Touch-Sensing Input Devices

Ken Hinckley and Mike Sinclair
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh, sinclair}@microsoft.com; Tel: +1-425-703-9065

ABSTRACT
We can touch things, and our senses tell us when our hands are touching something. But most computer input devices cannot detect when the user touches or releases the device or some portion of the device. Thus, adding touch sensors to input devices offers many possibilities for novel interaction techniques. We demonstrate the *TouchTrackball* and the *Scrolling TouchMouse*, which use unobtrusive capacitance sensors to detect contact from the user's hand without requiring pressure or mechanical actuation of a switch. We further demonstrate how the capabilities of these devices can be matched to an implicit interaction technique, the *On-Demand Interface*, which uses the passive information captured by touch sensors to fade in or fade out portions of a display depending on what the user is doing; a second technique uses explicit, intentional interaction with touch sensors for enhanced scrolling. We present our new devices in the context of a simple taxonomy of tactile input technologies. Finally, we discuss the properties of touch-sensing as an input channel in general.

Keywords
input devices, interaction techniques, sensor technologies, haptic input, tactile input, touch-sensing devices.

INTRODUCTION
The sense of touch is an important human sensory channel. In the present context, we use the term *touch* quite narrowly to refer to the cutaneous sense, or *tactile perception* [16]. During interaction with physical objects, pets or other human beings, touch (physical contact) constitutes an extremely significant event. Yet computer input devices, for the most part, are indifferent to human contact in the sense that making physical contact, maintaining contact, or breaking contact provokes no reaction whatsoever from most software. As such, touch-sensing input devices offer many novel interaction possibilities.

Touch-sensing devices do not include devices that provide active tactile or force feedback [22]. These are all *output* modalities that allow a device to physically respond to user actions by moving, resisting motion, or changing texture under software control. Touch sensing is an *input* channel; touch sensing allows the computer to have greater awareness of what the user is doing with the input device.

To appear in ACM CHI'99
Conf. on Human Factors in Computing Systems

Fig. 1 *Left*: The TouchTrackball (a modified Kensington Expert Mouse) senses when the user touches the ball. *Right*: The Scrolling TouchMouse (a modified Microsoft IntelliMouse Pro) senses when the user is holding the mouse by detecting touch in the combined palm/thumb areas. It can also sense when the user touches the wheel, the areas immediately above and below the wheel, or the left mouse button.

Of course, certain input devices (such as touchpads, touchscreens, and touch tablets) that require touch as part of their normal operation have been available for many years. In all of these devices, one cannot specify positional data without touching the device, nor can one touch the device without specifying a position; hence touch sensing and position sensing are tightly coupled in these devices. Yet once it is recognized that touch sensing is an orthogonal property of input devices that need not be strictly coupled to position sensing, it becomes clear that there are many unexplored possibilities for input devices such as mice or trackballs that can sense one or more independent bits of touch data (*Fig. 1*).

We present two examples of interaction techniques that match these new input devices to appropriate tasks. The *On-Demand Interface* dynamically partitions screen real estate depending on what the user is doing, as sensed by implicit interaction with touch sensors. For example, when the user lets go of the mouse, an application's toolbars are no longer needed, so we fade out the toolbars and maximize the screen real estate of the underlying document, thus presenting a simpler and less cluttered display. By contrast, we use the touch sensors located above and below the wheel on the *Scrolling TouchMouse* to support explicit, consciously activated interactions; the user can *tap* on these touch sensors to issue Page Up and Page Down requests. Touch sensors allow this functionality to be supported in very little physical real estate and without imposing undue restrictions on the shape or curvature of the region to be sensed. We conclude by enumerating some general properties of touch sensors that we hope will prove useful to consider in the design of touch-sensing input devices and interaction techniques.

PREVIOUS WORK

Buxton proposes a taxonomy of input devices [3] that draws a distinction between input devices that operate by touch (such as a touchpad) versus input devices that operate via a mechanical intermediary (such as a stylus on a tablet). Card, Mackinlay, and Robertson [5] extend this taxonomy but give no special treatment to devices that operate via touch. These taxonomies do not suggest examples of touch-sensing positioning devices other than the touchpad, touchscreen, and touch tablet. Buxton et al. provide an insightful analysis of touch-sensitive tablet input [4], noting that touch tablets can sense a pair of signals that a traditional mouse cannot: *Touch* and *Release*. Our work shows how multiple pairs of such signals, in the form of touch sensors, can be applied to the mouse or other devices.

For the case of the mouse, we have already introduced one version of such a device, called the TouchMouse, in previous work [10]. This particular TouchMouse incorporated a pair of contact sensors, one for the thumb/palm rest area of the mouse, and a second for the left mouse button. This TouchMouse was used in combination with a touchpad (for the nonpreferred hand) to support two-handed input. The present paper demonstrates the TouchTrackball and a new variation of the TouchMouse, matches these devices to new interaction techniques, and discusses the properties of touch-sensing devices in general.

Balakrishnan and Patel describe the PadMouse, which is a touchpad integrated with a mouse [1]. The PadMouse can sense when the user's finger touches the touchpad. The TouchCube [12] is a cube that has touchpads mounted on its faces to allow 3D manipulations. Rouse [21] uses a panel with 4 control pads, surrounding a fifth central pad, to implement a "touch sensitive joystick." Rouse's technique only senses *simultaneous* contact between the thumb on the central pad and the surrounding directional pads. Fakespace sells *Pinch Gloves* (derived from ChordGloves [17]), which detect contact between two or more digits of the gloves.

Harrison et al. [7] detect contact with handheld displays using pressure sensors, and demonstrate interaction techniques for scrolling and for automatically detecting the user's handedness. Harrison et al. also draw a distinction between explicit actions that are consciously initiated by the user, versus implicit actions where the computer senses what the user naturally does with the device.

The Haptic Lens and HoloWall do not directly sense touch, but nonetheless achieve a similar effect using cameras. The Haptic Lens [23] senses the depression of an elastomer at multiple points using a camera mounted behind the elastomer. The HoloWall [18] uses an infrared camera to track the position of the user's hands or a physical object held against a projection screen. Only objects close to the projection surface are visible to the camera and thus the HoloWall can detect when objects enter or leave proximity.

Pickering [20] describes a number of technologies for touchscreens (including capacitive, infrared (IR) detection systems, resistive membrane, and surface acoustic wave detection); any of these technologies could potentially be used to implement touch-sensing input devices. For example, when a user grabs a Microsoft Sidewinder Force Feedback Pro joystick, this triggers an IR beam sensor and enables the joystick's force feedback response.

Looking beyond direct contact sensors, a number of non-contact proximity sensing devices and technologies are available. Sinks in public restrooms activate when the user's hands reflect an IR beam. Burglar alarms and outdoor lights often include motion detectors or light-level sensors. Electric field sensing devices [26][24] can detect the capacitance of the user's hand or body to allow deviceless position or orientation sensing in multiple dimensions. Our touch-sensing input devices also sense capacitance, but by design we use this signal in a contact-sensing role. In principle, an input device could incorporate both contact sensors and proximity sensors based on electric fields or other technologies.

The following taxonomy organizes the various tactile input technologies discussed above. The columns are divided into *contact* and *non-contact* technologies, with the *contact* category subdivided into touch-sensing versus pressure or force sensing technologies. The rows of the table classify these technologies as either *discrete* (providing an on / off signal only) or *continuous* if they return a proportional signal (e.g., contact area, pressure, or range to a target). A technology is *single-channel* if it measures touch, pressure, or proximity at a single point, or *multi-channel* if it includes multiple sensors or multiple points of contact. The table omits the position and orientation-sensing properties of input devices as these are handled well by previous taxonomies [3][5]. The table also does not attempt to organize the various technologies listed within each cell.

		CONTACT		NON-CONTACT
		Touch-sensing	**Pressure / Force**	**Proximity**
DISCRETE	**Single channel**	Touchpad touch tablet touchscreens (except IR) touch-based switches PadMouse [1]	push button membrane switch Palm Pilot screen (pressure required) supermarket floor mats car seat: weight sensors for airbag	motion detectors electro-magnetic field sensor [11] Light-level sensor Sidewinder force-feedback joystick (IR beam sensor) IR touchscreens
	Multi-channel	TouchMouse TouchCube [12] touch-sensitive joystick [21] Pinch Gloves [17]	Psychic Space [13] (A grid of floor tiles that can sense which tiles a user is standing on.)	
CONTINUOUS	**Single channel**	contact area (e.g. some touchpads & touchscreens)	pressure-sensitive touch tablet [4] vector input touchscreen [9] torque sensor isometric joystick	laser rangefinder stud finder
	Multi-channel		Multi-touch tablet w/ pressure [15] pressure sensors on handhelds [7] Haptic lens (deform-ation at multiple points) [23]	HoloWall [18] Field-sensing devices [24][26]

Table 1 : Classification of tactile input technologies.

TOUCH SENSING: HOW IT WORKS

The touch-sensing input devices described in this paper employ the circuitry shown in Fig. 2, which senses contact from the user's hand– no pressure or mechanical actuation of a switch is necessary to trigger the touch sensor. The "touch sensors" are conductive surfaces on the exterior of the device shell that are applied using conductive paint (available from Chemtronics [6]). The conductive paint is then connected internally to the touch sensing circuitry.

The internal circuitry generates a 30 Hz square wave that is present on the conductive paint pad. The parasitic capacitance of the user's hand induces a slight time delay in this square wave. When this time delay passes a critical threshold, a *Touch* or *Release* event is generated. A potentiometer (shown in the circuit diagram) allows adjustment of this threshold to accommodate conductive surfaces of various sizes; this only needs to be set once when the circuit is constructed (no calibration step is required for individual users). To provide a good coupling with the tactile feedback that the user feels, the capacitance sensors are set to generate *Touch / Release* events only and exactly when the user's hand actually makes (or breaks) contact with the surface. Our current prototype sends the touch data to the host PC's parallel port.

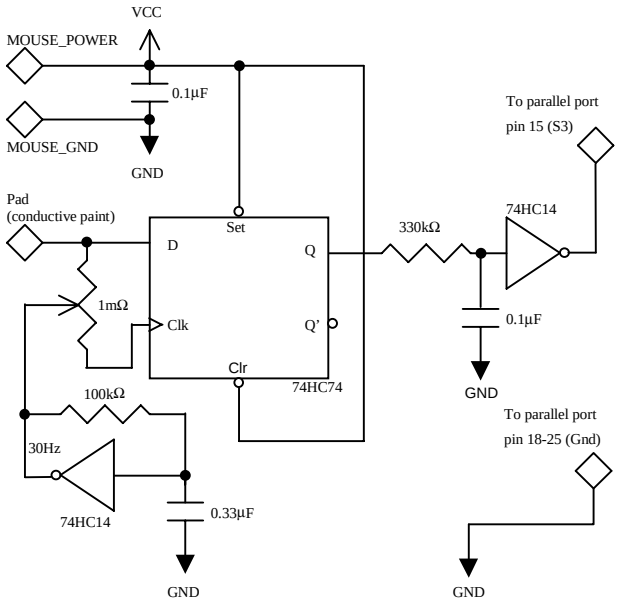

Fig. 2 Circuit diagram for a single touch sensor.

When providing *multiple* touch sensors with the circuit described above, the 30 Hz square wave can pass through the user's body and be picked up by another touch sensor as a false *Touch* or *Release* signal. Thus, to avoid interference, all devices that the user may be touching at a given time should be synchronized to the same square wave.

Software Emulation

One could attempt to emulate *Touch* and *Release* events from software based only on the events provided by a normal mouse. Although this approach may be "good enough" for some interaction techniques or to support situations in which a touch-sensing device is not available, it suffers from two significant drawbacks. First, one cannot distinguish a user holding the mouse still from a user that has let go of the mouse; this also implies that one cannot know with certainty that subsequent mouse motion occurs because the user just touched the mouse, or because the user moved the mouse after holding it stationary for some period of time. A second limitation of software emulation is that only a single *Touch / Release* event pair for the entire input device can be inferred in this way. Without using actual touch sensors, it is impossible to know precisely which part(s) of the input device the user is touching, or to integrate multiple touch-sensitive controls with a device.

TOUCH-SENSITIVE INTERACTION TECHNIQUES

We now discuss specific interaction techniques that use touch sensors to good advantage. These techniques can be broadly categorized as implicit techniques, which passively sense how the user naturally uses an input device, versus explicit techniques, which require the user to learn a new way of touching or using the input device.

Implicit Actions Based on Touching an Input Device

Touch sensors can provide applications with information about the context of the user's work, at the level of which input devices the user is currently holding. Implicit actions use this information to improve the delivery and timeliness of user interface services, without requiring the user to learn a new way to use the input device. The user can benefit from touch sensing without necessarily even realizing that the device senses when he or she touches it. The following section demonstrates how this style of implicit interaction can be used to support the On-Demand Interface, and presents initial usability testing results for this technique.

The On-Demand Interface

Limited screen real estate is one of the most enduring design constraints of graphical user interfaces. Display resolutions are creeping upward, but quite slowly when compared to advances in memory and processor speed. Current market research data suggest that 66% of PC users are still restricted to a 640x480 pixel display surface [19].

The On-Demand Interface uses touch sensors to derive contextual information that can be used to make decisions about the relative importance of different parts of a graphical interface display. We use the touch sensors provided by the TouchTrackball and the Scrolling TouchMouse to determine changes to the current task context, and thus to dynamically shift the focus of attention between different layers or portions of the display. It may be possible to use traditional input events such as mouse motion or clicks to emulate some aspects of the On-Demand Interface, but given that the signals from the touch sensors are reliable, unambiguous, and require little or no overhead to use, we believe these provide a superior information source upon which to base the technique.

For example, toolbars can make a large number of functions "discoverable" and easy to access for the user, but they have often been criticized because these benefits come at the cost of permanently consuming screen real estate

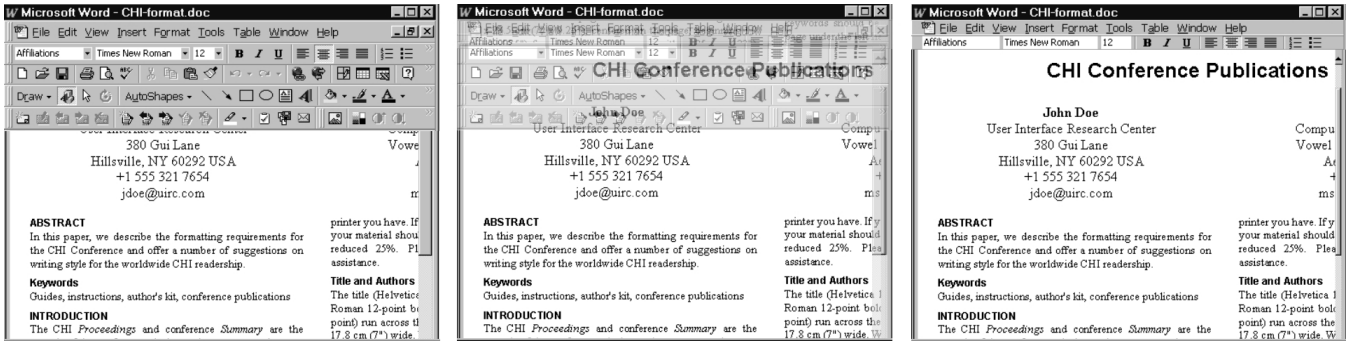

Fig. 3: When the user releases the mouse, the toolbars fade out to maximize screen real estate for the document.

[14]. Although some toolbars do provide visual indications of state (e.g. the current font and point size), most toolbars display no useful state information when the user is just looking at a document or entering text with the keyboard.

In the On-Demand Interface, when the user touches or releases the TouchMouse, the toolbars fade in or fade out on an as-needed basis using smooth alpha-transparency animation[1]. Touching the mouse causes the tool bars to fade in quickly, while releasing the mouse causes the toolbars to fade out gradually. The end result is that when the user is not actively using the toolbars, the screen appears simpler and less cluttered, while the display real estate allocated to the document itself is maximized (*Fig. 3*). In the current prototype, we leave the toolbar slightly transparent even when it is faded in so that the user can maintain awareness of parts of the document that are underneath the toolbar.

We chose to use animations of alpha-transparency rather than animated motion such as sliding or zooming. Motion draws the user's attention, and our design goal is for the interface to change in a manner that is minimally distracting. Fading in takes place quickly (over 0.3 seconds) because the user may be grabbing the mouse with the intent to select an item from the toolbar; fading out takes place more gradually (over 2.0 seconds) because we do not want to draw the user's attention to the withdrawal of the toolbars. The toolbars could appear instantaneously, but we find that instantaneous transitions seem very jarring and unpleasant, especially given that such a transition will occur every time the user grabs the mouse.

Note that although it would be possible to fade out *all* menus and toolbars, this may not always be appropriate. Menus serve as reminder for keyboard shortcuts during text entry, and some toolbars do provide visual indications of state. However, one can distinguish the size of the toolbar that is best for interaction with the mouse versus the size of the toolbar that is necessary to visually display the desired state information. As seen in Fig. 3, the On-Demand Interface fades in a *compact* toolbar, scrollbar, and menu representation while the toolbars fade out. During our usability tests, most users did not notice or comment on this change in appearance, although one user did mention that "I would expect the Bold icon to stay in the same place."

We also use the touch sensors on the wheel and on the mouse button of the Scrolling TouchMouse to support a *reading mode* of interaction when the user engages the wheel. Rotating the wheel on a standard IntelliMouse scrolls the document line-by-line, and we have observed that users will often keep their finger perched on the wheel when they pause to read the document. Since the user does not need the toolbars while using the wheel, the On-Demand Interface senses initial contact with the wheel and uses this signal to gradually fade out the toolbars and again maximize the display real estate allocated to the document. In our current design, a faint trace of the toolbars remains visible while in reading mode so that the user can see where the toolbars will appear upon return to normal mouse usage. The interface reverts to normal mouse usage when the user's index finger returns to and touches the mouse button, which quickly fades the toolbars back in. Although accidentally touching the wheel and thus switching to reading mode might seem to be a problem, during our usability tests we found this was not a significant issue. Regarding this point, one test user commented that "I like that it fades back in kind of quick. So if you had accidentally touched [the wheel] it's no big deal."

Fig. 4 When the user touches the trackball, the ToolGlass fades in quickly; the toolbars simultaneously fade out.

We use the TouchTrackball to apply the On-Demand Interface concept to the ToolGlass technique [2], which provides the user with a set of movable semi-transparent "click-through" tools that are controlled with a trackball in the nonpreferred hand. When the user touches the trackball, the ToolGlass fades in quickly over 0.3 seconds; if the user is also touching the mouse, the toolbars simultaneously fade out (*Fig. 4*). When the user releases the trackball, after a brief time delay[2] the ToolGlass fades out gradually, and if the user is touching the mouse, the toolbars simultaneously fade in (over 1.0 second). If the user clicks-through a tool

[1] We implemented this prototype using a 3D graphics accelerator to provide alpha-blending of texture maps; it is not a fully functional implementation of Microsoft Word.

[2] This time delay (0.5 seconds) allows one to "reclutch" the trackball (release and recenter one's hand on the ball) without undesired changes to the display.

to initiate a command with the ToolGlass, it fades out immediately (over 0.2 seconds) and does not fade back in unless the user moves the trackball or releases and touches the trackball again.

Informal Usability Evaluation

We conducted informal usability tests of the On-Demand Interface, which were intended to explore user acceptance of this technique and to identify usability problems with our current implementation. We recruited 11 users from an internal pool of administrative assistants for this study. All users were familiar with Microsoft Word but none had seen or tried our touch-sensing input devices before.

For the testing, we implemented the On-Demand Interface technique in a prototype that fully supported the various fade in / fade out transitions in response to interacting with the input devices, but only supported limited interaction with the document itself (users could click and drag with the mouse to circle regions of text) and limited keyboard text entry (as the user typed, text actually appeared in a small separate box below the main window). Nonetheless, we feel that this functionality was sufficient to test the utility of the On-Demand interface concept.

In particular, since we felt that *transitions* between the different task contexts recognized by the On-Demand Interface might result in usability problems, we tried to test interleaving of the various task contexts as much as possible. For example, we asked users to highlight a word with the mouse; then type in some text to replace this; then click on the Bold icon in the toolbar; then switch back to typing again, and so on. After performing several structured tasks of this sort, users were also encouraged to play around with the interface to get a more thorough feel for what they did or did not like.

Test users were quite enthusiastic about the ability to see more of the screen during typing and scrolling tasks, while at the same time having the toolbar available on short notice. One user explained that "I like that [the toolbar] comes up quickly when you need it and you can control how long it stays up" and that "all the extra stuff isn't there when I don't need it." Subjective questionnaire ratings on a 1 (disagree) to 5 (agree) scale confirmed these comments: users reported that the TouchMouse was easy to use and that they liked seeing more of the document at once (average rating 4.5 for both questions).

Most users also liked the fading animations that transitioned between screen layouts. Two users did feel that the transition from the toolbars to a "clean screen" for text entry was too slow. One user wanted the toolbar to slide into place instead of fading. However, it was clear that transitions between the toolbars and the "clean screen" were well accepted overall and were not the source of any significant usability problems; when asked if "switching between the keyboard and mouse is disconcerting," users clearly disagreed (average rating 1.9). Users also felt that the touch sensors provided an appropriate way to control these transitions, offering comments such as "I really like the touch-sensitive – I really like that a lot."

As noted above, in this prototype we experimented with leaving the toolbars slightly transparent even when they were fully faded in to allow some awareness of the occluded portions of the document. We felt this was a useful feature, but *all 11 users* reported that they disliked the slightly transparent toolbar, and often in no uncertain terms: one user described it as looking "like a wet newspaper" while another simply stated, "I hate that!" Users clearly felt the display should always transition to fully opaque or fully invisible states. In retrospect, we realized that this dissatisfaction with semi-transparent toolbars on top of a text editing application perhaps should have been expected given that studies of transparent interfaces have shown text backgrounds lead to relatively poor performance [8], and we may not have chosen the icon colors, styles, or transparency levels with sufficient care.

With regard to the TouchTrackball and ToolGlass, users also generally liked that the ToolGlass faded in when they touched the trackball: "That's cool when the ball takes over the hand commands." One user did comment that the appearance of the ToolGlass, and simultaneous disappearance of the toolbars, was the only transition where "I felt like too much was going on." Perhaps the toolbars should stay put or fade out more slowly in this case. Interestingly, in contrast to the strongly negative response to the slightly see-through toolbars, most users had no problem with the semi-transparency of the ToolGlass; it was probably easier to visually separate the foreground and background layers in this case because the user can move the ToolGlass. For example, one user mentioned that "It's good to see where an action would be and what it would look like." A couple of users commented that using two hands "would definitely take some getting used to," but in general users seemed to agree that that "using the trackball was easy" (average 4.3).

The On-Demand Interface demonstrates a novel application of touch sensors that dynamically adjusts screen real estate to get unnecessary portions of the interface out of the user's face. Since the initial user feedback has been encouraging, we plan to add these capabilities to a more fully functional application and perform further studies of the technique to determine if additional issues might arise with long-term use. We are also investigating the appropriateness of the technique for other interface components such as floating tool palettes or dialog boxes.

Explicit Actions Based on Touch Sensors

A second general class of interaction techniques uses touch sensors to allow an input device to express an enhanced vocabulary of explicit actions, but the user must learn these new ways of touching or using the input device to fully benefit from them. Clearly, such actions should have minimal impact on the way one would normally use the device, so that the new capabilities do not interfere with the user's existing skills for controlling the input device.

The Scrolling TouchMouse

The Scrolling TouchMouse (*Fig. 1, right*) is a modified Microsoft IntelliMouse Pro mouse. This mouse includes a

wheel that can be used for scrolling, and an oblong plastic basin that surrounds the wheel. The wheel can also be clicked for use as a middle mouse button.

In the previous section, we described how several of the touch sensors on the Scrolling TouchMouse could be used for implicit sensing of the user's task context. In this section, we describe the use of two touch sensors that we have added to the basin, one above and one below the wheel. In addition to the usual line-by-line scrolling supporting by rolling the wheel, these touch sensors enhance scrolling actions with several new behaviors:

- *Tapping*: Tapping the top part of the basin triggers a Page Up command; tapping the bottom of the basin triggers a Page Down. The wheel is good for short-range scrolling, but is less effective for long range scrolling [25]; the tapping gesture provides an effective means for discrete scrolling at a larger scale of motion.
- *Roll-and-hold*: This extends the gesture of rolling the wheel to support smooth scrolling. Rolling the wheel until the finger contacts the top touch sensor or the bottom touch sensor initiates continuous up scrolling or continuous down scrolling, respectively. The scrolling starts after a brief delay (0.15 seconds) to prevent accidental activation from briefly brushing the sensor.
- *Reading sensor*: We already use the wheel touch sensor in the On-Demand interface to sense when the user begins a scrolling interaction. Since IntelliMouse users often leave their finger perched on the wheel while reading, an intriguing possibility is that dwell time on the wheel may prove useful as a predictor of how much time the user has spent reading content on a web page, for example. We have not yet tested the wheel sensor in this role.

We performed informal evaluations with ten test users recruited from the Microsoft Usability pool; 3 of the 10 users had previously used a mouse including a scrolling wheel. Test users were asked to scroll to various points within a long web page containing approximately 10 pages of content. For this informal study, we did not control the distances to the various scrolling targets, nor did we test the interleaving of scrolling with other common mouse tasks; a future study should address these issues. Our main goals were to observe user responses to the device, discover some potential usability problems, and see if touch sensors were effective for these kinds of interactions.

Users found the tapping feature extremely appealing. When asked to respond to the statement "Paging up and down with the TouchMouse was easier than paging with my current mouse" user responses averaged a 4.6 (again on a 1-5 scale). One user commented "I really like this, it's pretty cool… just tap, tap, tap, done!" while another commented that "I didn't really see a reason for the wheel. Just touching the gold [sensor] was easy enough." One user did feel that "the tap surface should be larger."

Several users expected the tapping sensors to support an additional gesture that we currently have not implemented, the *tap-and-hold*. Tapping and then holding one's finger would trigger a paging command followed by more rapid continuous up or down scrolling. One potential problem with the tap-and-hold is that simply resting one's finger on the basin after tapping would now trigger an action. We plan to experiment with a tap-and-hold gesture to see whether or not it is genuinely useful.

Problems with the device related to the wheel itself and the roll-and-hold behavior. When asked to respond to "I liked the way the wheel on the TouchMouse felt while scrolling," user responses averaged a 3.2 (with 3 = neither agree nor disagree). Several difficulties led to this lukewarm response. Our touch-sensing modifications to the wheel made it slightly slippery and harder to turn; this problem also made it more likely that users would click the wheel by mistake, and due to a technical glitch, the roll-and-hold did not work correctly when this happened. Also, the "continuous" scrolling implemented in our prototype was jerky and moved too slowly. Users did not like this. Fortunately, these are not inherent problems and will be improved in our next design iteration.

Despite the problems with the roll-and-hold mentioned above, users felt that overall "The TouchMouse was easy to use for scrolling" (responses averaged 4.1). Users also clearly liked the concept of having additional scrolling or paging commands on the mouse (responses averaged 4.8). In combination with the enthusiastic user response to the tapping feature, this demonstrates that the Scrolling TouchMouse successfully employs touch sensors to support new functionality while occupying a minimum of device real estate, and without making the device look significantly more cluttered with physical buttons.

PROPERTIES OF TOUCH-SENSING DEVICES

We now consider the properties of touch sensors and touch sensing input devices in general. Based on our design experience, we feel these are useful issues to consider when designing touch-sensing input devices and interaction techniques, and hope that they may be suggestive of additional possibilities.

Similarities between Touch Sensors and Touch Tablets

Although the touch sensors that we use do not sense positional information, since the geometric arrangement of sensors is known ahead of time**,** one can potentially confer to the mouse properties that, in the past, have normally been associated with touch tablets. Thus touch sensors have some properties similar to those of touch tablets as enumerated by Buxton, Hill, and Rowley [4]. For example:

- *No moving parts:* Touch sensors have no moving parts.
- *No mechanical intermediary:* Touch sensors require no mechanical intermediary to activate them.
- *Operation by feel:* Touch sensors can be arranged into regions that act like a physical template on a touch tablet. The user can *feel* the touch-sensing regions

(e.g., the Page Up / Down controls on the Scrolling TouchMouse) without looking at the device or at the screen. This can reduce the time that would be required to switch between devices or widgets on the screen.

- *Feedback:* Touch sensors differ from traditional pushbuttons in the amount and type of feedback provided. Compared to a mouse button, for example, the user does not feel or hear a distinct "click" when a touch sensor is activated. For cases where a touch sensor is being used in an implicit role and is not being used to simulate such devices, however, such feedback may not be needed or even desired.

Other Properties of Touch Sensors

Touch sensors have a number of additional unique properties that can be useful to consider in the design of devices and interaction techniques:

- *Accidental activation:* Because touch sensors require zero activation force, they may be prone to accidental activation due to inadvertent contact. In particular, when touch sensors are used to trigger explicit actions, care needs to be taken so that the user can rest his or her hand comfortably on the device without triggering an undesired action. Of course, for implicit sensing applications, "accidental" activation is precisely the property that makes touch sensors useful.

- *Flexible form factor:* Unlike a touchpad, which generally requires a planar form factor, touch sensors can have an extremely flexible shape; curved surfaces, uneven surfaces, or even moving parts such as wheels and trackballs can be touch sensitive. Touch sensors also have a near zero vertical profile (assuming the touch-sensing electronics can be located elsewhere), which allows them to be used in tight spaces that may not readily accommodate a traditional pushbutton.

- *Unobtrusive:* Touch sensors can be added to a device without necessarily making it *look* complex and cluttered with buttons. The user may not even have to be aware that the device incorporates a touch sensor.

- *Low overhead to disengage:* Some input devices, such as a puck on a Wacom tablet, can provide *In Proximity* and *Out Of Proximity* signals when the puck is placed on or removed from the tablet. Although this pair of events is similar to the *Touch* and *Release* events generated by touch sensors, they are useful for different things. For example, removing one's finger from a touchpad requires considerably less overhead than lifting a puck from a tablet. Thus, the proximity signals provided by a tablet and the touch signals provided by a touch sensor support logically distinct device states [10].

- *Deactivation from software:* Touch sensors lend themselves to deactivation from software, because a touch sensor does not respond to user input with a physical "click." Thus, unlike a pushbutton, a disabled touch sensor does not offer any false physical feedback when it is touched, which is useful if the user is in a context where the action is not valid or if the user does not want an added feature.

- *Additional physical gestures:* Some gestures that are not captured well by pushbuttons, such as tapping or simply maintaining contact with a portion of the device, can be captured by touch sensors. A pushbutton that includes a touch sensor [10] can capture these gestures, in addition to the traditional *click* and *drag*.

Intentional Control vs. Cognitive and Physical Burden

Touch-sensing and proximity-sensing technologies offer an inherent tradeoff in intentional control versus the cognitive and physical burden of an input transaction. The progression from button-click, to touch, to hand-near-device is potentially accompanied by a decrease in intentional control by the user, and hence increases the inferential burden (and error rates) of interpretation. This means that, although error rates can be minimized by good design, accidental activation will occur and thus actions triggered by touch or proximity sensors should have a low cost from errors of interpretation.

Yet this apparent weakness is also a strength, as a reduction of intentional control also implies a potential decrease in the cognitive burden of making explicit decisions to perform an action. Thus, when used in an implicit role, touch sensing can provide enhanced device functionality with little or no added cognitive burden. Touching or letting go of the device is an inherent part of *what the user would have to do anyway to use the input device*, so nothing new has to be learned by the user in terms of operating the input device. Touch-sensing devices are capable of sensing the *Touch* and *Release* events that in a manner of thinking have always been available, but were ignored by traditional devices such as mice and trackballs.

CONCLUSIONS AND FUTURE WORK

The present work has demonstrated that touch-sensing is an orthogonal property of input devices that does not necessarily have to be coupled with position sensing. This observation suggests new possibilities for touch-sensing input devices, exemplified by the TouchTrackball and Scrolling TouchMouse presented herein. We have also described the hardware needed to actually build touch sensors in the hope that this will encourage other interface designers to experiment with our techniques and explore additional possibilities for touch-sensing devices.

Touch sensors allow some properties that have normally only been associated with touch tablets to be integrated with other input devices such as the mouse. Thus, the touch-sensing mouse provides a set of design properties that neither traditional mice nor traditional touch tablets can match. Touch sensors also provide a number of other unique properties, perhaps the most important of which are (1) zero activation force, allowing implicit sensing of "accidental" contact, and (2) great flexibility of form factor which allows touch sensors to be applied to tight spaces, curved surfaces, or even moving parts.

When matched to appropriate interaction techniques these unique properties of touch-sensing input devices allow user interfaces to effectively support a number of new behaviors. Our initial usability testing results of the On-Demand Interface and the Scrolling TouchMouse demonstrate that touch-sensing devices can provide new behaviors that users find compelling and useful.

However, much future work is still required. We need to refine and more formally evaluate the specific interaction techniques that we have described. Additional study is needed to better understand and characterize the strengths and weaknesses of touch sensors. Also, we feel that a more detailed taxonomy of touch sensing and proximity sensing devices could help to better understand and explore the design space. A good taxonomy of such devices should probably include both sensors and actuators. For that matter, a more unified treatment including audio I/O (microphones and speakers), visual I/O (cameras and displays), and the haptic channels (tactile, force, and kinesthetic I/O) might be useful to describe a wider range of existing devices and suggest future possibilities.

ACKNOWLEDGEMENTS

We would like to thank the Microsoft Hardware Group for ideas and discussions; Hunter Hoffman, Barry Peterson, and Mary Czerwinski for assistance with usability studies; Bill Gaver for thoughtful comments on touch-sensing; Matt Conway for suggested improvements to the paper; Dan Robbins for the photographs; and George Robertson for managerial support and design discussions.

REFERENCES

1. Balakrishnan, R., Patel, P., "The PadMouse: Facilitating Selection and Spatial Positioning for the Non-Dominant Hand," CHI'98, 1998, 9-16.
2. Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T., "Toolglass and Magic Lenses: The See-Through Interface," SIGGRAPH 93, 1993, 73-80.
3. Buxton, W., "Touch, Gesture, and Marking," in *Readings in Human-Computer Interaction: Toward the Year 2000*, R. Baecker, *et al.*, Editors. 1995, Morgan Kaufmann Publishers. p. 469-482.
4. Buxton, W., Hill, R., Rowley, P., "Issues and Techniques in Touch-Sensitive Tablet Input," Computer Graphics, 19 (3): p. 215-224, 1985.
5. Card, S., Mackinlay, J., Robertson, G., "The Design Space of Input Devices," CHI'90 Conf. on Human Factors in Computing Systems, 117-124.
6. Chemtronics, CircuitWorks Conductive Pen, : http://www.chemtronics.com/.
7. Harrison, B., Fishkin, K., Gujar, A., Mochon, C., Want, R., "Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces," CHI'98, 17-24.
8. Harrison & Vicente, "An Experimental Evaluation of Transparent Menu Usage," CHI'96, 391-398.
9. Herot, C., Weinzapfel, G., "One-Point Touch Input of Vector Information from Computer Displays," Computer Graphics, 12 (3): p. 210-216, 1978.
10. Hinckley, K., Czerwinski, M., Sinclair, M., "Interaction and Modeling Techniques for Desktop Two-Handed Input," UIST'98, 49-58.
11. Infusion Systems, "Reach" electromagnetic field sensor: http://www.infusionsystems.com/.
12. ITU Research, TouchCube: www.ituresearch.com.
13. Krueger, M., "Artificial Reality II". 1991, Reading, MA: Addison-Wesley.
14. Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B., "The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency," CHI'97, 35-42.
15. Lee, S., Buxton, W., Smith, K., "A Multi-Touch Three Dimensional Touch-Sensitive Tablet," Proc. CHI'85, 1985, 21-25.
16. Loomis, J., Lederman, S., "Tactual Perception," in *Handbook of Perception and Human Performance: Vol. II*, K. Boff et al., eds. 1986, John Wiley and Sons: New York. Chapter 31.
17. Mapes, D., Moshell, J.M., "A Two-Handed Interface for Object Manipulation in Virtual Environments," Presence, 4 (4): p. 403-416, 1995.
18. Matsushita, N., Rekimoto, J., "Holo Wall: Designing a Finger, Hand, Body, and Object Sensitive Wall," UIST'97, 209-210.
19. Media Metrix Inc., HardScan Report, 1998, p. 4.
20. Pickering, J., "Touch-sensitive screens: the technologies and their application," International J. Man-Machine Studies, 25 (3): p. 249-69, 1986.
21. Rouse, P., "Touch-sensitive joystick," Radio & Electronics World, Feb. 1985, p. 23-26.
22. Ruspini, D., Kolarov, K., Khatib, O., "The Haptic Display of Complex Graphical Environments," SIGGRAPH'97, 345-352.
23. Sinclair, M., "The Haptic Lens," SIGRRAPH'97 Visual Proceedings, p. 179.
24. Smith, J.R., White, T., Dodge, C., Allport, D., Paradiso, J., Gershenfeld, N., "Electric Field Sensing for Graphical Interfaces," IEEE Computer Graphics and Applications, May, 1998.
25. Zhai, S., Smith, B.A., Selker, T., "Improving Browsing Performance: A study of four input devices for scrolling and pointing tasks," Proc. Interact '97: The 6th IFIP Conf. on HCI, 286-92.
26. Zimmerman, T., Smith, J, Paradiso, J., Allport, D., Gershenfeld, N., "Applying Electric Field Sensing to Human-Computer Interfaces," CHI'95, 280-287.