motivates interface designers to consider the use of touch tablets and shows some ways to use them effectively. Also, we hope it encourages designers and manufacturers of input devices to develop and market input devices with the enhancements that we have discussed.

The challenge for the future is to develop touch tablets that sense continuous pressure at multiple points of contact and incorporate them in practical interfaces. We believe that we have shown that this is worthwhile and have shown some practical ways to use touch tablets. However, interface designers must still do a great deal of work to determine where a mouse is better than a touch tablet and *vice versa.*

Finally, we have illustrated, by example, an approach to the study of input devices, summarized by the credo: “Know the interactions a device is intended to participate in, and the strengths and weaknesses of the device.” This approach stresses that there is no such thing as a “good input device,” only good interaction task/device combinations.

TO DO

Add material on my navigation modes, including sliding along edge in navigation for orthogonal ctl of 1 D in relative motion.

Albinsson, P. & Zhai, S. (2003). High Precision Touch Screen Interaction. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'93*), 105-112.

Arnault, L. Y., & Greenstein, J. S. (1986). Optimizing the touch tablet: The effects of control-display gain and method of cursor control. Human Factors, 28(6), 717-726.

Becker, J. A., & Greenstein, J. S. (1988). Optimizing the touch tablet: The effects of lead-lag compensation and tablet size (NOSC-TD-1212): San Diego, CA: Naval Ocean Systems Center.

Beringer, D.B. & Peterson, J.G. (1985). Underlying behavioural parameters of the operation of touch-input devices: biases, models and feedback, *Human Factors* 27(4), 445-458.

Boie, R.A. (1984). Capacitive Impedance Readout Tactile Image Sensor, *IEEE International Confonference on Robotics and Automation*, 370-378.

Diamond Touch:

Dietz, P.H.& Leigh, D.L. (2001). DiamondTouch: A Multi-User Touch Technology, *ACM Symposium on User Interface Software and Technology (UIST '01)*, 219-226.

Harmon, L. D. 1980. Touch-sensing technology: A review. Tech. rep. MSR80-03. SME Technical Report, Society of Manufacturing Engineers, Dearborn, MI.

Harmon, L. D. 1982. Automated tactile sensing. *Int. J. Robotics Research* 1(2):3–31.

Harrison, C., Sato, M., & Poupyrev, I. (2012). Capacitive Fingerprinting: Exploring User Differentiation by Sensing Electrical Properties of the Human Body. *Proceedings of the 25th Annual ACM Symposium on User interface Software and Technology, UIST'12*, 537-543.

Lee, Mark (2000), Tactile Sensing: New Directions, New Challenges. *The International Journal of Robotixs Research.* 19(7), 636-643.

Metha, Nimish (1982), "A Flexible Machine Interface", M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto.

Parng, A. K. (1988). Automated test of Fitts' law and effects of target width and control/display gain using a digitizer tablet (NOSC-TD-1214): San Diego: Naval Ocean Systems Center.

- 2-hand navigation/selection: include discussion of absolute vs relative
- windows on tablets + templates + keyboard repeat absolute vs relative
- Video clips: Buxton, Hill & Rowley

Potter, R., Shneiderman, B. & Weldon, L. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'88)*, 27-32.

Ringel, M., Ryall, K., Shen, C., Forlines, C. & Vernier, F.(2004). Release, Relocate, Reorient, Resize: Fluid Techniques for Document Sharing on Multi-User Interactive

Tables. Extended Abstracts of the 2004 *ACM Conference on Human Factors in Computing Systems (CHI'04),* 1441-1444.

Rosenberg, I., Grau, A., Hendee, C., Awad, N. & Perlin, K. (2009). IMPAD – An Inexpensive Multi-Touch Pressure Acquisition Device. To Appear.

Sears, A. and B. Shneiderman (1991). High Precision Touchscreens: Design Strategies and Comparison with a Mouse*. International Journal of Man-Machine Studies*, 34(4): p. 593-613.

Shen, C., Everitt, K.M. & Ryall, K.(2003). UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces. Technote. *ACM International Conference on Ubiquitous Computing (UbiComp '03)*, 281-288.

Shen, C., Lesh, N.B., Moghaddam, B., Beardsley, P.A. & Bardsley, R.S.(2001). Personal Digital Historian: User Interface Design. *ACM Conference on Human Factors in Computing Systems (CHI '01)*, 29-30.

Shen, C., Lesh, N. & Vernier, F.(2003). Personal Digital Historian: Story Sharing Around the Table. *ACM Interactions*, 10(2), 15-22.

Shen, C., Lesh, N.B., Vernier, F., Forlines, C. & Frost, J. (2002). Sharing and Building Digital Group Histories. ACM Conference on Computer Supported Cooperative Work (CSCW '02), 324-333..

Shen, C., Vernier, F.D., Forlines, C. & Ringel, M., DiamondSpin (2004). An Extensible Toolkit for Around-the-Table Interaction. *ACM Conference on Human Factors in Computing Systems (CHI '04)*, 167-174.

Vernier, F., Lesh, N.B., Shen, C.(2002). Visualization Techniques for Circular Tabletop Interfaces. ACM Advanced Visual Interfaces (AVI '02), 257-263.

Young, G. (1989). *The Sackbutt Blues: Hugh Le Caine – Pioneer in Electronic Music.* Ottawa: National Museum of Science and Technology.

Zhang, Hong, So, Eric. & Guan, Yi-sheng (1999). Sensing contact with analog resistive technology. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC '99,* vol. 2, 806-811.

Zhang, Hong & So, Eric. (2002). Hybrid Resistive Tactile Sensing. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*. 32(1), 57 – 65.