Improving Inattentive Operation of Peripheral Touch Controls

Michel Jansen

September 2012

Abstract

Although touch screens are increasingly ubiquitous because of their versatility and flexibility, they still suffer from some problems that negatively affect their suitability for certain situations. Most notably, touch screens are difficult to operate blindly, without looking at the screen. If we can understand what aspects of touch screens are responsible for which problems, and subsequently find solutions to mitigate those problems, the application area of touch screens can be greatly improved.

In this study, we look at the problems of inattentive operation of peripheral touch controls and their solutions, before trying out a novel solution based on adding richer tactile feedback to the touch interface. In an experimental evaluation, a prototype with this feedback resulted in significantly less visual distraction compared to an interface without tactile feedback and was generally preferred by most users.

"Any man who can drive safely while kissing a pretty girl is simply not giving the kiss the attention it deserves."

– Albert Einstein

Contents

Chapter 1

Introduction

Touch screens have long been used for numerous applications. The technology can already be produced cheaply enough that it is found even in budget consumer devices like car navigation systems, handheld gaming consoles and digital photo frames. A recent trend in smart phones and tablet computers is to completely abolish physical buttons in favor of more touch screen real estate.

Touch screens, in particular those supporting multi-touch, have a number of unique advantages over physical buttons. In some contexts, they have been shown to allow users to work twice as fast as with mouse input [35]. Additionally, the layout of the whole screen – and thus the whole control surface – can be changed dynamically. This means the same space can theoretically be reused for different purposes in different contexts. For example, in one mode the screen might be filled with buttons whereas in a next mode, the system uses drag-and-drop style interaction.

As always, this flexibility comes with a price. Whereas physical buttons, dials and other controls have a distinct tangible identity, a touch screen is generally a uniform sheet of glass or plastic; making it hard to identify individual controls that are on the screen. If a touch screen is used as a drop-in replacement for physical controls, it will thus likely be more difficult to use without looking at the screen. These issues limit the applicability of touch screens in all contexts where the user's attention is required elsewhere. Examples include in-vehicle information systems (IVIS) [38], universal remote controls, portable music players, game consoles [78] but also medical instruments, industrial machines and so on. In some cases the higher attentional demand of touch controls merely leads to a decrease in the performance of their operation [31, 78], or in the performance of the larger task the controls are used to perform [38]. In other cases, visual distraction caused by glancing at controls can trigger Inattentional Blindness [63], with serious consequences. If a driver of a car looks away from the road for just 200 ms, it already causes variance in lane position. When they look at a navigation system with a visual display, the effect on driving performance is six times greater.

A better understanding of the limitations of touch screen controls that cause them to demand so much visual attention to operate is needed. This in turn, would allow us to find solutions that make touch screens easier to use for some of these purposes and as a consequence, increase the accessibility and applicability of touch screens. This leads to the following research question:

"How can inattentive operation of peripheral touch screen interfaces be improved".

To find answers to this question, the following sub-questions are considered:

- How do touch screen controls compare to alternative input methods in terms of performance and attention demands and other usability factors.

- What properties of touch screen controls cause problems and at what points in the interaction do these occur?
- What are potential solutions to these problems and what are their effects on performance and (visual) attention demands and other usability factors.

Before rushing off to answer these questions, it is important to have a good definition of the problem and the terminology associated with it.

Inattentive Operation refers to the fact that the controls are operated without paying attention to them. There are many examples of controls in everyday life that are happy to demand our full attention. We fiddle with the buttons on our alarm clock until it is set to the right time; we turn the knob on a blender until our smoothie has reached just the right consistency and when our phone beeps or vibrates in our pocket, we don't hesitate to perform a complicated sequence of gestures to unlock the screen and see what is up. Paying that much attention to a touch screen is not always possible and in some cases it is potentially very dangerous. As easily as we reach for our mobile phones to check a message, type a reply or quickly dial a number, doing so while driving increases the risk of crashing 23-fold by taking our eyes off the road for just a few seconds [54]. Any interface that is used while performing some primary task, such as driving, should not divert attention away from that task for its operation. Preventing distraction is a complex subject that involves not only visual attention, but also a whole range of other cognitive processes and their integration. This is beyond the scope of this study. Although the words "distraction" and "inattention" are used interchangeably throughout this thesis, it is the latter that is the primary consideration of this study. Specifically, the purpose is to reduce the need for visual attention. For touch screen controls, improving *inattentive operation* therefore means reducing the need to glance away from a primary task while using them.

Peripheral touch screen interfaces are a subset of touch screen interfaces that are part of or control a larger system. Just like a keyboard and mouse are peripherals to a computer, where they control things on the screen, touch screens are often used to control various things as well. A screen on the dashboard of a car allows the driver to control the car's radio, air conditioner and navigation; a touch panel in a fancy hotel room lets the guest change the lighting on the fly and so on. What distinguishes these peripheral controls from other interfaces is how the input is separated from the output. When a user is browsing a web site on a tablet computer, he or she interacts with the controls on the screen to change what is displayed on the screen. When a user turns up the heat in a car, the screen shows the new temperature, but the user also hears the fans starting to blow harder, feels the hot air blowing and the temperature rise. This also explains the importance of attention for this class of controls. Peripheral touch screens are by definition secondary to something bigger. They are not meant to engage the user fully, but rather stay in the background and allow the user to focus on the task at hand. They are tools, a means to an end and as such they should not detract attention from that end.

The research questions will be considered in two parts. In the first part, an attempt is made to better understand the problem of inattentive touch interaction by looking closer at the nature of touch input, control interaction and the problems that arise in an inattentive context. The second part then looks at various solutions to these problems before introducing a new solution based on tactile feedback, which is evaluated in terms of performance, attention demand and usability.

Part I

Understanding Inattentive Touch Interaction

Chapter 2

Modeling Control Interaction

Before coming up with solutions addressing the usability problems of touch screen controls, it is necessary to understand those problems and the context in which they occur. There are several existing models of human-computer interaction, but none that are specific and detailed, but at the same time universal enough to help understand the interaction between a user and different types of physical or touch screen controls. This section presents the Control Interaction Model; a model that describes the different stages of user-control interaction that can serve as a framework for understanding interaction problems and opportunities for solutions in both touch screen and physical control types. To put the model to the test, it is used to analyze several existing controls and input methods.

2.1 Background and Related Work

There have been many efforts to understand and model various aspects of human computer interaction. Although not all equally suitable for predicting and analysing problems with the specific class of peripheral touch interfaces meant for inattentive operation, they provide an essential foundation for the more specialised model presented in the second half of this chapter.

One of the earliest contributions to the understanding of human eye-hand coordination and accuracy in tool use, was published in 1899 by R.S. Woodworth [75]. In a series of experiments, Woodworth researched the effects of various factors, such as speed, visual or other sensory feedback, fatigue and practice on the accuracy of movement. He also devised what would later be referred to as the two-component model of upper limb movement, which recognises two phases in targeted motion: a central ballistic phase and a more precise feedback driven phase. Many of Woodworth's findings are still relevant today and have served as the basis for theories such as the Iterative Correction Model [16] and Fitts's law [18].

Fitt's law in turn has proven important in understanding human on-screen interaction and pointing devices, originally in one dimension, but later also in two on-screen dimensions [44] and for physical buttons and touch interfaces [71, 41]. More importantly, it predicts the time needed to activate a control as a function of the size of the target and the distance the user needs to move to reach it.

There are several frameworks and models aimed at comparing and analysing input methods for graphical interfaces. Buxton's Three State Model [10] identifies three discrete states in common input devices, such as the mouse, touch pad, pen tablet, and touch. The model explains some inherent strengths and difficulties found in these input methods, by showing states are missing or skipped. For example, a direct manipulation touch screen interface allows the user to move their finger to the right place without the system's awareness (State 0), and then select a control (State 2), skipping over the process of moving or "tracking" a cursor (State 1). This means the system cannot react to user movement in that state, which among other things explains the lack of a hover state in touch interfaces.

The mental processes involved in human cognition are incredibly complex. The theory of Interacting Cognitive Subsystems, also referred to as the ICS model [17], provides a way to describe cognition as a collection of interacting subsystems using a common architecture. Using the ICS model, an interaction process can be described as a flow of information through sensory- (input) and effectory (output) subsystems [17] and consider them at a higher level of abstraction.

2.2 The Control Interaction Model

To better understand the way humans operate touch screens and to be able to compare them to alterative input methods, it is helpful to have a common framework for analysing different kinds of user-control interaction.

The Control Interaction Model presented in this section was designed for this purpose. The model, shown as an UML activity diagram in Figure 2.1, shows the flow of interaction in five stages, from moving the hands to the control area to finally releasing it. Each of the four phases can be seen as a feedback loop of action (hand motion) and feedback perception (observation) followed by judgement and decision making. The model is a crude abstraction of the real processes at play while a user interacts with touch screen or physical controls, but it is useful in relating different challenges and solutions to different components of the interaction.

Although the model does not explicitly take parallel operation of multiple controls into account, it is relatively straightforward to apply this model in parallel. Depending on the situation, the point where parallelism starts may be different. For example, after bringing a single hand to the control area, the fingers might each individually acquire a different control.

2.2.1 Acquire Control Area Phase

Although the user's intent is to operate certain controls, the first step is to reach for and touch the *control area*; the area or surface that contains the controls. Ideally, the user is able to directly reach the right place of the control area, e.g. where the desired control is located. In that case this step can be coalesced with the *acquire control* phase. There is, however, a good chance that this is not the case, especially when the user has limited visual attention available.

The relation between the *acquire control area* phase and the *acquire control* phase is analogous to the *ballistic phase* and the *control phase* in Woodworth's Two-Component Model [75]. The first phase consists of coarsely positioning the hand, while the second is characterized by more precise movements.

Moreover, the fact that there has not yet been any contact with the system results in a number of important aspects that distinguish this interaction phase from the next. First of all, as there is not yet any contact between the user and the system, no input is produced. In other words, the system is likely not aware of the user's intentions and as such cannot provide response or feedback.

Another aspect that separates the acquisition of the control area from the next phases, is the senses the user has to rely upon. While the user is moving their hand towards the control area, the hands are basically floating in mid-air which means the only methods available for accurate positioning are proprioception and visually observing the hand position. This means that until the user reaches the control area, the only modalities available for feedback are either rather crude or visually demanding.

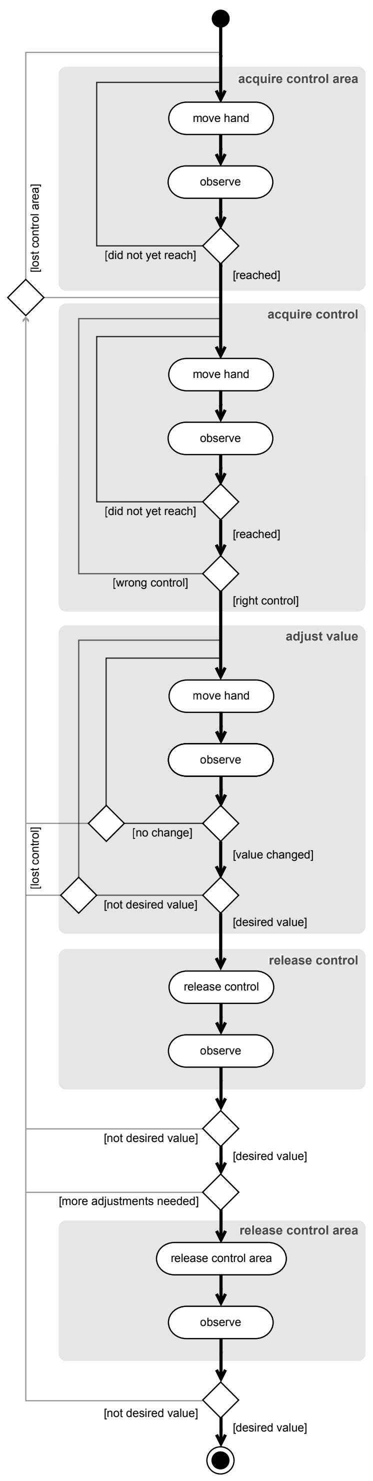


Figure 2.1: Activity diagram showing four phases of control interaction

All in all, acquiring the control area involves a rather short period of moving the arm and hand towards the control area, while monitoring sensomotor and possibly visual feedback. This feedback loop runs until the control area is reached, after which a transition to the next activity takes place.

2.2.2 Acquire Control Phase

The second phase of control interaction involves more precise actions to get hold of the desired control. It is characterized by a feedback loop where the user is moving their hand and positioning their fingers while monitoring feedback to judge whether the appropriate control has been acquired.

If the first phase corresponds to the *ballistic phase* of Woodworth's Two-Component Model [75], this phase can be thought of as the *control phase*. As stated before, these phases are not strictly separate. In most cases there will be some transition from entering the general control area to acquiring a specific control in that area, but there are some characteristics specific to this phase.

First of all, while the user had to rely mostly on proprioception and observation of the hand to reach the control area, there are now generally more senses available for fine positioning. Having acquired the control area, the challenge shifts to finding a specific control within that area. This results in a change from absolute to relative positioning. Moreover, in many cases, the control area will be some kind of surface on which the controls can be found. This essentially reduces the search for the desired control from a 3D to a 2D positioning problem. If the controls have a physical form, the user can feel their shape and position and distinguish them from each other. For some types of controls, such as switches, even their state is discernible by touch.

Unlike in the previous interaction phase, there has now potentially been 'first contact' between the user and the system. From this point on, the system can be aware of the user. Information such as where the user entered the control area can be used to infer the user's intentions and provide feedback and help the user acquire the right controls.

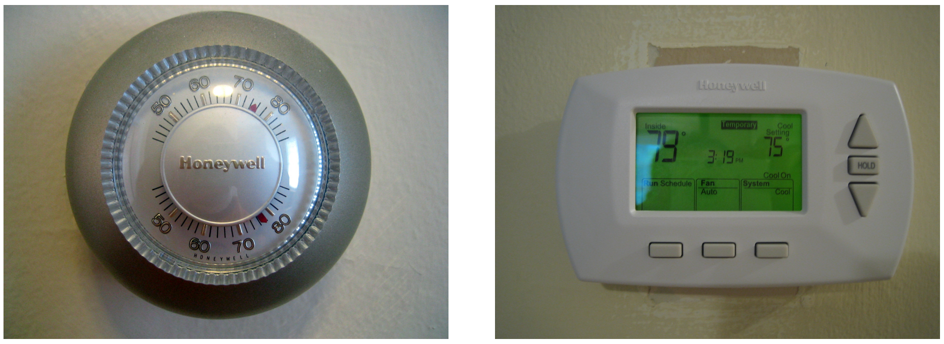
Factors that help predict problems and challenges in this phase are, among other things, the size, amount and distance between controls as well as their individual recognisability. The little raised bumps on the F and J keys of a standard QWERTY keyboard are enough to set them apart from the otherwise identical surrounding keys, and allow experienced touch typists to position their hands without looking.

Fitts already found that the difficulty and time it takes to perform precise motions such as tapping a stylus, transfering a disc from one peg to another or a pin from one hole to another, can be modeled as a function of the size of the target and the distance that needs to be crossed [18]. This principle can be used to make predictions about many kinds of human input devices [71].

2.2.3 Adjust Value Phase

The goal of operating any control is generally to cause some change in the system. This can range from discrete changes, such as activating or deactivating functions to altering continuous parameters of the system. In all cases, the user somehow *adjusts a value* (discrete or continuous) of the system state using controls while observing the output.

The exact method of value manipulation differs per type of control. Push buttons and switches are a good example of discrete controls; sliders, turn knobs, dials etc. are examples of controls that can be used for continuous adjustments. Some controls have an absolute mapping to a value, others have a relative impact on the value they control. What they all have in common,



(a) Analog. Image by Jason Coleman (b) Digital. Image by bobafred

Figure 2.2: Two types of thermostats.

is that there is again a feedback loop consisting of the user using their hands to manipulate the control while monitoring feedback from the system until the desired value is achieved.

The feedback in this source can come from two sources: the control itself and the system's output. In a well-designed system, these two sources reinforce each other. Flipping a light switch causes the light to turn on as soon as the switch clicks into the "on" position. Turning the knob on a sink results in immediate changes in the stream of water coming out of the faucet. Modern systems sometimes suffer from problems when these direct relations are broken. A fluorescent light taking a while to start up may leave the user wonder if she turned the light on right, especially if the light switch is a capacitance switch that only requires a light touch.

As the fluorescent light shows, not all systems allow users to clearly gauge their output. Fiddling with the controls on a climate control system does not cause immediate changes in the ambient temperature. Even though the heating or air conditioner might immediately turn in response to changes to the temperature setting, it will take a while for those changes to become noticable to the user who made them. In such cases, it's important that the system provides other feedback, either by an alternative output display (such as a temperature setting LCD) or through the control itself.

Old-fashioned analog thermostats, like the one pictured in Figure 2.2a, combined the input and display of the target temperature setting. The temperature was changed by turning the dial on the panel to the left or right. Digital thermostats with buttons (as in Figure 2.2b) usually have a small display to show the temperature, but they allow the user to increase the temperature in exact increments by pressing the up or down buttons a certain number of times. If a user of the analog thermostat changes their mind after changing the temperature, they need to carefully turn the dial back to the desired temperature. On the digital thermostat, one can return to the previous temperature by pressing the same number of times on the opposite button. The push buttons of the digital thermostat thus give more feedback about the temperature change than the dial of its analog predecessor, although they arguably require more work to make big adjustments.

Some controls intrinsically provide tactile or audible feedback while they are operated. Other controls do not and require the user to pay attention to the output of the system to track progress towards the target value. A good quality feedback loop helps users feel in control [6] and be aware of the changes they are making. Without it, there is a risk of under- of overshooting the target, both of which can be costly to correct [16]. The quality of the feedback loop and thus the ease of use and performance of operation can be influenced by many factors. The modality of the output and feedback, latency in system response and directness of input mapping are but a few examples.

Depending on the type of control, it may be possible for the user to let it slip or lose control of it while in the process of making adjustments. A person driving a car may let their foot slip off the gas pedal; a digital illustrator may move his or her pen past the boundaries of the tablet and so on.

In some cases there is immediate feedback of having accidentally released the control, It is easy to notice that one's foot has slipped off the pedal, because there is immediate haptic feedback through the foot. Additionally, the car immediately responds by slowing down, resulting in a different engine sound and so on. In those cases, the losing of control is noticed as an unexpected observed change. In other cases, it is not that easy. Moving a pen digitiser outside the tablet's active area does not produce an immediate change in the cursor's position. It does cause the cursor to stop moving along with the pen, which is observed as a lack of expected change, but may take longer to notice.

In both cases, the user effectively drops out of the "adjust value" phase and has to re-acquire the control. The *adjust value* phase ends when the target value is reached.

2.2.4 Release Control Phase

After the user has finished adjusting the target value using the acquired control (and the desired value has been reached), the user is ready to release the control. For most controls, this is a relatively easy step, that simply involves stop pressing down on a control, letting go of one's grip on the control or lifting the finger or hand that is touching it.

Still, there is opportunity for mistakes or problems at this point. Several control types, especialy the ones that require precise control, can be accidentally triggered or activated upon release, causing a change past the desired value. This is a big problem, as the user then has to go back to reacquiring the control or even the control area.

The model also leaves room for situations where multiple controls have to be operated in sequence. After the user is done adjusting the value of one control, they can move on to the next one. Depending on whether the user has lost or remained within the control area, they start over at respectively the *acquire control area* or *acquire control* step.

2.2.5 Release Control Area Phase

The last step in the control interaction model is where the user releases the control area. In some cases, this step might overlap with the release control phase, because the system or the user cannot distinguish between the releasing of a single control or the end of interaction. This can again lead to unintended changes to values, in which case the user has to start over again with re-acquiring the control area.

2.3 Applying the Control Interaction Model

To demonstrate the utility of the control interaction model and establish a background against which to compare touch screen input, it is useful to apply the model to a number of existing control interfaces. As touch screens are often used to replace keyboard and mouse, they will be discussed first. To provide some variety, a rather different control interface has also been included: Microsoft's Kinect.

2.3.1 Keyboard

A keyboard is a control interface for inputting text or controlling a computer. The control area consists of a flat or curved surface with around a hundred keys laid out in a standard pattern. Each individual key is essentially a push button that is momentarily activated when pressed, after which it springs back into position. Each key corresponds to a letter, symbol or command. In this case, we focus on text input only: the user's intention is typing a word in a document on the screen.

Acquiring the control area is relatively straightforward. The keyboard is generally sitting on a desk in front of the user such that the users hands touch it when they are left to rest. If not, the keyboard's large elevated surface area makes it an easy target to find and reach, even without looking or using only peripheral vision.

Next, the user will want to activate several keys in succession; one for each letter in the word. Experienced blind typists can do this without looking at their fingers. They position their hands in a 'home position' by feeling around for two keys (F and J) with a small bump on their surface, and therefore a sligthly different tactile signature. Positioning the hands such that these keys are below the index fingers of each hand, the position of all the other keys is known relative to this home position. Although almost all the keys have the same shape, it is a shape that is easily recognizable by touch. *Acquiring controls* can therefore be done based solely on proprioception and tactile feedback, without looking at the fingers.

For less experienced typists that adopt a 'hunt and peck' strategy, the picture looks slightly different. Rather than placing the hands in the aforementioned 'home position' and typing from there, the user hovers with their hands, or even just the the index fingers of each hand, above the keyboard and 'hunts' for keys like that. Although the hands stay above the keyboard while typing, the user more or less maintains within the control area, yet continuous visual feedback is required to guide each finger to the right keys.

In both strategies, the next step in the interaction is pressing the key, which corresponds to the *adjust value* phase. Most keyboards give off clear feedback in the form of a tactile and audible "click" when the keys are pressed. Additionally, when inputting text, the user can instantly see the result of successful activation on the screen. Finally, the user lifts their finger off the key to *release* the control.

As the keys on a keyboard have just two states (idle or pressed), there will never be a case where the value is "not desired", as in the last to phases of the model. What can happen, is that the user accidentally acquired the wrong control in the first place; resulting in the user pressing the wrong key. Especially in the blind scenario, it is very difficult for the user to detect whether the finger has been put on the right key during the *acquire control* phase. This results in this error only being detected after the *adjust value* step, and ultimately in having to go back to correct after judging there are *more adjustments needed* (e.g. pressing the backspace key).

2.3.2 Mouse

A mouse is an indirect input device that maps relative two-dimensional motion to a cursor on the screen. Additionally, a mouse often has one or more buttons and other input mechanisms, such as a scroll wheel. Operating the buttons works similar to operating a keyboard, so we will focus on using the mouse as a pointing device.

Operating a mouse begins with finding it on the table surface. The *acquire control surface* step thus consists of moving the hand to the mouse and grasping it. By design, a mouse tends to move around, so while it is generally in the same predictable area, the exact position varies. By moving the arm along the table surface, in the periphery of the one's vision, the mouse can usually be found without looking away from the screen. Its ergonomic shape allows the user to easily place their hands around the mouse, after which the fingers more or less automatically rest on the buttons and scroll wheel; final adjustments to hand positioning can be made entirely by touch. The step of acquiring these controls on the mouse's surface is thus largely contained in the *acquire control surface* step. However, that is not all.

The bigger part of operating a mouse-based interface involves interaction that takes place on the screen, with virtual controls that are operated indirectly using the mouse pointer. There is evidence that suggests that for everyday users of a mouse, the area on the screeen becomes an extension of their own space [2], so there is reason to believe that moving the mouse pointer is a lot like moving a hand or finger. An important difference, however, is that adjusting the mouse pointer position is subject to a tight visual feedback loop. Moving the mouse on the table has an immediate effect on the mouse pointer on the screen, but while the mapping is straightforward, moving the mouse left moves the pointer left as well, it is not absolute. There is usually a lot less physical motion required to move the pointer accross the screen; a result of sensitivity and accelleration settings. As a consequence, the user has to keep an eye on the mouse pointer to keep track of the current position (value) and to judge when it has reached its target. This has consequences for both the activity of acquiring on-screen controls as well as adjusting their value.

The *acquire control* phase consists of moving the mouse pointer over the on-screen control. The time and effort this takes depends on the size, position and distance of the control, the specifics of which are well understood using Fitts's law [18] and its derivatives [44].

Depending on the kind of control, adjusting its value can involve as little as a single click on a button – similar to pressing a key on a keyboard – to more complicated interaction that involves "dragging". Controls that require dragging, require the user to first acquire a control, such as the handle of a slider or the corner of a window, then press and hold the mouse button and move the pointer to change. As explained earlier, moving the mouse requires constant visual monitoring, but in many cases (such as resizing a window) the changes are visible in the same area as the mouse pointer.

Releasing the control is done by depressing the mouse button. Additionally, leaving the control area is done by lifting the hand from the mouse. Both actions almost inevitably cause the mouse to move a little, resulting in a shift in on-screen pointer position, especially with high sensitivity settings. This unintentional nudge might mean there are *more adjustments needed*, requiring the user to re-acquire the control or the whole mouse and start over.

2.3.3 Kinect Pointer Interface

Kinect is an input device for Microsoft's XBOX 360 game console. It uses computer vision and a microphone to provide gestural and voice control. The gestural input is more relevant in comparison to touch input, so the voice input is left out in this analysis. Kinect is still rather new, so clear conventions in using its capabilities have yet to emerge. There are several games and they all use different interaction styles, so we restrict this discussion to the "pointer interface" found in the console's menus and Kinect Hub.

In the menus of the XBOX 360 that can be controlled by Kinect, the user can use their hands to select and activate menu options. This is very similar to the point-and-click interaction style that is common for mouse-based interfaces.

Interaction starts with the user stepping into the Kinect's view range (which can be seen as black and white camera output in the bottom of the screen) and waving their hand to the camera. The system responds immediately by showing a small 'wave' icon in the lower right corner of the screen. After a few seconds of waving, the icon disappears and a cursor appears on the screen. Although there is no physical contact between the user and the system, this step still fits the *acquire control area* phase of the control interaction model.

Once the cursor has been acquired, it is mapped to the absolute position of the user's hand. Moving the hand has a direct effect on the cursor position. As with the mouse, a tight visual feedback loop is required to track changes to the cursor position. Selecting a menu option corresponds to the *acquire control* phase and is done by moving the cursor over the desired item on the screen and holding the still. While the cursor moves over each menu item, the system emits visual and audible feedback.

Most of the controls on the Kinect Hub are simple buttons or menu options, which work quite similarly to how one would expect in a mouse-based interface. The biggest difference is that rather than clicking, controls are activated by hovering, as the Kinect interface does not have any physical buttons. This hover-based manipulation is vulnerable to two types of problems. If the user accidentally holds the cursor over a menu option, they might end up unintentially activating something (also known as the Midas-touch problem). The Kinect interface attempts to prevent this by showing a visual countdown timer in a circle before activating a control. Moving the cursor outside of the circle cancels the activation and re-sets the hover timeout. This results in a second problem: if the user does not keep the cursor still enough, they might "slip" off the control and have to re-acquire it as shown in the *adjust value* phase of the model.

Another problem unique to the Kinect is that the system can lose track of the user in mid-interaction. If the user's hand gets occluded behind another player or object, or the user steps outside of the camera's view, the system stops responding. The user then has to step back into view and wave their hands again to continue.

Chapter 3

Touch Interaction; Properties and Problems

3.1 Types of Touch Panels

There are many types of touch screens. A full discussion of all the different touch screen technologies falls outside of the scope of this thesis. A good overview is given by Schöning et al. [69] and by Malik [46]. In short, all touch screens are built from two parts: a screen for output and some sort of touch sensor for input. Larger touch screens and tabletops often use a projected display for their output. The types of screens that are often used in the type of peripheral control panels that are the subject of this study almost all use a Liquid Crystal Display (LCD).

Projected touch screens often use some sort of vision-based technique for detecting where the user touches the screen. The two most popular techniques, as outlined by Schöning et al. [69], are Diffused Illumination (DI) and Frustrated Total Internal Reflection (FTIR). Both techniques rely on a camera pointed at the touch surface to detect where fingers, hands or other objects touch the screen. These techniques are very flexible and can reliably track many touches at the same time.

When an LCD or plasma display is used, it is usually not possible to place a camera behind the touch surface. This means that smaller peripheral touch panels typically employ one of four other techniques:

1. Optical (Infrared-based)
2. Resistive
3. Surface Acoustic Wave
4. Capacitive

The optical and acoustic-based techniques are cheap and easy to install on top of existing systems, and they work with styli as well as fingers, but they are limited in the number of touch points they can reliably detect (they are at best dual-touch). They are not commonly used. Resistance-based touch panels work by placing a pressure sensitive grid over the display. They also work with a stylus as well as fingers (or fingernails), but because they really detect presses and taps, they are not so good at picking up finger swipes and other gestures. Because of their low price, they are still commonly used in consumer devices. The most popular type, however, is the capacitive touch panel. Like the touch pads ubiquitous in laptops, these panels sense the changes in the surface's electrostatic field when fingers touch it. Capacitive touch panels only work with fingers, not with nails, styli or gloved hands (although special styli and gloves exist that do work with capacitive surfaces) and have problems with wet fingers or humid environments [68], but they are very very accurate and robust and able to reliably detect the positions of multiple fingers.

3.2 Interacting with a Touch Panel

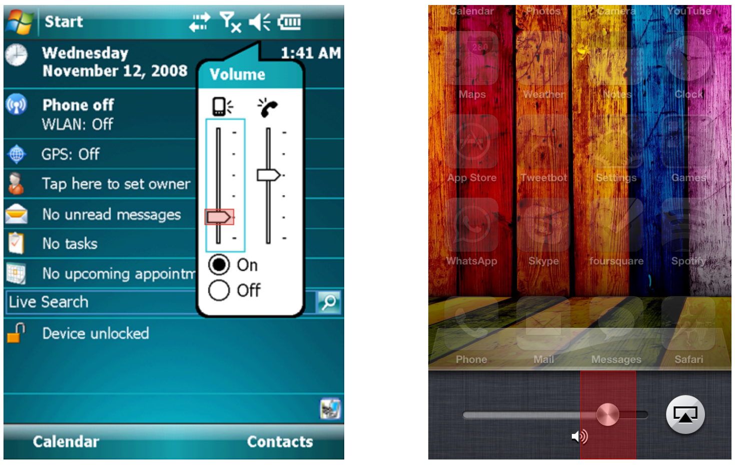
Regardless of the technology used to implement them, the single most distinguishing factor of a touch interface is that it combines visual output with touch input, allowing direct interaction with a GUI through touch presses and gestures. Before diving into the specific challenges of direct manipulation of specific GUI elements, controls and gestures, it's important to start with a closer look at interacting with the touch panel itself.

Ignoring the actual user interface on the screen for a second, a touch panel is essentially a large touch-sensitive slab of glass or plastic. A touch panel can be as small as 3 inches across for a car navigation system to as large as 30 inches or more for a desktop touch screen or touch table. It generally has a completely smooth surface, with edges that may be easily distinguished as a plastic ridge (like on generic PC touch screens) or not at all (as is the case with the smooth bezel around Apple's iPad). With some exceptions, such as Hirsch et al.'s Bidi Screen [29], touch panels are not able to detect the user's fingers until they touch the screen. In the words of Buxton's Three-State Model of Graphical Input [10], this means that there is no "tracking" (or hover) state. Rather than moving a mouse pointer over the desired on-screen control, the user moves their finger there directly, while still in the "out of range" state where the system is not aware of the user's intention.

In terms of the Control Interaction Model of the previous section, this means that the user generally transitions from the *acquire control area* phase to the *acquire control* phase without touching the system. Once the user touches the screen, the system immediately perceives this as input, whether it was intentional or not. There is no way to discriminate between intentional and unintentional touches. This means that users get very little help going from being outside of the control area to adjusting values on a touch screen. There is little sensory input to rely on other than proprioception and vision. The situation is worsened by the fact that there is often no clear "home position" that can be used to rest the hand and base relative motions on.

3.3 Interacting with Touch Controls

Early touch screen interfaces were often nearly identical to mouse-based Graphical User Interfaces, using the same widgets designed for a mouse. They simply treat touch input as if it were mouse input and map it absolutely to the screen not unlike is done for pen tablets. Although touch interfaces have come a long way, there are still plenty GUI controls that have found their way into touch interfaces, as can be seen in Figure 3.1. Control widgets designed for mouse-based input that were initially inspired by real-world controls, such as push buttons, radio buttons and sliders, have come full circle and can now be manipulated directly again. Without their actual physical properties, however, they are not without problems. Understanding these problems is a first step towards creating a better touch experience, but before diving into an in-depth analysis of the problems that plague each type of control, it's important to first identify the different types and see what sets them apart.



(a) Windows Mobile 6 (2008) (b) iOS 5 (2011)

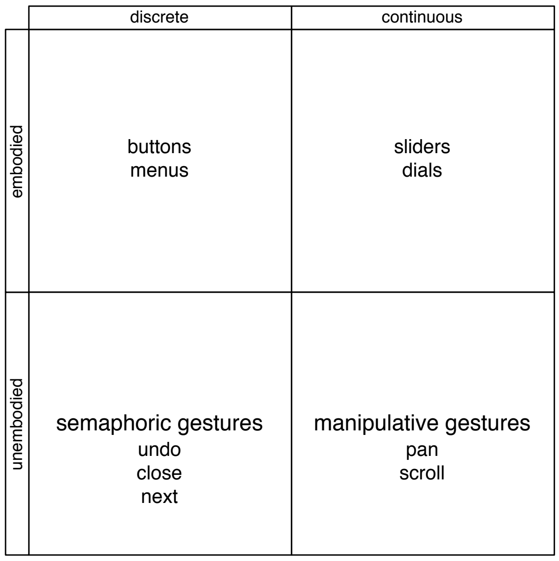
Figure 3.1: Volume controls and their touch areas on mobile phones.

3.3.1 The Design Space of Touch Controls

There are several taxonomies for input devices and controls, such as the one presented by Buxton [9] and later extended in detail for generic controls by Card, Mackinlay and Robinson [12, 11] and multi-touch mice by Benko et al. [5]. Although these also consider touch screen input, and go into great detail about various different types of physical controls, they do not accurately describe on-screen touch controls. Frameworks that do focus on touch input specifically, such as the Karam and Schraefel's taxonomy of gestures [34] and Lao and Wang's Gestural Design Model [40], are often too specific and do not include all types of touch input. In this section, we therefore quickly outline the design space of touch controls by combining these frameworks and identifying some dimensions that set different touch controls apart.

Looking at the screenshots of two generations of touch screen interfaces in Figure 3.1, two examples of GUI controls stick out: buttons and sliders. Like the physical controls they are based on (push buttons and linear potentiometers), one is restricted to discrete input, whereas the other can represent a continuous range. A first dimension in the design space of touch controls is thus *Discrete vs. Continuous*. As with logical input devices, continuous controls can have one or more degrees of freedom [9, 45].

Another dimension of the design space, is *Embodiment* or control visibility. One thing that classic GUI controls and many of their modern variants have in common, is that when used on a touch screen, they use the 1:1 mapping between input and output to allow users to directly manipulate the controls. The user can see a button on the screen and touch it directly. A slider is directly manipulated by touching the handle and dragging it along the axis. In both cases, the control occupies a certain area on the screen. Touch screens, however, are also capable of recognising complex gestures. Although some gestures, such as those used to scroll a page of text or pan a map on the screen, can be seen as directly manipulating the content, gestures almost never involve interacting with an on-screen control. Performing a pinch gesture to zoom a map is different than doing so by pressing a button. In the same vein, scrolling a page by dragging and flicking the page in the opposite direction is different than doing so by dragging a scroll bar, yet in both cases the user is clearly controlling the system. The difference is in the level of embodiment of the controls. The controls for zooming a map or scrolling a page can



	discrete	continuous
embodied	buttons menus	sliders dials
unembodied	semaphoric gestures undo close next	manipulative gestures pan scroll

Figure 3.2: The design space of touch controls, with the two dimensions Direct/Indirect and Discrete/Continuous

be embodied as virtual buttons and sliders, or implied from gestures performed on the content. When multi-touch gestures are supported, a complex language of commands can be created that is devoid of any on-screen representation.

As with embodied, on-screen controls, gestures can be used to manipulate discrete or continuous values. Zooming in a map or scrolling a page are good examples of continuous gestures: they manipulate a continuous value, such as the zoomlevel or scroll position. These gestures are more commonly known as *manipulative gestures* [34]. Gestures can also be used for discrete commands, such as "next page", "close tab" or "undo". The Opera Browser was one of the first to widely allow using such gestures (drawn with the mouse) to issue common commands without having to click on on-screen buttons. These discrete gestures are called *semaphoric gestures* [34].

Together the two dimensions of *Discrete vs Continuous* and *Embodiment* span a design space as shown in Figure 3.2. The two dimensions divide the design space roughly in four quadrants: at the top are discrete embodied controls, such as buttons, and continuous embodied controls, such as sliders. At the bottom, there are unembodied semaphoric and manipulative gestures. It should be noted that this division is not strictly black and white. Continuous embodied touch controls such as sliders often keep following the user's finger even when it strays from the control's on-screen position, which makes adjusting their value very similar to how a manipulative gesture would work. A scroll gesture often directly manipulates the scroll position, but a quick flick gesture with inertial scrolling feels closer to issueing a discrete "page down" command. Still, these four quadrants are a good point to start further analysing touch controls. The next sections take a closer look at each class of controls; Embodied Discrete, Embodied Continuous, Semaphoric Gestures and Manipulative Gestures.

3.3.2 Embodied Discrete: Buttons

Buttons are easily the simplest and most common controls in mouse and touch-based graphical user interfaces. Although they are not always styled to resemble their physical counterparts, a

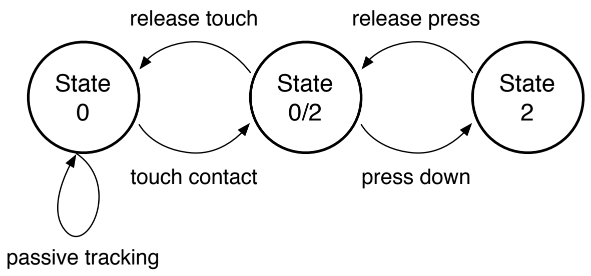


Figure 3.3: States in operating a physical button. Between moving towards the button (State 0) and having pressed the button (State 2), there is an intermediate state where the button is touched but not yet pressed (State 0/2).

lot of the interactive parts of touch screen interfaces, including hyperlinks, thumbnails and icons, are in fact plain buttons. They are the primitives from which more advanced controls, such as radio button groups, pop out menus, item lists and tab bars are built. They are limited in their information capacity (a single button can not easily represent more than two states), but are easily mapped to labeled commands, such as "Calendar" in Figure 3.1a, or to binary system states, such as the "AirPlay" toggle in Figure 3.1b. An advantage of on-screen buttons is that they combine visual representation in the form of a label or icon with input. A disadvantage is that that without such a label or icon, the function of the button is not self-explanatory.

Although on-screen virtual buttons are inspired by real-world physical push buttons, they do not share all the properties of their physical counterparts. The most important difference between a physical button and a touch button is that the latter cannot distinguish between being pressed and being touched.

When pressing a physical button, our finger always first touches the button after which we apply pressure to push the button down to close the electrical circuit inside, which is registered by the system as a button press. Releasing the button breaks the circuit, which the system interprets as the end of the button press. This intermediate state, where the user has successfully acquired the control (e.g. the button), but the system is not yet aware of this is missing from Buxton's Three State Model [10], as it only describes pointer-based input devices. Adding this state as "0/2" – for being simultaneously "out of range" (state 0) from the system's point of view and "acquired" (state 2) for the user – leads to the state diagram in Figure 3.3.

Operating a touch interface (either with fingers or with a stylus) also lacks the tracking state 1, as Buxton already noted when formulating his Three State Model [10]. As with the physical buttons, the user directly moves their finger to the on-screen button, rather than an on-screen mouse pointer. As soon as the user touches the screen, the control in that location is activated, jumping straight to State 2. Another way of saying this, is that as soon as the user's finger touches the screen, he or she has simultaneously finished acquiring the control area, acquiring the control itself and started adjusting the value. The first three phases of the Control Interaction Model are all coalesced into one indiscriminate process.

This is problematic, because the user might have accidentally touched the screen in the wrong place, unintentionally pressing a button. Most touch interfaces therefore only consider a button pushed when the user releases the touch. Moving the finger or stylus outside of the button's surface area while still touching the screen and then releasing the touch cancels the button's activation. This results in the situation shown in Figure 3.4.

In addition to the two states from Buxton's Three State Model, there is an additional state: State 2/1. This state is a mixture of State 2 and State 1. From the user's point of view, a button

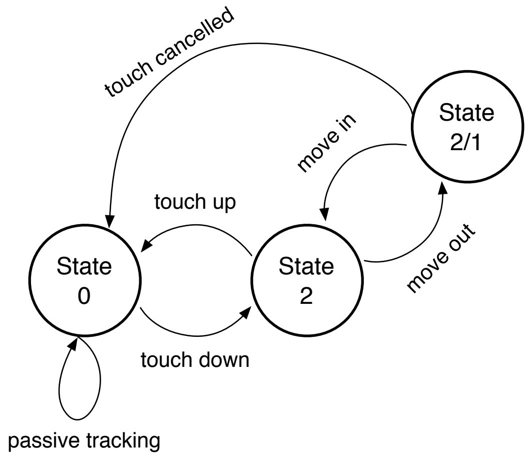
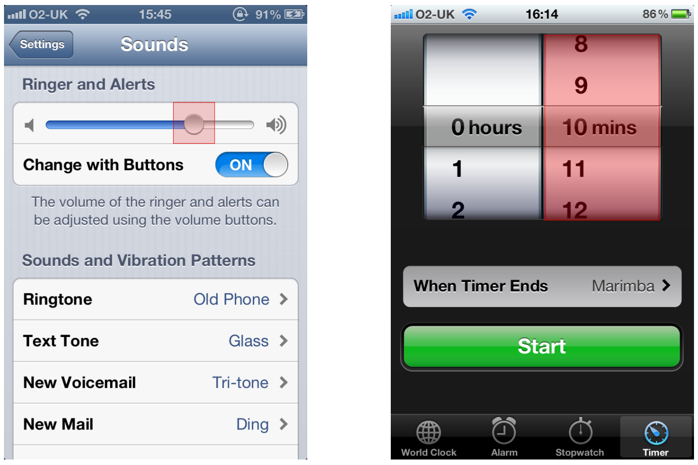


Figure 3.4: States in operating a touch button. As soon as the user touches it, it is activated, but the user can leave State 2 by moving outside of the button (State 2/1).

is being pressed, but by moving outside the button, this can be cancelled. From the system's point of view, as soon as the finger leaves the button, it is no longer activating it, but it is being tracked across the screen. If it moves back into the button, it can go back to state 2, if it gets lifted, the button press gets cancelled. This State 2/1 thus offers a way out to recover from errors caused by the lack of a clear state 1 and acquire control phase preceding the initial touch. By moving outside of the button, the user can escape the adjust value phase and stay close to the screen to facilitate reacquiring the right control.

Apart from initially pressing the wrong button, there is one more scenario that falls outside of Buxton's original Three State Model. The user might also miss the intended button and touch the screen in a place where there is no control. In this case, touching the screen does not immediately result in activating a button. The user is still in the process of acquiring the right control, but just happens to be touching the screen. Taking into account the way most touch screens are implemented, however, the fact that the user is now touching the screen makes all the difference. Although the user is touching the screen and the system is aware of the position of the user's finger, moving the finger generally has no effect on any of the controls on the screen. In the words of Buxton's model, the user is stuck in State 1: the system is tracking the touch position, but there is no way (or risk) to activate anything and transition to State 2 without first releasing the touch. This means that it's easy to undo accidentally pressing a button, but if one misses the button on first touch, it's impossible to activate it without releasing the touch first and trying again.

In addition to the parameters of size and distance that are well known parameters of Fitts' law, there are some other factors that predict performance in operating touch buttons. In a study by Rogers et al. [65], the performance of a grid of smaller, widely spaced buttons was compared to a grid of bigger adjacent buttons. Despite the fact that they were touching each other, the bigger buttons performed far better. They were faster to operate, without leading to more errors. Presumably, the bigger buttons required less precise aiming in intermediate movement stages. Additionally, participants were faster moving to buttons directly to the side or above the last button they pressed than moving to buttons that required diagonal movement, and buttons in the centre of the screen were easier (faster) to press in general. This suggests that in addition to button size and distance, their relative placement is also important, whereas it's not as much of


O2-UK 15:45 91%

Settings Sounds

Ringer and Alerts

Change with Buttons ON

The volume of the ringer and alerts can be adjusted using the volume buttons.

Sounds and Vibration Patterns

Ringtone Old Phone

Text Tone Glass

New Voicemail Tri-tone

New Mail Ding

O2-UK 16:14 86 %

8

9

0 hours 10 mins

1 11

2 12

When Timer Ends Marimba

Start

(a) Slider control in iOS (b) Picker control in iOS

Figure 3.5: Two embodied continuous controls and their touch areas on mobile phones.

a problem when buttons are closely spaced.

What this means for inattentive operation, specifically, is that while the lack of inherent tactile feedback of touch screen buttons makes it challenging to find buttons without glancing at the screen, there are some factors that make it easier, most notably the size and placement of the buttons. If glancing at the screen cannot be avoided altogether, paying attention to these can at least reduce the time needed to operate the controls.

3.3.3 Embodied Continuous: Sliders

As we saw before, a large portion of graphical user interfaces is built from discrete controls such as buttons. In many systems, however it is also desirable to be able to make continuous adjustments. Although it is sometimes sufficient to divide a continuous parameter range into discrete blocks and assign one button (often a radio button) to each value, there are controls specifically for continuous adjustments. Two representative examples of embodied continuous controls are sliders (Figure 3.5a and pickers (Figure 3.5b).

A good example of a parameter controlled by discrete and continuous controls is volume. On many portable devices, this can be controlled using discrete controls: one button reduces the volume by one step, another button increases the volume by one step. Holding either button keeps increasing or decreasing until the button is released or the volume reaches a limit. This way, the whole range of values can be accessed using discrete controls. On many touch devices, however, the volume can also be controlled using a slider. The position of the slider handle maps directly to devices' volume, just like with an analog potentiometer. One extreme end of the slider represents a volume of 0%, the other a volume of 100%. As the user moves the slider, the volume changes immediately.

There are several benefits to using sliders over discrete controls to represent continuous variables. Firstly, sliders offer superior adjustment precision, both spatial and temporal, over discrete controls of the same physical size. A slider of 200 pixels wide can spatially represent 200 steps. It would not be possible to fit many discrete controls in the same space to allow direct access to individual values, which would reduce the number of accessible values. The relative up/down

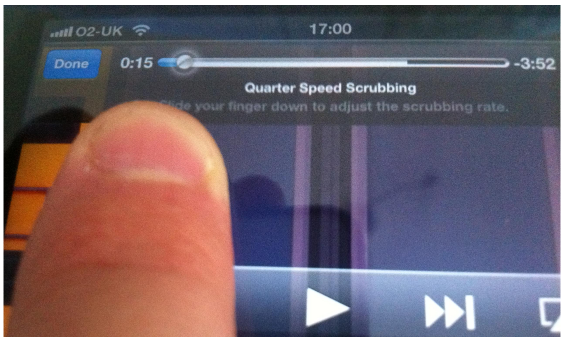


Figure 3.6: Some sliders change their tracking speed based on the distance between the finger and the handle to increase precision

volume controls from the example would require multiple presses or a press-and-hold interaction style, which would either be less precise or much slower. More importantly, sliders are dragged, not pressed. While the slider is being dragged, the user is touching the screen and the system is aware of the position of the user's finger. What this means, in terms of Buxton's Three State Model [10] and the Control Interaction Model, is that the user spends more time in the *adjust value* phase (State 2) compared to acquiring the control (State 0, passive tracking). As a result, there is more opportunity for a feedback loop between system and user. The user drags the volume up, the system immediately becomes louder, until the user finds it too loud and drags the volume down a bit.

Acquiring a slider control is very similar to acquiring a button. The user needs to touch down on the touchable area of the slider's handle and if he or she misses it, release the touch and try again. What happens once the slider handle has been acquired, however, is different. The main way to operate a slider is by dragging its handle across the axis of the slider. As long as the user keeps their finger on the screen, the slider handle will follow its movement. Like their physical couterpart, a slider only moves in one direction (usually horizontal or vertical). Unlike a physical potentiometer, a slider on a touch screen cannot physically restrict the user from moving their finger in other directions. As a result, while dragging the slider handle, the user can move their finger freely across the screen. The handle will remain restricted to the slider's axis and will just follow the finger in that direction. Likewise, it will not go past the beginning or end of the slider. This means that once the slider's handle has been acquired, the slider has practically an infinite size. Some sliders, like the one in Figure 3.6, use the distance of the finger to the main axis to increase precision. Moving the finger further outside of the slider increases the ratio between finger movement and the movement of the slider handle, so the handle only moves one pixel per N pixels dragged.

For sliders like this, acquiring the handle is the hardest part. As described earlier for buttons, the smaller the touch area, the harder it is to touch. What makes the slider handle even harder to get a hold of is the fact that its position is not constant. A variant to the slider that does not suffer from this problem, is the Picker, which was popularised by Apple as shown in Figure 3.5b. The picker in Figure 3.5b resembles a spinning reel, like on a slot machine or early digital clock, and shows the current value in the middle. Like a slider, a picker has an axis and a range and is operated by dragging it along the axis. Unlike sliders, pickers may or may not be finite and can even wrap ranges, such as time in Figure 3.5b, The biggest difference between pickers and sliders, is that with sliders, the handle moves, whereas with pickers, the whole reel spins and the area indicating the current value remains stationary. In a way, it's as if the handle remains in place, while the whole slider moves. As a consequence, the touchable area of a picker is much larger than just the handle, as shown in Figure 3.5. This makes pickers much easier to acquire than sliders, as their movement is relative and independent of the current value, but at the cost of losing some of the visual expressivity of sliders. The slider on the left makes it easy to see the volume is set to approximately 80%. On the dial on the right, it's not easy to see the whole range of possible values.

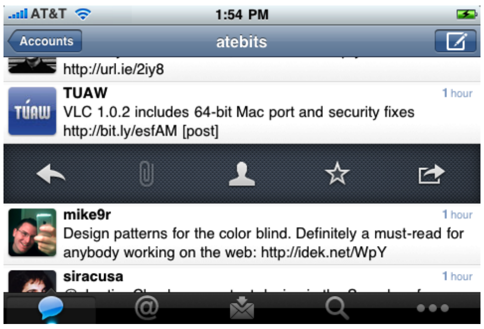
Adjusting a picker is similar to adjusting a slider: dragging one way goes up the range, the other way goes down. Apple's implementation of a picker is only suitable for rather crudely discretised continuous ranges – one step always occupies 44 vertical pixels – but pickers can be just as precise as sliders (and even more precise, as they can be infinitely long). However, when a lot of values are crammed in a short space, accuracy can suffer, which is why tricks like the one in Figure 3.6 are needed in the first place. One vulnerability that both sliders and tickers share is that releasing the control might change the value. It is not uncommon for touch screens to register some movement upon lifting the finger – especially optical and capacitive touch types. When this happens to a slider where one pixel on the screen corresponds to one value, the act of lifting the finger can cause the value to change. If this happens, the user has to re-acquire the control (which is hopefully still below the user's finger) and re-adjust the value. Pickers have an advantage over sliders here, as they are able to stretch out the range of values over more "tape" without increasing the amount of space the control takes up on the screen. Users can then use throttling (repeatedly dragging, lifting moving back and dragging again) and inertial scrolling gestures to move the extra distance.

Looking at the attentional demands of these two examples of continuous controls, the slider can be expected to be more demanding. First of all, a picker has a larger touch area, making it easier to acquire. Secondly, the position of a slider's handle is not constant; it depends on the value it is currently set to. The user needs to know the position of the handle in order to touch down on it, whereas a picker can be touched anywhere. Once either a slider or a picker has been acquired, however, they keep following the user's finger, even when it moves far out of the on-screen location of the control. This quality, which can be applied to other types of continous controls such as circular dials etc, makes it much easier to perform the "adjust value" stage without looking at the control. Finally, accidental adjustments on releasing the control are a problem for sensitive continuous controls. Correcting mistakes involves re-acquiring the control, possibly needing another glance.

3.3.4 Unembodied Discrete: Semaphoric Gestures

The controls in the previous two sections have all been similar in that they are embodied; they have a visual respresentation (often also used to communicate the control's current state or value) and clear affordances. As a result, they occupy a specific place on the virtual control panel, which in turn translates to a specific area of the physical touch panel where the user interacts with them. Gestural controls are different, because they lack this visual representation. Intead, the user interacts directly with content, a window or even the touch panel itself. Gestures that interact with content, such as drag & drop, or contextual gestures on list items such as used by Tweetie in Figure 3.7 may be very similar to interacting with embodied controls when the content is embodied. Dragging an icon may not be very different from dragging a slider. Other gestures may be independent of any Graphical User Interface. It is these unembodied controls that are of special interest for the purpose of inattentive operation.

Semaphoric gestures are cummunicative poses or motions with a symbolic meaning [34]. They have this meaning as a single unit: one gesture has exactly one meaning; it represents one bit of information, just like one button has one function. This type of gesture is therefore used for


AT&T 1:54 PM

Accounts atebits

http://url.ie/2iy8

TUAW 1 hour
VLC 1.0.2 includes 64-bit Mac port and security fixes
http://bit.ly/esfAM [post]

mike9r 1 hour
Design patterns for the color blind. Definitely a must-read for anybody working on the web: http://idek.net/WpY

siracusa 1 hour

Figure 3.7: Swiping horizontally over a Tweet in Tweetie reveals action controls. This semaphoric gesture is not embodied itself, but it has to be performed on a piece of content that is embodied.

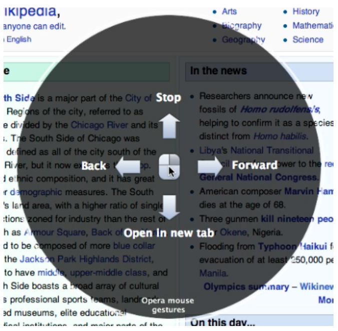


Figure 3.8: Opera helps the user learn gestures using a circular pie menu

discrete actions, commands and values.

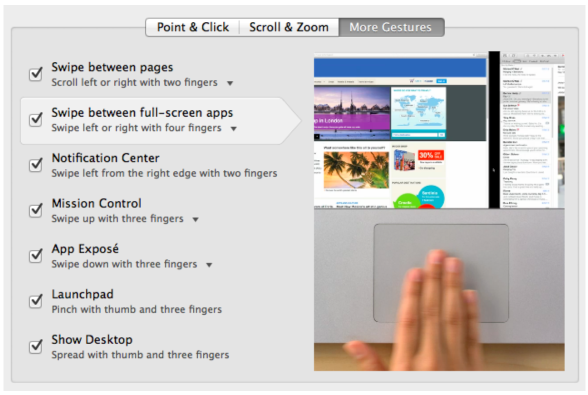
One of the earliest examples of the use screen-wide abstract semaphoric gestures can be found in the Opera Web Browser, which has had mouse gestures as of version 5.12 in 2001 [57]. Opera's mouse gestures allow the user to perform many of the most common actions, such as going back and forward in the browser history, opening and closing new tabs and windows and reloading the page – all by making symbolic motion patterns with the mouse while holding a mouse button. These gestures do not require interaction with any GUI controls on the screen; they can be performed anywhere on the screen as long as Opera is in the foreground.

The fact that gesture controls do not have a visual representation on the screen also means that it is harder for the user to know they are there. An on-screen button has a clear affordance: it needs to be pressed. Semaphoric gestures lack these natural affordances and require the user to memorise a language of gesture symbols. Solving the challenges of discoverability and learnability of gestures is still the topic of much research [3, 20, 30] and beyond the scope of this study. Still, it's important to be aware of the challenges.

For gestures that interact with content, the content itself may offer affordances. A good example of this is a slightly curled corner of a page, indicating that the user can flip the page by dragging from the corner inward. This helps the user *discover* the gesture. For more abstract cases, this is often not possible. In the case of Opera, some of Opera's gestures have a natural mapping to the commands they represent; a swipe left goes back in history, a swipe to the right goes forward. Other gestures require the user to draw L or U shapes and are not so obvious. Opera's solution for this is to reveal a sort of circular pie menu feedforward under the mouse (Figure 3.8) when the user holds the mouse button, but does not yet start to perform a gesture. The pie menu shows which gestures can be performed by moving in which directions and as the user moves the mouse, it shows what new gestures become available. Over time the user learns more gestures and is able to perform them without waiting for the feedforward menu to appear. This helps the user *learn* the gestures.

A final challenge and source of confusion is communicating succesful recognition of a gesture command. It's easy to understand what's going on if while performing a pagination gesture, the page curls up following the user's finger and flips over to reveal the next page, but when the system recognises a multitude of gestures that have more subtle effects, things can get confusing quickly, especially when the user misremembered a gesture or the system recognised a different gesture than the user intended. It is therefore important that the system somehow gives the user a way to *confirm* what gesture was recognised.

Although Opera pioneered abstract semaphoric gestures, they are all operated by a mouse



- ✓ Swipe between pages
  Scroll left or right with two fingers
- ✓ Swipe between full-screen apps
  Swipe left or right with four fingers
- ✓ Notification Center
  Swipe left from the right edge with two fingers
- ✓ Mission Control
  Swipe up with three fingers
- ✓ App Exposé
  Swipe down with three fingers
- ✓ Launchpad
  Pinch with thumb and three fingers
- ✓ Show Desktop
  Spread with thumb and three fingers

(a) Mac OS X "swipe between full-screen apps" gesture

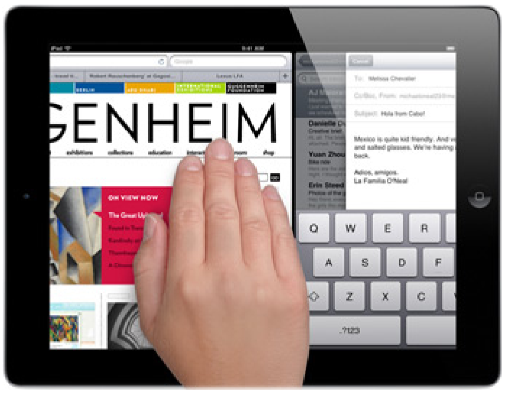

(b) iPad "swipe between apps" gesture

Figure 3.9: Semaphoric gestures in Apple's Mac OS X and iPad

or similar input device. The gesture is performed by the motion of the mouse pointer, not the hands, and it is triggered by pressing and holding the right mouse button. Capacitive surfaces, used in Touch Screens and Touch Pads, are able to recognise multiple touches and therefore the number of fingers that touches the surface can be used to trigger different gestures. This is extensively used in Mac OS X, where users can perform all kinds of commands by performing simple gestures, such as tapping or swiping left, right, down or up with two, three four or even five fingers (Figure 3.9a). Apple's iPad (Figure 3.9b) uses the same principle: using four or five fingers, system-wide multitasking gestures can be performed. On both Mac OS X and the iPad, these gestures are complementary; alternative ways to faster perform commands that can also be triggered with on-screen GUI controls. Gestures that are supported on both platforms have the same or similar meaning. On both the iPad and Mac OS X, four finger left/right swiping switches between applications or workspaces; swiping four fingers up shows running applications and five-finger pinching reveals the LaunchPad or SpringBoard from where applications can be launched. This way, semaphoric gestures learned on one can be used on the other, because they share a common gesture language.

What Apple's iPad and trackpad also show is that multi-touch semaphoric gestures are completely independent of Graphical User Interface components. When performing a "show desktop" gesture on a trackpad, it doesn't matter where the mouse pointer is currently located on the screen. When switching between applications on an iPad, it does not matter where on the screen the gesture is performed. In both cases, the entire capacitive surface acts as a control, regardless of what the user interface currently displays.

It is this property of semaporic gestures that makes them interesting for controls in an inattentive context. Rather than having to touch the screen in the location of the desired button or other control, gestures like on the iPad can start anywhere on the screen (or even outside of the screen). This makes such gestures much easier to perform without looking, as far less precise motion is required to land a touch in the right place. There is no on-screen control to acquire and all the phases of the control interaction model virtually merge together in one fluid motion. Using multiple fingers and motion that starts outside of the touch screen's active area are also relatively safe ways of distinguishing gestures from regular touch interaction with on-screen controls. Accidents can happen though and recovering from an accidentally activated button is likely to be more difficult if caused by a failed gesture than had it happened when the user was trying to press another button. Semaphoric gestures that are bound to on-screen content do not have this risk, but they also have a smaller surface area. In both cases, it remains a challenge how to make the user discover, learn and remember gestures, as well as confirm when the right gesture has been recognised by the system.

3.3.5 Unembodied Continuous: Manipulative Gestures

A final class of touch screen controls is made up by another type of gestures: manipulative gestures [34]. Like the semaphoric gestures discussed in the previous section, these controls lack a visual representation on the screen. They operate directly on content or the whole screen. Whereas semaphoric gestures are symbolic, and have a single meaning that can be mapped to a single command or a discrete value, manipulative gestures can be used to manipulate ranges of values. Manipulative gestures may work only on a small piece of on-screen content (in a way, the Picker control of Section 3.3.3 uses very localized drag gestures), but can also be used independent of the screen's contents, which opens up opportunities for inattentive operation as outlined in the previous section.

A common use case for manipulative gestures is browsing a map. Using one or two finger gestures, the user can interact directly with the map, rather than through on-screen buttons or sliders. Figure 3.10 shows the interface of Nokia's Ovi Maps application. There are still two discrete buttons to zoom the map in and out, but they are very small. The map itself spans nearly the whole screen and gestures can start anywhere on the map. The number of fingers touching the screen and the shape of the motion determines what value the gesture controls. Using two fingers, the zoomlevel of the map is controlled by changing the distance between the fingers. Moving the fingers wider apart zooms the map in and pinching them together zooms out. Dragging one finger along the surface pans the map in the X/Y plane. The distance the map pans is proportional to the distance the finger moves in the horizontal and vertical direction.

An interesting property of manipulative gestures is that they are able to adjust multiple values in a single integrated motion. If the map in the example would have used scroll bars, the user would have had to move the horizontal and vertical scroll bar separately. With the gesture, horizontal and vertical panning are done simultaneously, in a single gesture. Although integrating multiple degrees of adjustment freedom in a single control is not without problems [80, 24, 47, 48], it is also an opportunity to combine several controls into one. For inattentive controls, this is a big deal, because it prevents users from having to switch between controls, each of which would have to be acquired and released with the risk of errors.

Apart from problems that stem from the integration of multiple degrees of input freedom, manipulative gestures share the challenges of semaphoric gestures. Users need to somehow be able to discover which gestures are available (although there are generally fewer, simpler manipulative gestures), learn how they work and verify that the system is responding the way they expected. Additionally, whereas semaphoric gestures are either recognised correctly or not, manipulative gestures are only succesful when the user manages to adjust the parameters they manipulate to the desired values. As discussed earlier in Section 3.3.3, the human finger is not a very precise pointing instrument and there is always a tradeoff between speed and precision/accuracy. Increasing precision without sacrificing speed is beyond the scope of this study and still the topic of much research [55, 79, 56, 39], but it is important to be aware that the issue exists. Errors can also occur when the gesture is released. Starting over a gesture is likely easier than re-acquiring an on-screen control, but doing so will still incur a performance hit. Finally, the lack of visual representation makes it extra important that the system provides adequate feedback while the gesture is being performed. The user needs to be be aware of changes to the value as they occur, so the gesture can be performed accurately and efficiently.

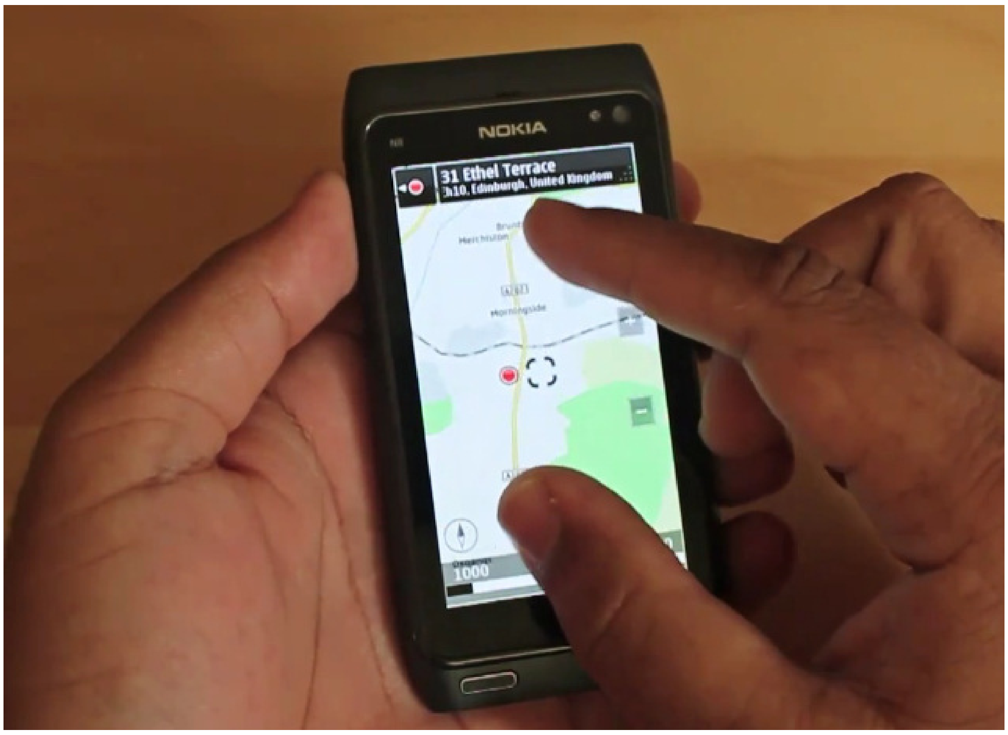


Figure 3.10: Zoom on Ovi Maps using a pinch gesture or buttons on the right.

3.4 The Problems of Inattentive Touch Interaction

Having taken a closer look at the nature of touch interaction and the different types of widgets and gestures in the design space of touch controls, it is clear that there are properties at all levels of interaction that pose challenges for inattentive operation. In this section, the main types of problems specific to inattentive interaction are identified and summarised.

3.4.1 Touch controls only respond to touch

Regardless of what technology they are based on, touch panels are almost without exception only sensitive to touch. They are not aware of the user until the user touches down on the screen and they lose track of the user's finger as soon as it is lifted. There is no "tracking" or "hover" state. This means the system is unable to provide feedback to the user during the first two phases of control interaction, where the control area and the target control are acquired. As a result, the boundaries between these two states become blurred. Users are on their own in reaching for the right control and the system does not know what the user is planning until it is too late to assist. Additionally, touch panels often lack a clear "home base" for users to base relative movements on, so it is hard to release controls without also losing grip of the control area.

3.4.2 Touch controls respond to any touch

The flip side of the fact that touch panels only respond to touch, is that they cannot distinguish between an correct touch and an incorrect touch, or between intentional touches and accidents.

This makes touch controls sensitive to the “Midas Touch” problem: the user cannot touch anything without potential side effects. Controls have to work around this problem by providing a “safe escape”: buttons do not respond when they are touched, they only get triggered when users touch down and lift their finger without moving outside of the button, but not all controls are able to offer the user a safe way out. As a result, users need to be very careful where they land their first touch and be precise in their movement if they want to avoid mistakes. Additionally, because any touch has effect, users cannot easily rest their hands on a touch panel without side effects. This aggravates the lack of a clear “home base” mentioned in the previous section.

3.4.3 Touch controls only respond to direct touch

As a result of the fact that touch controls are so sensitive to accidental activation, almost all common types of touch controls only respond to being touched directly. If the user intends to press a button but misses, the button will not respond to any touch until the touch is released and repeated on the button’s active surface. In other words, to be harder to activate by accident, controls are also harder to activate on purpose, requiring more precise action from the user. A physical potentiometer can be pushed up by approaching it from below and moving up until it slides along. A touch slider cannot support this interaction without increasing the risk of accidental changes.

3.4.4 Reliance on vision and proprioception

One of the biggest strengths of touch interfaces is also their biggest weakness: they are able to flexibly use a uniform surface for embodied controls. The same surface can be used for a grid of buttons, a row of sliders, interactive content or a mix of the above; sometimes even all of them, depending on the system’s current mode. The only real differences though, are the visual representation of the controls displayed on the touch screen and the way the system interprets the input it reads from the touch sensors. There are no outwardly perceivable differences between a touch panel displaying a button and one displaying a slider, other than the light emitted by the display.

Combined with the other properties of touch interaction, this means that during the first two phases of control interaction, only vision and properioception are available to assist in reaching for the controls and getting hold of the right one. In the initial (ballistic) phase of reaching for controls, it doesn’t make much of a difference whether the controls are real, physical things or virtual touch controls, but the closer the user gets to the controls, the more all the problems of touch interaction begin to act up. Rather than tangible 3D objects, the user needs to press down on a smooth piece of glass, in the right place indicated only by a visual representation of an embodied control. Up until the moment the user touches down on the screen, the only senses the user can rely on are vision and proprioception. As soon as the user has reached the screen and touches it, this can be felt using the tactile senses. The only thing this confirms, however, is that the screen has been touched; there is no inherent information on where the screen was touched and how the system interprets this. However, as described before, the first touch needs to be precise and mistakes are costly. Proprioception alone, however, is not accurate enough on its own. To accurately move to the right location, frequent visual feedback of the hand movement is needed to recalibrate the proprioceptive senses [75].

Switching between controls is not much better: it is possible to feel when the screen is no longer being touched and when the finger lands in a new place, but nothing in between. Compared to the way our fingers can feel the edges of the keys on a keyboard, a big part of the tactile feedback is missing from touch controls. Relying on proprioception is also much more difficult, as there is not much other sensory information available to help with the required recalibration. Combined with the fact that the hand position is in fact prone to drift (after all, it cannot rest in a home position), there are again not many options but to rely on vision.

The problem with vision is that unlike audio for example, it is a very focused sense. One can really only look at one thing at a time. Looking at one's hands and the controls thus means looking away from whatever else requires visual attention.

The part of interacting with touch controls that appears easiest to do without looking at one's hands, is adjusting the value. The user is now in constant contact with the touch panel, which can be felt and relied upon as a confirmation of still holding the control. Most controls, especially those of the continuous kind, are also very forgiving about any user movement as long as the user keeps touching the screen, which means less precision is required. Additionally, all movement is now at most two-dimensional, with the surface of the touch panel acting as a physical barrier and guide. Most importantly however, is that this part of interaction is likely to provide the most system output. A value changing in response to a slider being dragged is a good feedback mechanism to determine if the slider is being moved in the right direction and how much further it needs to be dragged.

Of the types of touch controls, the ones that are embodied rely the most on vision and proprioception. Gestures do not have a visual representation, so there is less need for the user to look.

3.5 The Performance of Touch Controls

There are many factors that influence the performance of touch controls. Despite all the problems of touch interfaces, they have actually been shown to compare quite favourably to other types of controls and inputs in certain circumstances.

In a study comparing numerical and textual input on two types of touch panels (resistive and capacitive) to physical hard buttons, Lee and Zhai [41] found that not only did users prefer touch input (especially capacitive), they were also significantly faster using soft keys than hard keys. Only with very narrow soft keys did participants begin to make more errors on the capacitive screen. The audio feedback provided by the keyboards of devices like the iPhone was enough of a confirmation for users to recognise successful button presses. The authors do recognise that their study was conducted with the full attention of the users and that more work is needed to understand the consequences of divided attention. Without any sort of feedback, however, text input is much slower on a touch keyboard than a keyboard with physical keys [31].

When touch input is used to allow direct interaction with on-screen elements that would otherwise be manipulated indirectly by hardware knobs or other controls, the results are not so black and white. Rogers et al. had participants of different age groups operate a complex "entertainment system" interface using both touch and hardware controls [65] They found that for tasks that involved mostly ballistic movements, such as moving a slider a long distance; pointing tasks and discrete tasks, touch was significantly faster, whereas for repetitive tasks, a knob (rotary encoder) was faster. These differences were strongest for younger users. Older users were more divided and generally preferred the physical controls.

For certain tasks, such as those that require direct interaction and pointing, touch can thus be quite appropriate. Being able to type on an iPhone's small touch keyboard, although it might seem difficult, is much faster than typing on a Wii's on-screen keyboard, because the user can press the keys directly, rather than having to control an on-screen pointer over the keys [43]. When touch interfaces are used as indirect controls, the results are not always so positive. For an interface to control a robot, Micire et al. [51] emulated a joy stick interface on a touch screen.

Users got confused and tried to use unsupported interaction styles. In a comparison between two versions of the same computer game, one designed for the touch-only iPhone 3G and one for the Nintendo DS Lite with physical controls, Zaman et al. [78] found that participants performed much worse using the touch controls (shown in Figure 3.11, needing more time to complete tasks and making more mistakes, even after an hour of practice.

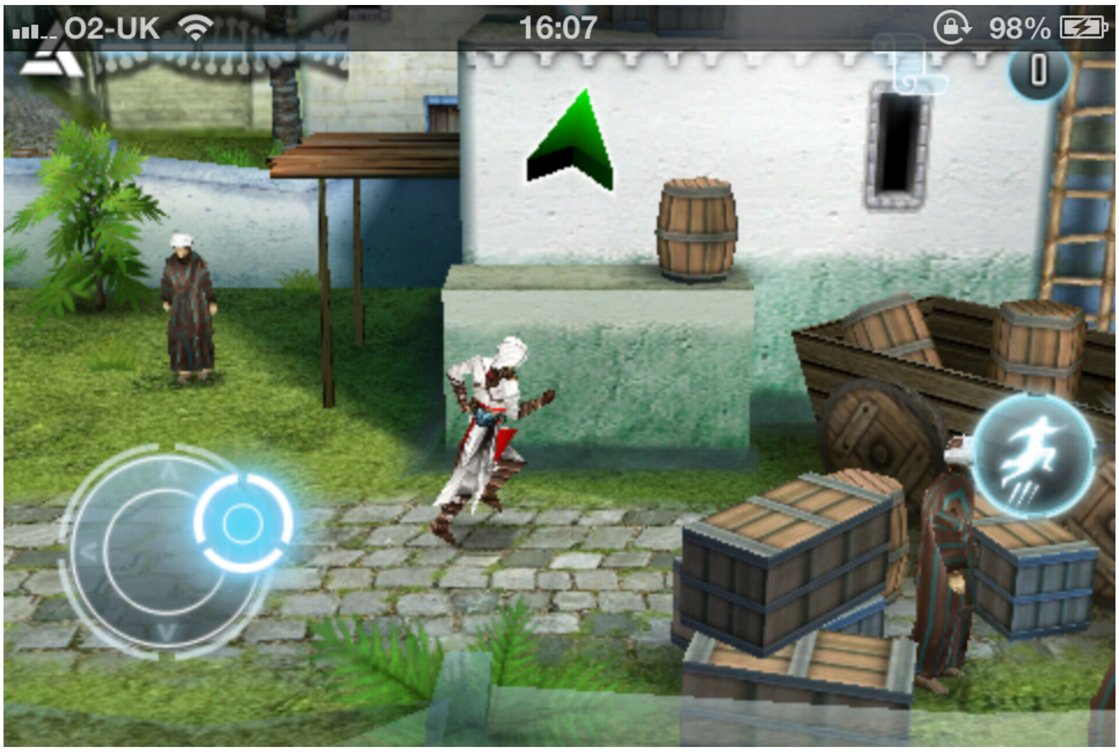


Figure 3.11: The touch version of Assassin's Creed, the game used in the study by Zaman et al. [78]. Players control the character by placing their thumb in the circle in the bottom left corner and moving the virtual joystick in the desired direction.

Part II

Improving Inattentive Touch Interaction

Chapter 4

Approaches and Solutions

The various problems of touch screen interaction in an inattentive context have been approached from a wide range of angles. Some solutions are aimed at the larger issue of touch screen usability in general, others have more specific or limited uses. In this chapter, these approaches and solutions are discussed in the light of the four main problems identified in Section 3.4. The problems caused by the fact that many touch controls only respond to (direct) touch, lack a hover or tracking state and cannot distinguish between intentional and accidental touch can be mitigated by adding a tracking state or changing the controls so they require less accuracy. Adding feedback other than the visual output already offered by touch screens, can further decrease the reliance on vision and proprioception.

4.1 Increase Accuracy: Restoring a tracking state

A good deal of the problems of touch interfaces can be attributed to the lack of a tracking or hover state. Rather than moving a cursor or pointer to an on-screen control, touch controls are operated by moving one or more fingers directly to the right location. As touch panels tend to be exclusively sensitive to touch input, this happens outside of the system's awareness, which in turn means the system cannot provide feedback or other output that can be used to guide the user. Unfortunately, as soon as the user touches the screen – and the system becomes aware of the user's position – it is too late for feedback. A button may already have been pressed, or a slider may already have moved.

There are roughly two ways to restore a tracking state to touch input: make the system aware of the position of the user's movement before he or she touches the screen or delay the activation of controls past the first touch.

The first approach involves augmenting the input so that the system can sense the user's hand position when it is in proximity of the touch sensitive surface. This literally introduces a hover state as it allows the system to sense when the user is hovering above active areas or controls and respond with feedback. Although this has not yet been done widely, preliminary experiments with hover-enabled touch interfaces, such as the one by Echtler et al. [14] and Han et al. [23] have shown improvements in the accuracy and performance of input. Another exploration of depth-sensing bidrectional screens by Hirsch et al. [29] showed promise for new interaction opportunities.

When a system provides appropriate feedback to users' motions above and on the touch surface, it can be much easier to use. An extreme example is the Visual Touchpad [46], which works with separate vision-based touch area for input and TFT display for output. Rather than

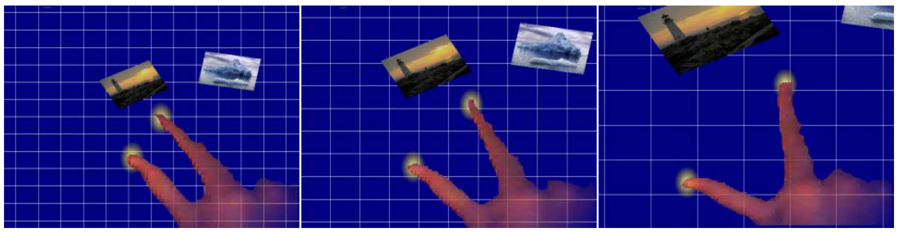


Figure 4.1: The Visual Touchpad [46] shows the user's hands on a separate display

touching the screen directly, the user interacts with it indirectly through the Visual Touchpad. The user's hands are tracked using two cameras, even when they are above the surface and their position is being shown on the other screen, as shown in Figure 4.1

All of these directions, however, require changes to the hardware. One relies on detecting a shadow cast by a light source positioned above the user's hand and the other uses a special type of depth-sensing LCD. Touch panels capable of recognising a hovering hand are not yet readily available.

The other main approach for restoring tracking, is adding an extra step after touching the screen to activate controls. This can be done using special hardware or by changing the interaction itself. The BlackBerry Storm 9500 smartphone was the first consumer phone to use the hardware approach [62]. Using a pressure sensitive switch behind the touch sensitive screen, the Storm was able to distinguish between a touch and a press. The user could select controls by touching them and activate them by pressing on the screen until the pressure sensor "clicked". In a study comparing the BlackBerry Storm to the iPhone and other touch smartphones, participants found the BlackBerry easiest to use because its interface highlights the selected option so users can confirm activation and the clicking of the touch screen provides tactile and auditory feedback to user input [42].

Adding a tracking state without additional hardware is also possible, but requires changes to the semantics of the touch language used by the interface, so that a first touch does not always immediately triggers activation. An example of this is creating separate "select" and "activate" gestures, for example by having a touch with one finger mean "select", and a tap with the other finger while holding the first finger on a control mean "activate" [56]. In a series of experiments, Hofmeester and Wolfe [30] tried various alternatives to tapping as an interaction methaphor. They replaced actions commonly performed by tapping on buttons, such as opening or closing windows, going back etc. with swipe gestures. They found that as long as the controls were not too similar to existing tap-based controls, users picked up on the new gesture-based language quickly and were not slower in performing functions. Changing the way people expect to interact with touch interfaces, however, is not an easy task and it may not always be possible to break with existing conventions.

These two approaches are not mutually exclusive and they each solve a slightly different problem. Whereas making the system aware of the user's motion before they touch the screen solves the problem that traditional touch panels *only* respond to touch (Section 3.4.1), moving the activation of controls past the first touch solves the problem that touch panels respond to *any* touch (Section 3.4.2). Whatever techinque is used, restoring a tracking state primarily helps in the early phases of control interaction. Adding a tracking or hover state restores a clearer separation between acquiring the control area (either hovering above a depth-sensitive panel or touching a tracking-enabled touch screen), acquiring a control and using it to adjust values. The system can give feedback earlier, making it easier to land in the right spot, or delays the moment where precision is required until after it can give feedback.

4.2 Decrease Required Precision of Movement

Pushing the acquire control phase past the first touch allows the system to assist the user with feedback in making the precise motions needed. Another set of approaches aims to reduce the amount of precision required to acquire controls altogether. As discussed in Section 3.3.3, a Picker control is easier to acquire than a Slider, because it has a larger surface area and a constant position. Similarly, larger buttons and controls that are laid out horizontally or vertically require less precision than smaller buttons or controls that require diagonal movement. Starting with a gesture can be even easier, gestures can be independent of specific areas on the screen and start anywhere, making the "acquire control" step trivial.

Acquiring the control area and embodied controls itself require the most accuracy: as soon as a slider or a picker has been acquired, for example, the user is no longer restricted to staying within the control's active area and can move around the whole screen, which is not that different from a gesture.

An interesting area for inattentive interaction lies between gestures and embodied touch controls. These controls do have a visual representation on the touch screen, but rather than requiring the user to move to the controls, the controls position themself based on the user's position. An example of such an interface is pieTouch [15]. This pie menu interface was designed by BMW for in-car information systems and is triggered by a press-and-hold gesture in the middle of the screen. The pie menu then appears around the user's finger, from where relative motion activates the menu items. Using pieTouch, participants were able to perform simple editing tasks faster while driving a simulation.

Another example can be found in the The Visual Touchpad [46]. Placing five fingers on the touch pad causes a virtual keyboard to appear under the user's fingers, positioned such that the keys that form the "home base" are aligned with the hands. These hybrid gestures/controls rely on some form of predefined semaphoric gesture to trigger, such as the tap-to hold or five-finger touch gestures in the examples, but then they act like regular visible embodied controls that have already been acquired.

4.3 Add Nonvisual Feedback

A lot of the problems of touch controls can be traced back to the fact that touch screens are inherently visually oriented. They can contain all kinds of different controls and the only cues to how they work are visual. As described in Section 3.4.4, this leaves them relying predominantly on vision and proprioception for their operation, because those are the only available modalities. Adding other types of feedback, such as tactile and auditory feedback, can open up new ways for users to get information about the position of controls, their current state and any value changes made and free up the visual channel. The human body is equipped to deal with various sensory inputs, including taste and smell. For feedback during control interaction, the auditory and tactile modalities have been most explored.

The most obvious approach for restoring some of the nonvisual properties of physical controls to a touch screen, is placing actual physical controls on top of it. A good example of this are the SLAP widgets by Weiss et al. [74]. SLAP widgets are fully haptic, three-dimensional controls made of translucent silicone. They are placed on top of a vision-based touch table, which is able to read input from the widgets through changes in their reflective properties. The widgets, which include rotary knobs, buttons, sliders and even a full keyboard, have the same physical properties of the electronic components they are based on. This means they have the same haptic, tactile and auditory feedback as normal physical controls, but also many of their limitations. The visual appearance of the controls is determined by the output of the underlying touch screen. However, although they can be positioned freely anywhere on the surface by the user, this is beyond the system's control. Similarly, the individual controls are bound by their physical limitations. A button cannot become something else and a slider that has been moved to a particular position will stay there unless it is moved by the user.

A different, but related approach, is to place a raised haptic overlay on top of the touch screen, which includes hints to the positions of on-screen controls [49]. This places constraints on the placement of controls – they have to align with the layout of the haptic overlay – but the meaning of the controls can still be controlled by the system. Dynamically changable haptic overlays are still the subject of ongoing research. Harrison et al. [26] compared a touch interface with a pneumatically controlled overlay to a flat touch panel and a control panel with real physical buttons, and found that it performed as good as physical controls, requiring less glances at the controls than a flat touch panel without overlay. Although the buttons of this overlay can be raised or lowered by the system to make place for gesture-based interaction, their shape and position is fixed. This also means that haptic overlays primarily help with the acquisition of controls, making buttons and other controls easier to find, but not so much in the other stages of their operation.

The main challenge in adding nonvisual feedback to touch interfaces is doing it in a way that does not detract from the flexibility that makes touch screens so special in the first place. The system should be able to control the nonvisual feedback as easily as it can control the visual output, which requires keeping the touch screen flat and uniform. Whereas the previously mentioned approaches added real tactile and auditory properties to the touch panel, a good deal of these can be simulated using various types of transducers. Audio can be synthesized by using magnetic, piezoelectric and other types of loudspeakers. Similarly, the tactile senses can be stimulated using vibro-mechanical, pneumatic, piezoelectric and electrocutaneous actuators [50]. Both the tactile and the audible components of the "click" of a physical button can be simulated this way, to serve as a confirmation of succesful activation without visual attention. In fact, adding either of these feedback modalities has been found to already bring the performance of touch screen soft buttons (measured as speed and error-rate) to the level of hard buttons [37, 41]. Given the context of this study – using touch screens in situations where limited attention is available for the controls themselves – tactile feedback is the most appropriate of the two. The tactile senses as a whole are more private and less likely to be occupied or interfered with by external impulses [52], which makes tactile feedback especially attractive. The tactile senses are also very fast; humans can detect stimuli as little as 5 ms apart [61] – five times faster than vision – and respond instantly. For example, we can sense that something we are holding is slipping and immediately adjust our grip to prevent it from falling [32].

With the exception of special types of actuators that are worn on the body, tactile feedback is most likely to be transmitted through the touch panel itself. It is easy to augment most touch panels with a tactile actuator and the touch surface can act as a conductor for tactile signals. This is appropriate, because the tips of the fingers, which are used to touch the screen, are also among the parts of the body that are most sensitive to touch feedback [52]. As a consequence, though, this feedback is only available while the user is touching the screen. This means that, unless it is combined with other techniques in this section, tactile feedback is of little to no help while acquiring controls or for getting feedback about what happens after the touch is released, as depicted in Figure 4.2.

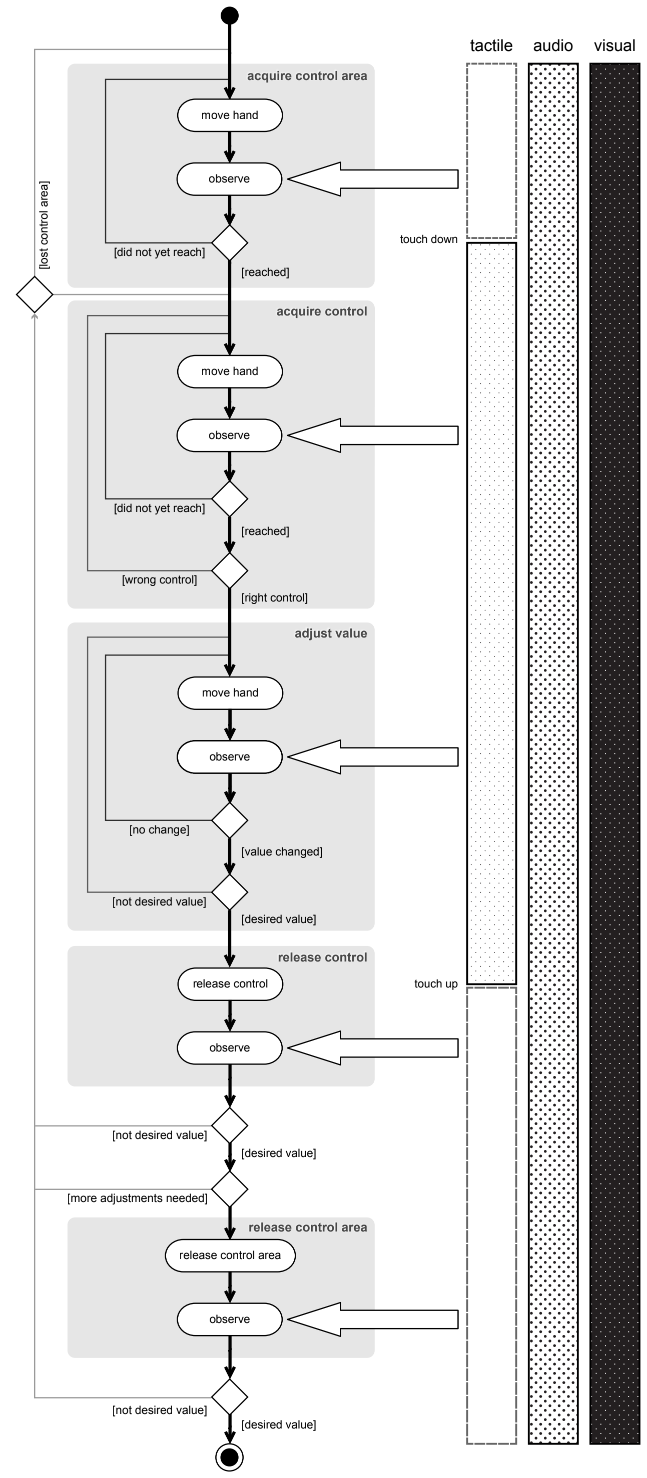


Figure 4.2: Modalities available for feedback at each phase of interaction

Chapter 5

A Tactile Solution

After looking at the different kinds of touch controls and the problems associated with using them in an inattentive context in Chapter 3 and the various solutions that exist for these problems in Chapter 4, it is clear that one of the reasons touch controls require so much visual atention, is because vision is often the only sensory channel that can be relied on. Tactile feedback is often added to touch interfaces to relieve the visual channel, but so far its use has often been limited to the confirmation the activation of buttons and other controls. While the value of tactile feedback for this purpose should not be underestimated – tactile confirmation of control activation significantly increased the speed of operation and reduced errors in several studies [8, 33, 41, 60, 37] – it is also fairly limited. Ideally, tactile feedback helps users to not just recognise that they successfully acquired or adjusted a control, but also find the controls in the first place and feel more in touch with them while they are being manipulated. It is these other tactile properties that are of special interest in this study.

Looking at real-world physical controls and their virtual touch counterparts, A big difference between them is that while touch controls have only a visual representation, physical controls also have several tactile and audible properties that help in their operation. Some of these properties only produce feedback in response to actions by the user or the system such as the clicking of a button in response to it being pressed. This type of feedback is often referred to as active feedback [25]. Others tactile features, such as the contours of a switch or the texture of the materials, are passive and can only be perceived by active exploration by the user. All of these features, however, can help the user in a different way. Designers of systems that are intended to be used frequently and eyes-free make smart use of materials to include clear cues to the user about where the controls are and how they are used.

The notebook computer is perhaps the most familiar example. It has a keyboard and a touchpad or other input and users expect to be able to input text and move the cursor while looking at the computer's screen. To facilitate this type of usage, the different components are made as easy as possible to find. Figure 5.1a shows the keyboard and touchpad of a MacBook Air. The surface on which both can be found is made of brushed aluminium that is slightly rough to the touch. The touchpad is made of a different material that is smooth and lies slightly lower than the surrounding surface, as a result of which it is surrounded by a distinctly sharp raised edge. The keys of the keyboard are also made of a smooth plastic, but unlike the touchpad, they stick out above the surface and are mounted on light springs, which causes them to slightly budge when touched and produce a light click when pressed more strongly. There is just enough space between the keys to make it easy to feel when the finger slips off one key and onto the next. Special keys can be distinguished from the rest by having a slightly different shape and,

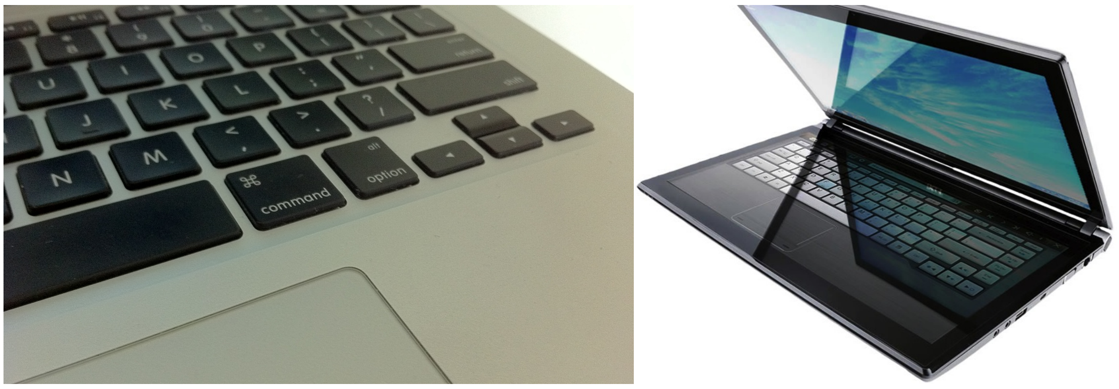

(a) MacBook Air (b) Acer Iconia

Figure 5.1: Physical and virtual keyboards and touchpads

in the case of the F and J keys important for establishing a “home base”, a distinct bump. To cordon off the keyboard area from the rest, it is surrounded by a sharp curve in the aluminium surface, which makes it easy to find from any direction. Although these edges, curves and lines can also be seen by users, it is more important that they can be felt. The keyboard area of an Acer Iconia, depicted in Figure 5.1b, displays all the same edges, ridges, textures and lines as the MacBook in Figure 5.1a, but it is arguably harder to use due to the fact that they are all just pixels below a smooth glass touch surface. While the keyboard of the Iconia can be configured to produce clicking sounds, and the user can easily see the mouse start and stop moving when the edges of the virtual touch pad are reached, this is hardly sufficient. As the preliminary results of the research with HapTouch [64] suggest, the passive properties [25] of the controls matter just as much as the active ones. In this case, the feedback the user gets from the edges around the touch pad are easier to perceive than the absence of change in the movement of the mouse pointer.

So far, most research has focused on the active types of tactile feedback, that help users recognise they successfully pressed buttons, dragged sliders and so on. This is useful during the later phases of control interaction, where values are changed, but does not aid much in discovering and acquiring those controls. In this study, both active and passive kinds tactile feedback are added to touch controls, so users can rely on it during a bigger portion of their operation and use them with less visual attention. This chapter describes the design, implementation and evaluation of this novel solution.

5.1 Related Work

The use of dynamic tactile feedback in touch and other interfaces has been the subject of many studies. What follows is a brief overview of related work concerning tactile feedback that was intended or could be used to improve inattentive operation of touch controls.

As noted before, a consequence of embedding a tactile actuator in a touch panel is that its output can only be felt while the user is touching the screen. This means that while interacting with traditional GUI controls such as sliders, scroll bars and menus, there is naturally most opportunity for feedback while the user is dragging or performing other gestures. In an early exploration of using tactile feedback with gestures, Poupyrev et al. noted that the combination of gestures and tactile feedback resulted in a strong feeling of physicality in interaction [61, 60].

A tight tactile feedback loop during the “adjust value” phase (Figure 4.2) can help users keep track of value adjustments without constant visual polling. An area where such a feedback loop is especially important is that of digital music instruments. Open-air gestural musical instruments, such as the theremin and modern derivatives, are especially hard to play because the performer does not get timely feedback from the instrument [66]. They rely on the use of vision (to determine they are within the instrument’s range), proprioception and audio for their operation. Compared to traditional instruments, they lack tactile and haptic information during the performance, resulting in the performer feeling out of touch. Reintroducing tactile feedback proved promising in this regard [66]. In a study about digital touch flutes, the instruments were described as cold and distant when they did not vibrate in the hands of the performer like a real instrument [6]. Synthesizing artificial feedback made the touch flute feel more warm and familiar.

In their work on Semfeel [77], Yatani and Truong added multiple vibrotactile actuators to the back of a mobile phone, which allowed so-called vibration patterns to be generated. By activating the actuators in sequence, different circular and linear patterns could be produced, to which different meanings could then be ascribed. Compared to a system with no feedback or only a single dimension of tactile feedback, this richer semantic tactile feedback lead to higher performance and lower error-rates when used to input numbers in an eyes-free setting.

Earlier work into richer tactile feedback that also helps users explore available actions often used indirect interaction or other special hardware. Bongers and van der Veer equipped a standard computer mouse with a tactile element to augment regular GUI elements with tactile feedback [7]. Participants operated standard controls such as menus, buttons and sliders, which were equipped with tactile and auditory responses to hovering and dragging. Interestingly, they found that while in some cases the participants’ response time went significantly down when tactile feedback was present, in other cases it went slightly up. This may be explained by the fact that – although participants indicated they felt the feedback was effective – it may cause some users to perceive the tactile feedback as resistance. The Blind Audio Tactile Mapping System (BATS) used adapted joysticks with tactile and spatial audio feedback to allow children with visual impairments to succesfully navigate historical geographical maps by controlling a virtual pointer cursor [58]. Tactile feedback of much higher fidelity can be found in The Virtual Active Touch system [76]. This system allows users to explore textures by moving a cursor across a virtual surface using a pointing stick interface.

The Ubi-Pen [39] is a tactile-augmented stylus that allows more direct interaction with on-screen GUI components. It has a speaker for sound, a vibrator motor for vibrotactile feedback and a magnetic element that can generate impact signals. Adding tactile feedback improved the performance of various tasks, such as text manipulation, and drag and drop file handling and was received positively during long-term field tests.

The Artex system applies artificial textures to different areas on a mobile phone’s touch screen in order to allow users to browse information on the screen non-visually [13]. The user can drag a finger over the touch screen to feel the textures. Unfortunately, the authors did not evaluate the effectiveness of this approach. HapTouch [64] also adds tactile cues to the location of controls on a touch screen. In this system, tactile feedback is fired when the user moves in and out of controls. Combined with using a pressure-sensitive touch panel that can distinguish between light touch and a full press, the authors intended to restore a tracking state. During experimental evaluation of the HapTouch system in an in-vehicle context, the tactile feedback appeared to reduce the amount of errors made (particularly for small controls), reduce the time needed to operate small controls and have less negative impact on driving performance compared to a normal touch interface that did not distinguish between touch and press and had no feedback. Although the

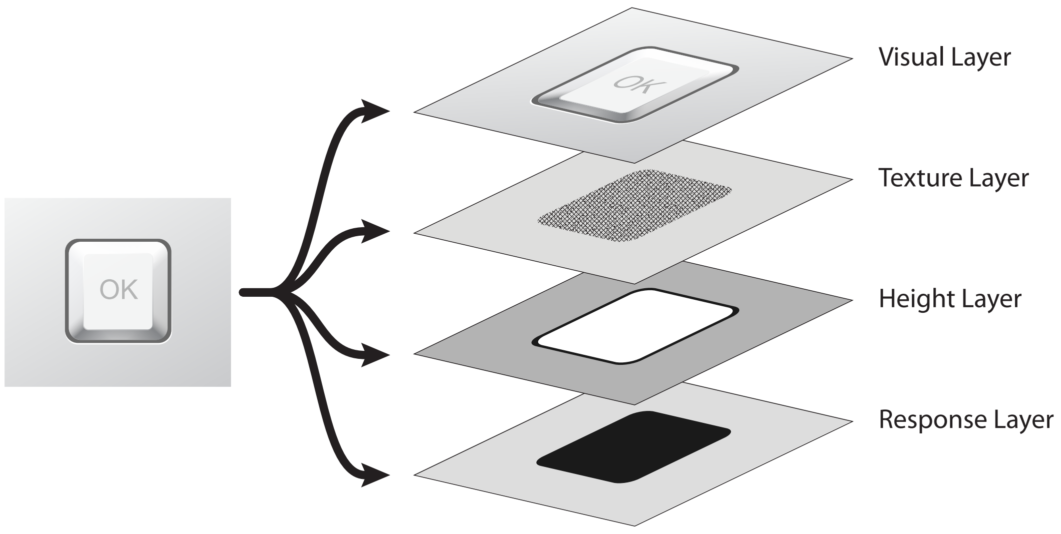


Figure 5.2: Tactile properties for a button in a touch interface

authors did identify some flaws in their experiment design, and some negative effects were also observed, these preliminary results are promising and suggest that adding tactile feedback can help users operate touch controls with less distraction.

5.2 Designing Rich Tactile Feedback

As described in the introduction of this chapter, the essence of the proposed approach lies in adding richer tactile feedback to touch controls in order to make the different stages of touch control interaction easier to perform without looking. This means that in addition to the *visual representation* user interface components already have in current touch screen interfaces, they are extended with a simulated *tactile representation* consisting of multiple properties. The central idea is that by making touch controls more like physical controls users are invited to rely more on their tactile senses and less on vision to explore, acquire and manipulate controls. This in turn frees up visual attention for more important tasks.

From the analysis of the problems of inattentive touch interaction in Section 3.4, it appeared that different types of controls have different problems. Embodied controls, that occupy a specific space on the touch interface, are mostly difficult to acquire. Acquiring these controls requires a precise touch in the right place, after which subsequent touch movements only affect the active control. The flip side of this is that if the screen is touched in a place where there is no control, subsequent movement is guaranteed to have no effect on any of the controls on the interface, as they would have to be acquired first. This property of touch interfaces can be taken advantage of when designing with tactile feedback for eyes-free operation, because in a way it restores a sort of tracking state, where the user can explore the interface while the system is aware of the position of the user's finger. It allows for passive feedback, e.g. feedback that emerges from properties of the interface when explored by the user rather than active system output, which can help during the earlier stages of touch interaction.

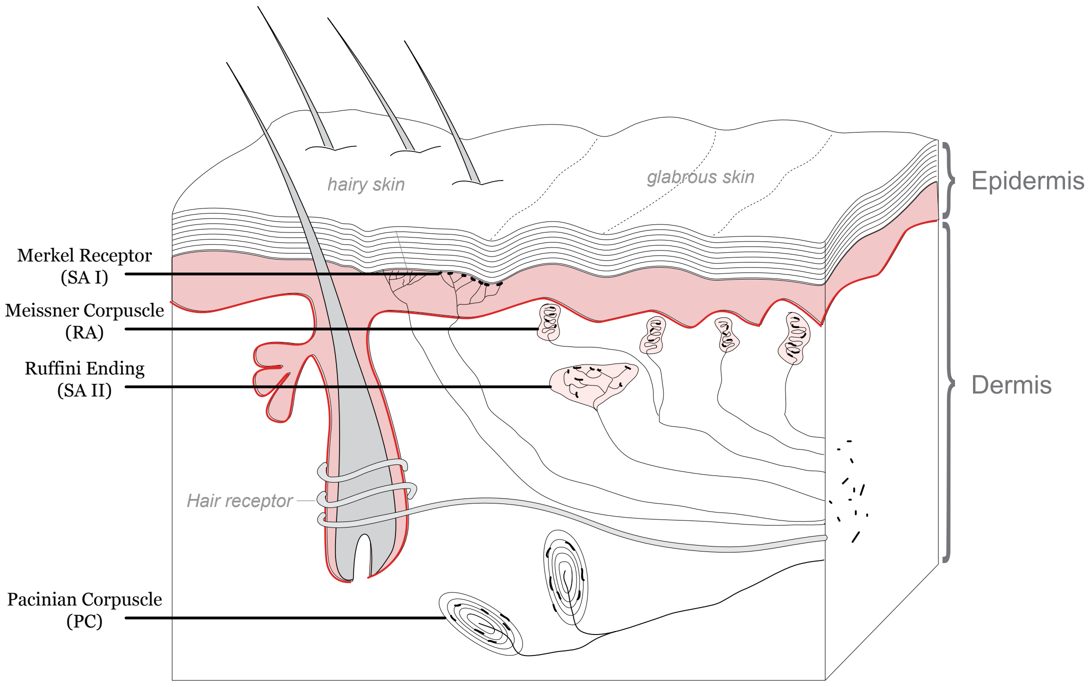


Figure 5.3: The different mechanoreceptors in the skin. Image adapted from Thomas Haslwanter, licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

The two passive tactile properties added in this design are *texture* and *height* – both common in physical interfaces. In order to benefit from users' existing expecations about touch interfaces, the behaviour of touch controls was kept the same. Buttons are activated when the user touches down on them and then lifts without moving outside, sliders track the touch outside of their boundaries and so on. This means that the different layers of feedback can be added on top of any existing interface and do not require learning new interaction methods. As a result, activating a control that has been found through passive feedback requires the user to briefly lift their finger and touch back in place.

Active feedback, in the shape of tactile responses, was added too. It is useful for recognising successful control activations and value changes. For discrete controls, those are likely the same, but sliders and other continuous controls can be used to make many successive value changes. Figure 5.2 shows how a single control is composed of not just visual information, but also has a *texture*, *height*-profile and *response* characteristics. Each of these properties helps the user to find, identify and operate the controls. When these properties are added throughout the interface, a coherent tactile experience can be created.

5.2.1 The Tactile Senses

Designing for the tactile senses requires at least some understanding of how they work. As a whole, our tactile senses are quite fast; we can distinguish between stimuli as little as 5 ms apart [61], which allows us to act fast and not drop things. They are also pretty precise. The skin in our fingers can recognise pointed indentations as close as 1mm apart [32], which makes Braille possible.

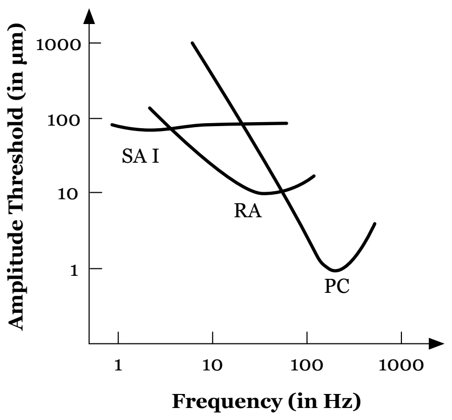


Figure 5.4: The sensitivity thresholds of mechanoreceptors targeted in this study, adapted from Konyo et al. [36]

Human fingertips – among the most sensitive parts of the body – are covered with glabrous (i.e. non-hairy) skin [52]. In this skin, shown in Figure 5.3, four types of mechanoreceptors (cutaneous nerve fibres) have been identified – each with their own role in the sensation of touch and each with their own distinct properties. They each pick up different signals and transmit impulses to the brain via the central nervous system that result in the perception of touch. An extensive review of the tactile senses can be found in Myles and Binseel, 2007 [52]. What follows is a simplified summary of the relevant properties of the different receptors.

Mechanoreceptors can be either *rapidly adapting* or *slowly adapting*. Slowly adapting receptors, such as the Merkel receptors respond slowly to stimulation and continue firing as long as the stimulus continues. There are two types of slowly adapting (SA) receptors: the Merkel Receptors (SA I) and the Ruffini Endings (SA II). Rapidly adapting receptors respond quickly to stimulation, but they quickly adapt and stop firing if the stimulus remains constant. The Meissner Corpuscle (RA) and the Pacinian Corpuscle (PC) are both rapidly adapting, one lies just below the surface and the other deep in the skin tissue.

The different receptors are each specialised in picking up different signals. The Meissner Corpuscle (RA) mostly picks up localised vibration and motion. The Pacinian Corpuscle (PC), on the other hand, responds to a larger area due to being located deeper in the skin and is mostly sensitive to pressure and temperature. The slowly adapting Merkel cells (SA I) perceive tactile form and roughness. Ruffini Endings SA II, finally, detect skin deformation and deep tension.

Due to their nature, and the fact that they lie embedded in the skin, mechanoreceptors (with the exception of the Ruffini Endings) operate on relatively indirect impulses. As a result, it is possible to stimulate the different receptors selectively, through alternative means such as vibrotactile or electrical stimulation [36]. Each receptor is sensitive to a specific range of signal frequencies and amplitudes, as shown in Figure 5.4. The Meissner Corpuscle (RA) is most sensitive to frequencies between 10 and 100 Hertz. The Pacinian Corpuscle (PC) on the other hand, is sensitive to a higher and wider range of frequencies; between 40 and 800 Hertz, at lower amplitudes. SA I (Merkel Receptor), is most sensitive to very low frequencies, generally below 100 Hertz and as low as 0.4 Hertz. Although these ranges overlap, each receptor has frequencies for which it is most sensitive. By applying artificial impulses within those specific ranges, sensations of vibration, motion, pressure and roughness can be evoked, as Konyo et al. demonstrated experimentally [36]. This way, texture, height and other material properties can be simulated.

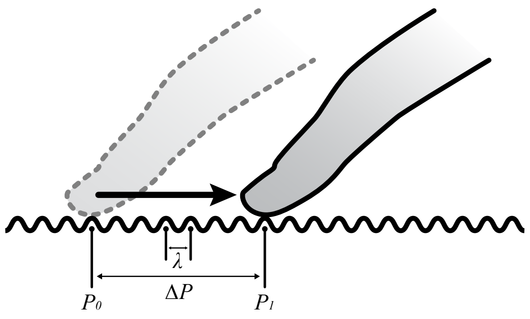


Figure 5.5: Moving the finger over a bumpy surface

5.2.2 Simulated Texture

In the real world, every material has a set of surface characteristics that define its texture. Surfaces can be rough or fine, smooth or sticky. These properties can be sensed by moving the skin, in this case a finger, across the surface and help to identify and distinguish between materials. While moving the finger over a surface, the small surface details and friction of that surface trigger mechanoreceptors in the skin that subsequently translate to perception of the texture in the brain [67]. Using a high-precision wearable ciliary device (that used 60 tiny pins, or cilia, to stimulate the skin directly), Konyo et al. [36] were able to control perceived roughness, softness and friction by selectively stimulating the different mechanoreceptors in the skin.

For this study, less precise hardware was available, and the main focus was placed on simulating roughness. As depicted in Figure 5.5, perceived roughness is a function of movement speed and perceived impulses. Movement speed is sensed as velocity through amplitude by the rapidly adapting Meissner corpuscle (RA) mechanoreceptors and changes in the pressure on the skin are sensed by the Merkel copuscle (SA I) [36]. Together, this translates into the perception of texture. If the properties of the material and the movement of the finger over a surface is known, the expected impulses can be calculated and suplied to these mechanoreceptors through alternate means. To understand how these impulses are normally produced, consider a bumpy surface like in Figure 5.5. If the distance between individual bumps is λ, moving a finger over the bumps with velocity t, causes the skin to vibrate at a frequency $f = v/\lambda$. In other words, rough surfaces induce lower frequencies than smooth surfaces, relative to the velocity. Turning this around, if the velocity of the finger is known, the frequency of the impulses that a smooth or a rough surface would produce can be calculated and artificially generated using a tactile transducer. The RA receptors are most sensitive to frequencies between 10 and 100 Hz [52].

5.2.3 Simulated Height

The design of the MacBook in Figure 5.1a makes extensive use of distinct height differences, edges and curves to separate and group controls. As a touch panel is inherently flat, true height differences are a bit more difficult to simulate. The mechanoreceptors that are sensitive to surface curvature are rather crude. In previous research, the illusion of smooth or sharp edges was successfully created through either slow or rapid changes in feedback patterns [67]. This too is a function of motion and surface properties, but also of the direction of the movement. Moving perpendicularly into sharp curve causes rapid changes, whereas moving along an edge or a smooth curve causes slow changes.

In this study, height and surface curvature were only crudely approximated to indicate the edges of controls. Moving into and out of controls produced a predefined rapid response, whereas the direction of movement was ignored.

5.2.4 Tactile Response

The response layer is distinct from the texture and height layers in that it is not limited to being perceived through motion. There are two types of response: passive and active response.

The passive response describes how the virtual material responds to touch. A good example of this type of response is hardness/softness [1]. Hardness can be sensed using the static pressure mechanoreceptors SA I. These mechanoreceptors are most sensitive to very low frequencies, between 0.4 and 100 Hz, so using very low frequencies of around 5Hz, SA I can be triggered without triggering the other receptors. This will be perceived as soft [36], whereas a touch panel normally feels hard.

Controls may also have an active response, which is the product of changes in the system. A button being pressed may click, a slider handle being dragged may pop once for each value it crosses and so on. These active responses are less emergent properties of the system and more programmed behaviours. The tactile responses used in this study are described in detail in Section 5.4.3.

5.3 Prototype Platform

To try out the design concepts outlined in the previous section, a functional prototype was developed. The user interface of this prototype was intimately tied to the test-set up and task used in the evaluation. Therefore it will be discussed in detail in Section 5.4.3. This section describes the prototype platform – the canvas on which the interface was drawn – and the choices that were made regarding the hardware and software used.

5.3.1 Tactile Actuator

In order to add tactile feedback to a touch screen interface, an actuator is needed that can produce the type of signals our fingers can feel. There are many kinds of actuators, each of them with their own properties. This section is by no means a full overview of all existing actuators – see for example McGrath et al. [50] for that – but rather a summary of the options considered for this prototype and their respective advantages and disadvantages followed by a description of the Aura Interactor actuator used in this study.

Rotary Inertial "Pager Motors"

A motor with a small off-balance weight attached to its rotor produces vibrations as the motor spins. These "pager motors" are commonly found in consumer electronics such as mobile phones and game controllers and are among the cheapest and simplest actuators available.

Although they are easy to control – a higher voltage makes the motor spin faster and increases the frequency of the vibrations – the level of control over the output is rather coarse. The physical properties of these actuators restrict the output to sine-shaped vibrations, with no separate control of their frequency and amplitude. Additionally, they take some time to spin up and down depending on the size of the weight, which makes them relatively slow to respond.

Piezo-Electric Actuators

Piezo-electric actuators use the properties of piezo-electric materials, that deform when a current runs through them, to displace or vibrate a surface. They allow for detailed control over the tactile feedback produced. For example, the TouchEngine by Poupyrev et al. [61], very accurately reproduces both smooth waves and sharp impulses. In a study of Koskinen and Kaaresoja [37], the authors found that the output of piezo-electric actuators were perceived as slightly more pleasant than "pager motor"-type actuators.

Piezo-electric actuators can be made very small, but they are capable of only very small displacements (in the case of the TouchEngine less than 0.1 mm). Furthermore, they require a high operating voltage, which requires more special hardware.

Linear Actuators

Although generally used in direct contact with the skin, linear actuators can also be used in conjunction with a touch screen. This type of actuator consists of an element that moves or oscillates linearly in response to electrical signals applied to its enclosure [4].

The most common types of linear actuator is the voice coil. It consists of a coil and an element with a permanent magnet, similar to the one on the inside of a speaker driver. As a result, voice coils are very easy to control. All it takes is an audio signal to drive the coil. The element can be made to resonate at any frequency (within the physical limitations of the voice coil) and at various amplitudes. Compared to rotary intertial actuators, voice coils have very accurate transient response [6]. Additionally, there is reasonable control over the shape of the waveform, albeit less than with piezo-electric actuators.

Voice coils are cheap to produce and they require no complicated control hardware (a simple audio amplifier is sufficient). Their biggest drawback is the amount of noise produced as a byproduct. As with conventional speakers, bigger displacements (higher amplitudes and lower frequencies) require bigger actuators, but they also produce more noise.

Electrical Stimulation

Another type of tactile actuator uses electrotactile excitation, also known as electrocutaneous stimulation, to stimulate the nerve endings in the skin [50]. One way to use this for touch interfaces is by applying a conductive film to the touch screen. Examples of this are Senseg's E-Sense tixel laminate [70] and TeslaTouch [4].

Using electrical stimulation has several advantages. First of all, unlike mechanical vibrations, electrical signals travel much more uniformly through large touch surfaces; regardless of their shape [4]. Also, because the stimulation requires no moving parts, it can be made very small and moreover it does not produce any sound that could possibly distract users. Finally, electrical stimulation allows fine-grained control over the feedback signal: amplitude, frequency, duty cycle, and polarity can all be controlled individually [50] resulting in feedback that can be more subtle than possible with existing mechanical solutions [4].

Unfortunately, the use of electrotactile feedback for touch screens is still rather new. At the time of this study, Senseg's E-Sense technology was not yet available. The method used by TeslaTouch involves inverting the function of the capacitive layer of a capacitive touch screen and using it for output rather than input, which could not easily be applied to the touch screen used in this study.

The Aura Interactor Actuator

After carefully considering the available options, a voice coil linear actuator was chosen for use in this study. As described before, voice coils can quickly respond and output the different frequencies required by the virtual textures described in Section 5.2.2. Voice coils are also extremely easy to control. They do not require complicated high-voltage circuitry like piezo elements, and steering them requires a simple audio signal. This means that they can readily be driven by anything that can has an output, whether it is a PC or iPad.

A voice coil is a rather crude tactile actuator, especially compared those that use electrical stimulation, but it is also very easy to apply. It can readily be attached to anything that is solid and stiff enough to transport vibrations; no other modifications are required and it does not interfere with the touch hardware. This means it can be used to add tactile feedback to existing hardware, rather than requiring a custom-built vision-based touch screen set-up as used in TeslaTouch [4]. One important limitation of voice coils, as well as of most of the other actuators mentioned in this section, is that they cannot target their output. The actuators cause the whole surface of the touch panel to vibrate, so any finger, hand or arm touching the screen will feel its output. As a result, the same feedback is felt by all fingers when used in a multi-touch scenario. This was not an issue in this study, but places some limitations on their suitability.

The specific actuator used in this study was taken from an Aura Interactor force-feedback vest. It is a rather big voice-coil-based linear actuator that – inside the vest – is mostly meant to reproduce the heavy, low-frequency vibrations associated with the crashes, explosions and gunfire that are common in movies and video games. In tests, the actuator was able to vibrate at frequencies as low as 5 Hertz. The Aura Interactor kit can be purchased for less than $50 and consists of one actuator (shown in Figure 5.6), mounted inside a wearable plastic enclosure, as well as a simple amplifier that has a built-in frequency filter. The frequency filter has three modes: A, B and “Music”. The “Music” setting passes the input signal through unfiltered, which was found to be the most suitable for this study.

5.3.2 Multi-touch Platform

Before finding a solution that was good enough for the purpose of this study, several prototypes were produced to test the suitability of different platforms. The main factor in choosing a platform was responsiveness. As described in Section 5.2.1, the tactile senses are very fast, having a latency of around 5 milliseconds. This means that the response time, e.g. the time it takes the system to go from detecting touch events and processing them, to generating the response and sending the output to the actuator should ideally be less than 5 ms. As the voice coil linear actuator used was controlled using an audio signal, this meant the prototype should have a low latency for both touch and audio.

A first proof-of-concept prototype was implemented in Adobe Flash 10.1 on a PC running Windows 7 with a 22-inch 3M M2256PW multi-touch display. Although it is easy to track individual touches in Flash (the GestureWorks library provides low-level access) and generate tones in response, it does not provide any guarantees about latency. Empirical testing showed the latency from touch to sound was over 500ms. As a result, there seemed to be no relation between the touch input and the tactile response. In particular, this made the illusion of texture fall apart completely, as it relies heavily on the response to the finger sliding across the surface.

Another option that was tried, was a Motorola Xoom tablet running Android 3.0 “Honeycomb”. According to the official Android Compatibility Definition [21] this device should have a continuous output latency of at most 45ms. A quick test however, showed that the touch-to-output latency was near 100 ms and again very noticable. Although Android 3.0 also provides

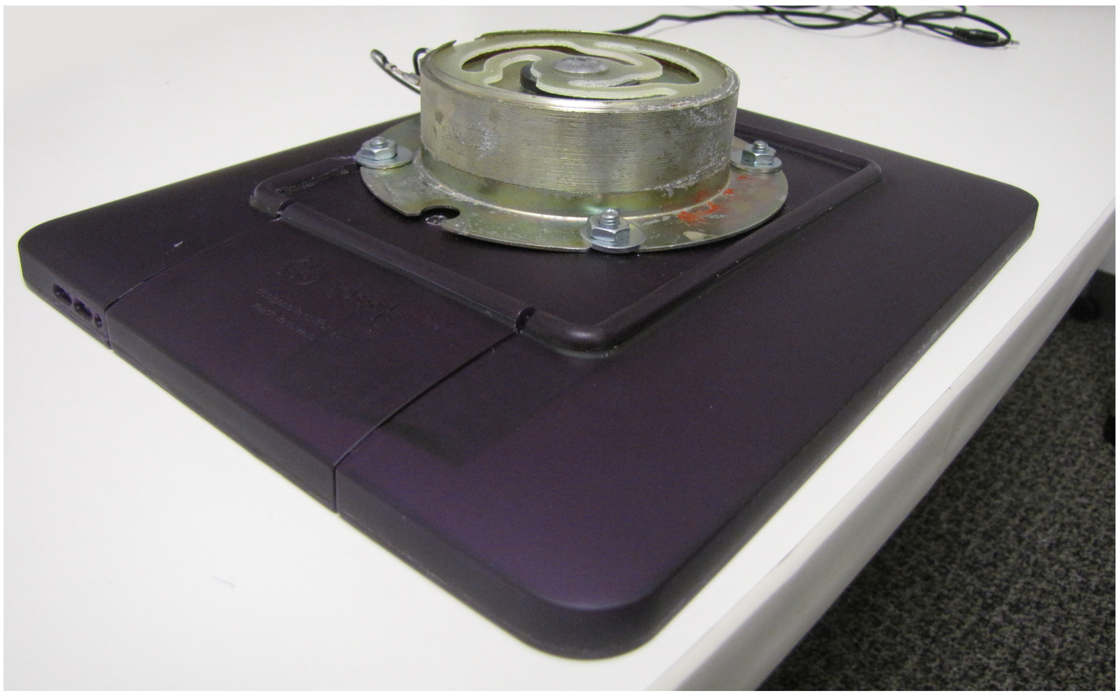


Figure 5.6: The Aura Interactor voice coil linear actuator mounted to an iPad

an alternative OpenSL ES interface for audio, potential latency problems with Android's Dalvik garbage collector led to abandoning the Xoom as a platform.

The platform that was finally chosen was the Apple iPad. The iOS SDK allows for low level "Audio Units" to be programmed in C, with small buffers and thus small delays in audio processing. Additionally, it is very fast at tracking touches, resulting in total delays under 20 ms. While this is still above the theoretical limit for human perception, the latency was no longer noticable in empirical tests.

The iPad used was a first generation model. It has a 10 inch display with a resolution of 1024x768 and a capacitive touch screen. The top layer of the display and the surrounding bezel is made of glass and is smooth to the touch. iPad and actuator were integrated by screwing the actuator tightly onto an external iPad enclosure and putting the iPad firmly inside (Figure 5.6). This allowed for good transferral of vibrations with minimal resonance or additional noise. The actuator was padded with foam and put on a heavy rubber mat to reduce the amount of vibration transferred into the table and to ensure the iPad would stay in place.

5.3.3 System Architecture

As described earlier, the main focus of this study is the specific class of touch screen interfaces that is used as a control for a bigger system. To simplify the technical aspect of the prototype and eliminate any latencies that might be the result of multiple components communicating together, the whole system was implemented on the iPad. Using an A/V adapter, it is possible to connect a computer monitor to an iPad and use it as a secondary screen. This way, the iPad could be used to not just function as the control panel of our mocked up system, but was actually running the whole system itself.

The iPad used in this study was running iOS 4.3. The prototype software was written in Objective-C using the iOS SDK. In order to implement the tactile texture and edge feedback, the touch screen interface was composed of custom *TactileViews* that reimplement the way touch events are handled. The components that come with the iOS SDK are designed to make it easy to adhere to Apple's Human Interface Guidelines. This means that the SDK takes care of processing touch events to figure out which control was touched first and subsquently make sure that the events only get sent to that control until the touch is released. This behaviour was replaced with a custom implementation that replaces and mirrors this functionality, but adds new events that get sent to the controls when the user drags their finger over them.

All of the controls, widgets and views in the user interface were instances of these custom *TactileViews* which, in addition to a graphical representation, could also be given a tactile representation. The response and edge representation was specified programmatically in code, but the tactile representation was loaded dynamically from "texture maps". These texture maps were normal RGBA PNG bitmaps images, the same kind used for the graphical representation, but rather than being displayed on the screen, their pixel data was interpreted as texture information for each location. In the current implementation, the red channel was interpreted as *roughness* and the alpha channel as *strength*, but it is easy to extend this to anisotropic textures by interpreting the R, G, B tuples as vectors instead of using a single scalar.

The texture maps made it very easy to experiment with different textures; changing the feedback for a component was as simple as swapping out one image for another. If the same approach is used for the edge and first-response layers, it potentially allows non-programmers to design tactile feedback the same way they currently design the graphical aspect of the user interface.

5.4 Evaluation

To better understand the effects of the tactile feedback designed in the previous chapter, a user study was conducted. The goal of this study was to find the answers to the following research question:

> "What are the effects of tactile feedback on inattentive operation of indirect touch screen controls".

As mentioned in the introduction, we are interested in three kinds of effects of feedback, namely performance, attention and user experience. There are also two phases of touch interaction, as defined in the model in Chapter 2, that might benefit from tactile feedback and the effects might vary across control types. This leads to the following sub-questions:

1. What are the effects of tactile feedback on task performance.
2. What are the effects of tactile feedback on attention.
3. What are the effects of tactile feedback on user experience.
4. How does tactile feedback affect the different phases of control interaction.
5. How does tactile feedback affect different types of controls.

The main hypothesis was that tactile feedback will have a positive effect on inattentive operation of indirect touch screen controls. This was translated into the following hypotheses:

I. Tactile feedback will improve task performance.

II. Tactile feedback will decrease distraction from the task.

III. Tactile feedback will improve the user experience.

Furthermore, it is expected that tactile feedback will have different effects on the various types of controls and be different during the phases of control interaction involved. In other words, the first three questions deal with what the effects are, whereas the remaining two are about explaining why and where these effects occur. The hypotheses are rather broad. What follows is a more detailed explanation of what is meant by each hypothesis.

Hypothesis I: Tactile feedback will improve task performance

In several previous studies, simple tactile feedback had a positive effect on the speed and error-rate of performed tasks [39, 41]. It is expected that the richer tactile feedback used in this study will also allow users to complete tasks faster, spending less time finding and operating the controls while making fewer errors. When errors are made, they will be of lesser magnitude and thus easier to correct.

Hypothesis II: Tactile feedback will decrease distraction from the task

Tactile feedback will give users an extra sensory channel to rely on. This frees up their eyes from looking at the controls all the time. As a result, users will spend less time distracted from the task they are trying to accomplish using the controls. They will look at the controls fewer times and spend less of their times with their eyes off the main task. This also means they will feel less distracted and overloaded.

Hypothesis III: Tactile feedback will improve the user experience

Partly as a result from improved performance and decreased distraction, users will have a better subjective experience of a system that provides tactile feedback. A system with tactile feedback will be perceived as easier to use and make the user feel more in control. The feedback helps the user find controls more easily by adding an additional sensory channel. It will also help distinguish between successful and failed activation of controls, as well as help recognise when values change by providing tactile confirmation in addition to the visual confirmation that already exists, making it easier to operate the controls and track changes to parameter values.

Additionally, tactile feedback will make not just the system, but also the task appear easier. This will lead to a lower perceived workload and a higher perceived performance.

5.4.1 Method

To investigate the answers to the research questions and test the hypotheses, the prototype was evaluated in a laboratory experiment. The experiment used a within-subject repeated measures design, with two conditions:

1. Touch interface without tactile feedback
2. Touch interface with tactile feedback

Participants were asked to complete a series of tasks under both the tactile and no feedback condition. The tasks, which are described in detail in Section 5.4.2, were specifically designed to measure the variables of interest to the experiment. In order to prevent order bias, the order of conditions was randomised across participants. To reduce practice or learning effects, each condition was tested twice.

To measure the task performance and test Hypothesis I, the completion time and errors were logged for each task.

For measuring visual attention and testing hyphothesis II, the number of times the user looked away from the task display as well as the duration of glances were recorded using an eye tracker. Participants were also asked to indicate how distracting the controls was and to rate their performance and perceived mental, physical and temporal demands on the six factors of the Nasa Task Load Index (TLX) [28, 27].

The presence of tactile feedback was also expected to improve the overall user experience. "User Experience" is a concept that has many facets. Considering that in this study, we were looking at an interface concept (tactile feedback), rather than a concrete system, not all of these facets could be considered.

With that in mind, a minimal questionnaire (Appendix A) was devised based on one concept (Perceived Ease of Use) from the Extended Technology Acceptance Model [72] and two constructs (Performance Expectancy and Effort Expectancy) from the Unified Theory of Acceptance and Use of Technology (UTAUT) [73]. Both of these models include many more concepts; most of which deal with the acceptance of technology within a business or organisation with an existing process and workflow. As mentioned earlier, this experiment was designed to evaluate the concept of tactile feedback standalone, decoupled from potential application areas. As such, questions about the social norms in the participants' organisation, job relevance and other factors that predict technology acceptance in an organisation were not applicable.

To ensure that the participants answered the questions with the controls in mind, rather than thinking about how easy or difficult the task was to complete, the questions in questionnaire were modified to reference the controls, rather than the system as a whole. The *Perceived Ease of Use* was assessed using two questions adapted from the Extended TAM: "Overall, I believe that the controls are easy to use." and "I would find it easy to get a system with these controls to do what I want it to do.". The original Extended TAM includes a third question: "Interacting with the system does not require a lot of my mental effort.". Due to overlap with a similar item on the NASA TLX, this question was dropped. For Performance Expectancy and Effort Expactancy, there was also overlap with the items on the TLX. To assess Performance Expectancy (not considering job performance) the following question was additionally included in the questionnaire: "The controls enable me to accomplish tasks quickly".

5.4.2 Task

In order to obtain performance and eye tracking data and additionally providing the user with a use case to relate to in describing their experiences, a simple task was provided. Rather than choosing a task from any of the real-world use cases that exist for indirect touch control interfaces, a more abstract and artificial task was chosen that offers more control over what can be measured. Specifically, there were several properties required to allow for all measurements, e.g. the task:

- Requires constant visual monitoring of the display, so there are no "free" moments for the user to look at their hands
- Has several modifiable parameters that can be mapped to the different control types, e.g. buttons, sliders etc.
- Can be done with just one hand on the touch screen (the single-channel nature of the vibrotactile feedback limits it to single-touch operation)

Table 5.1: Metrics

Aspect	Metric	Source
Performance		
operation speed	task completion times	action log
amount of errors made	amount of errors made	action log
magnitude of errors	target value - actual value	action log
Attention		
freq. of glances at controls	times looked down	eye tracker
distraction from task	% of time looked away	eye tracker
User Experience		
perceived ease of use	ease of use rating	questionnaire
perceived control	control rating	questionnaire
perceived performance	performance rating	questionnaire
perceived distraction	distraction rating	questionnaire
perceived workload	task load ratings	questionnaire
perceived effort per control	individual controls ratings	questionnaire
preference	preference ratings	questionnaire

- Can be made to require many or few context switches between different controls, so both control aquisition and the changing of values can be evaluated.

There are several existing tasks that satisfy some of these properties to be found in literature. Two examples of such tests are the Lane Change Test (ISO26022) and docking tasks. What follows is a short analysis of both options and their pros and cons, followed by a description of a new type of test that is more approprtiate for this study: the Block Matching Test.

Lane Change Test

The *Lane Change Test* is a popular test for simulating the attention constraints in the context of driving a car [59, 15, 22]. It measures the impact of a secondary task (such as operating an in-vehicle information system) on the performance of a primary task that involves driving a simulated car and changing lanes when instructed. Performance metrics include lane deviation, steering wheel angle, gaze and response time. Driving a car clearly requires constant visual monitoring of the road; even short glances away lead to variance in steering wheel angle and a drift in the position on the road [38]. It is also very suitable for testing interfaces with one hand, since it seems natural to keep the other hand on the wheel.

What makes the lane change test less suitable for the purposes of this study, is the divide between primary and secondary task. The test assumes that the secondary task is unrelated to driving itself and has only a potentially negative effect on driving parformance. In other words, the test only measures the performance of driving under distraction from a secondary task, but not the performance of the primary and secondary task combined. The steering wheel, pedals and levers are what control the car and its position on the road, not the (touch screen) interface of the in-vehicle information system that operates the secondary task. Therefore, it is difficult to use this test to measure the performance of operating touch screen controls.

Docking Test

A more integrated type of test that is popular for assessing touch screen control performance, is the *Docking Test*. Many variations of this test have been used, but they generally involve translating, rotating and/or scaling shapes in 2D or 3D so they fit ('dock') targets. The test was initially mostly used for evaluating multiple-degrees-of-freedom input devices [80], and is still used to evaluate devices like multi-touch mice [5]. Lately, it has also been used to compare table top touch interaction to a mouse [19], investigate different ways to separate the dimensions being manipulated simultaneously [48, 53, 47], and research the differences between one-handed and two-handed multi-touch interaction [5].

In addition to quantitative data about the speed and accuracy (e.g. error rate), it also provides insight into how users control multiple dimensions simultaneously and how efficient the trajectories of movements and other adjustments are [80]. In short, it provides very detailed insight into multiple-degrees-of-freedom movement. Unfortunately, this also places limitations on the interfaces this test can successfully evaluate. Although the test includes multiple parameters that need to be successfully manipulated, all of these parameters are of a similar nature: Angle, translation and scale are all scalar variables with a very specific meaning; there are only limited ways to intuitively map these variables to controls. When testing all kinds of different controls, such as buttons, sliders etc. this becomes a problem. Additionally, the targets in the docking task are static. The user can observe the target and then adjust each parameter until done and then move on to the next one. There is no penalty for looking down at the controls, because the user can trust the target to remain in place. A variation to this test exists that is called the "chase test". where the target moves and the user has to constantly keep adjusting to keep up [80], but this does not provide most of the metrics that make the docking task attractive.

Block Matching Test

Rather than retrofitting the lane change test for measuring touch screen operation performance or extending the docking test to require more constant visual attention and support more kinds of parameters, a new test called the "block matching test" was designed. This new test, shown in Figure 5.7 and inspired by the lane change and docking tests, satisifes all the requirements for this study. It aims to combine the apparent constant need of attention found in the lane change test with the highly measurable parameter adjustments from the docking tests. Additionally, the parameters are of a variety of types, have fewer interactions between them and are easy to test in isolation or together.

The test consists of blocks falling down from the top of the screen – much like in the game Tetris – and a 'cursor' that is fixed to a horizontal rail at the bottom of the screen – like in games such as Break Out and Klax. The objective is to match the *shape*, *colour* and *position* of the cursor to those of the falling shapes. The shape is a two-dimensional variable consisting of an X and Y component; each with three possible positions, resulting in a total of 9 possible shapes. The colour is an ordinal variable that consists of five stops on the spectrum from green to red. Finally, the position of the cursor can be moved freely along the X axis. The target blocks fall down on a field of 1024 pixels wide and 768 pixels tall. After subtracting the size of the blocks (96 by 96 pixels each) and some marigins, this leaves 864 pixels for the cursor to move and thus 864 possible values for *position*.

These parameters were chosen to be as little demanding to match as possible. No mental rotation, complex arithmetic or other time consuming processes were involved. As a result, the test measures the time to operate the controls, as opposed to the time the user spends thinking about the task and judging whether it has been completed.

A final "input" for the test is the acceptance of the current parameters. Once the user has

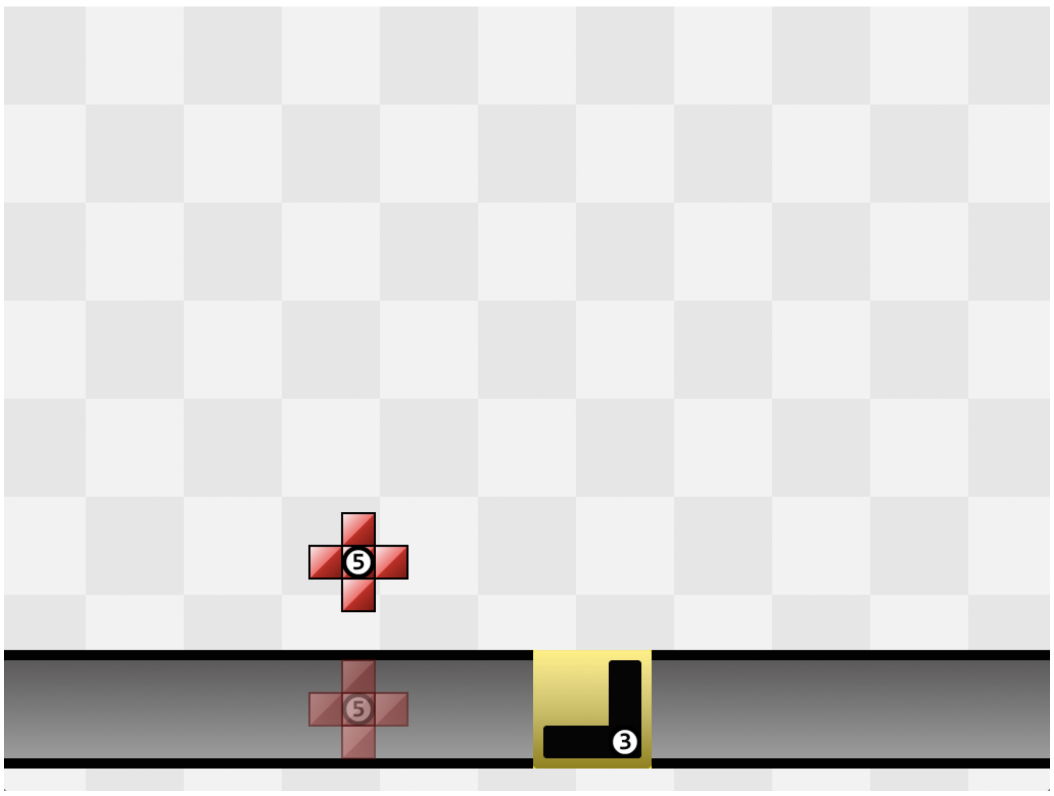


Figure 5.7: The block matching test. Users match the colour, shape and position of the cursor at the bottom of the screen to those of the target block falling down.

finished making all adjustments to their satisfaction, they can signal that they are done and ready to move on to the next target. The "accept" command thus indicates both the start and end of each target in the test.

The target blocks start at the top of the screen and fall down until the participant provides the accept command. If the participant fails to do so, the blocks fall all the way down until they leave the screen at the bottom, at which point the test moves on to the next target. The blocks fall at a rate of 60 pixels per seconds and the total distance is 768 pixels, which means the maximum time a participant can spend on any one target is 12.8 seconds. This speed was found to provide sufficient time pressure to elicit a sense of urgency that discourages the user to look away from the screen, but also slow enough to allow the users enough time to complete the tasks before the blocks disappear.

The test provides performance metrics in the form of error rates for each individual parameter (target value versus actual value), as well as the time it took to make the adjustments. The time is recorded from the moment the target block appears at the top of the screen to the moment the user gives the "accept" command. The parameters of the cursor are also compared to the target at that time. By logging the adjustments made, their paths can be analyzed similar to what was done by Zhai et. al in their docking test [80].

5.4.3 Prototype Interface

For the purpose of the evaluation, a prototype system was designed. It consisted of a task display and a control panel. The task display shows the output of the block matching task as described in Section 5.4.2, which is controlled by set of touch controls on a touch panel. The touch control interface runs on the tactile prototype platform described in Section 5.3. Its user interface, shown in Figure 5.8, contains several touch widgets and gestures that control the cursor on the screen. Each control widget is linked to a different modifiable parameter of the cursor; e.g. shape, colour, position and acceptance. Furthermore, the interface was designed such that the different classes of touch controls mentioned in Section 3 are represented. Due to implementation problems, semaphoric gestures were not included in the prototype interface[1].

Shape Control

The shape of the cursor is controlled by the large area in the centre. As mentioned in Section 5.4.2, there are nine possible shapes – the result of three possible positions for the horizontal part and three for the vertical part. Pressing or moving anywhere in the twodimensional area, changes the shape of the cursor accordingly. The mapping from the cursor control area to the cursor position is one-on-one: pressing in the bottom left corner results in an L-shaped cursor, where the horizontal segment is on the left and the vertical segment at the bottom; pressing in the centre results in a plus-shaped cursor, with both segments in the middle. In other words, the user interacts directly with an on-screen widget, which maps absolutely to the on-screen result. The control can thus be used as a grid of 9 discrete controls, but also as a twodimensional gesture area, where the user's finger position maps absolutely to the two dimensions of the shape.

The tactile feedback for the shape control consists of a texture and edge, as well as response. The texture of the control itself smooth (no vibrations, so it just feels like a glass surface), and it is surrounded by a finger-thick border that has a rough texture. Combined with the edge layer, which defines a sharp border around the area, this helps users recognize when they are leaving the control similar to how bumpy road marks alert drivers they are drifting out of their lane.

[1] Semaphoric gestures were used by the moderator to call up a separate interface to control the experiment, but this was not part of the experiment itself.

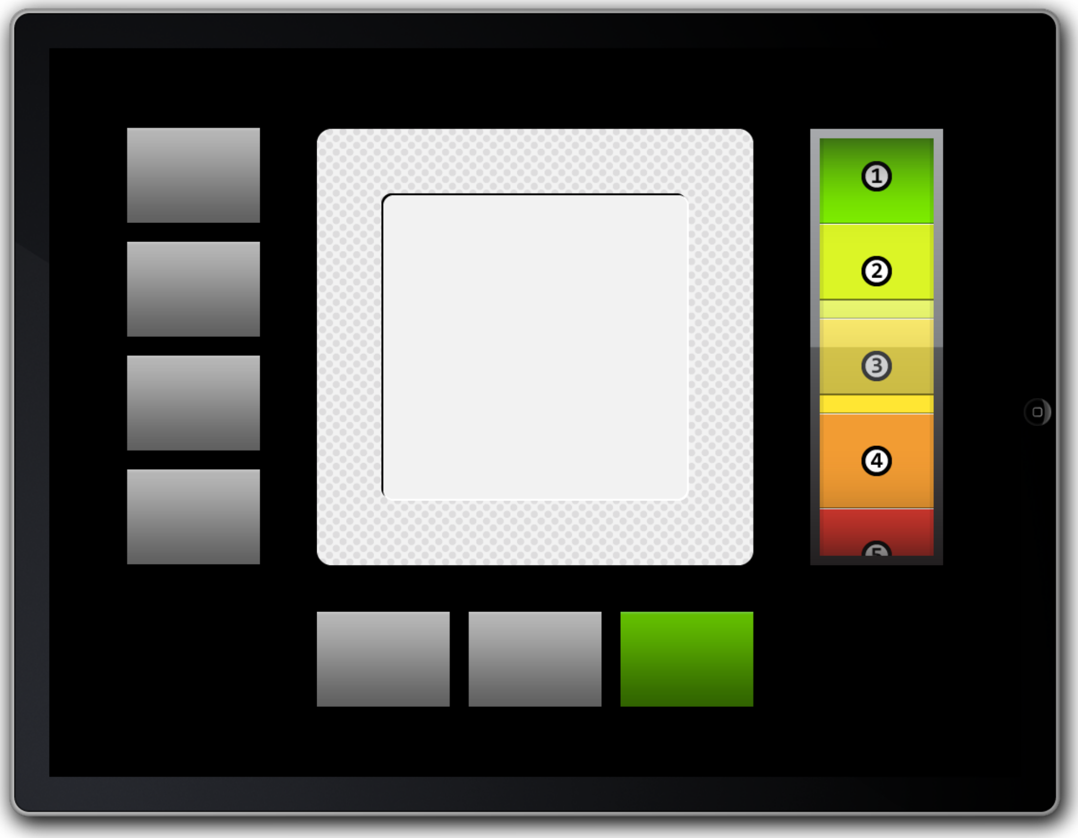


Figure 5.8: The GUI of the touch control panel.

The control responds to user interaction by emitting a soft round bump tacton every time the cursor value changes. This means that as the user slides their finger across the control area, they feel a bump every time they cross a shape boundary and cause the shape to change. An initial bump will also be felt when the user touches down within the control area, but only if the touch causes a shape change (e.g. not when they touch in the middle when the cursor is already plus-shaped).

The bump tacton was created from a recording of a mouth "pop"[2] modified to reduce its pitch so that the maximum frequencies fall in the 0-1000 Hz range to be more tactile. Figure 5.9a shows a plot of this waveform. With the actuator used in the prototype, this waveform was found during informal empirical testing to result in a soft, smooth bumpy sensation.

Colour Control

The colour of the cursor is controlled using a picker widget, located on the right side of the touch screen. The user drags the list up or down to change the current selection (highlighted in the centre of the control). Although the list remains "glued" to the user's finger even when it is outside the touch area, the user may have to make the desired adjustments in several steps; lifting their finger and putting it back down to move more. The widget uses direct manipulation interaction – the list moves exactly as much as the user's finger, but the adjustments are of a relative nature. In other words, the new value for colour depends on the current value and the total displacement made while dragging the list. As mentioned in Section 3.3.1, pickers are a type of embodied continuous control.

The colour control has similar tactile feedback to the shape control. It too has a texture, edge and responses. Like the shape control, there is a tactile edge around the control that has a sharp feeling when crossed. However, it does not have the surrounding rough border that is present in the shape control. Instead, the texture of the control is rough when it is not being controlled (e.g. when the user started touching elsewhere and moves their finger over the control) and smooth when the control is being used. The reason for this is that unlike the shape control, the colour slider keeps tracking the user's finger even after it leaves the control surface. As long as the touch was initiated within the slider, the user can use the whole screen to drag the list up or down.

Similar to the shape control, the slider emits a distinctive tacton every time the value changes. The tacton used is that of a sharp mechanical click[3] (Figure 5.9b). This tacton contains predominantly higher frequencies (between 2000 and 200 Hz), resulting in a sharp sensation.

Additionally, the control starts to vibrate when the list is dragged beyond its beginning or end. The intensity and frequency of the feedback increases with the distance the list is dragged beyond the boundary, to evoke a sense of friction. Finally, when the user touches down on the control, a "soft sliding" tacton is triggered, as if the list came loose from the surface to slide freely. This tacton was created by generating a single 10Hz sine wave – low enough to only trigger the pressure mechanoreceptors – manipulating it so it has a slightly stronger attack and then fading it out at the end (Figure 5.9c). During informal testing, the resulting feeling was described as very soft and being almost as if the touch panel bends a bit under one's touch.

Position Control

The cursor's position is changed using a manipulative gesture. Placing two fingers anywhere on the screen initiates the gesture; there is no on-screen widget associated with the position

[2]Recording used under Creative Commons license – freesound.org/people/Ch0cchi/sounds/15348/

[3]Recording used under Creative Commons license – freesound.org/people/KorgMS2000B/sounds/54405/

control. The position control works much like the middle-mouse "Autoscroll" found in Microsoft Internet Explorer. Rather than affecting the position of the cursor directly, the gesture affects the *movement* of the cursor. The position where the gesture starts is defined as the central or "neutral" point. Moving to the left or to the right from that point causes the cursor to start moving in that direction with a speed proportional to the distance between the user's fingers and the neutral starting point (the distance measured in units of 20 pixels). The resulting effect is similar to that of a steering wheel: turning the wheel to either side causes the car to start turning in that direction until the wheel is turned back into the central position. Additionally, the gesture ends when the user releases the touch, in which case movement also stops instantly. The indirect nature of the control makes it more challenging to operate. Accurate positioning of the cursor involves careful planning and moving back to the neutral position in time in order to not overshoot.

As the position is not embodied on the touch screen, it does not have a texture or edges. It does have two kinds of responses. To indicate succesful activation of the gesture using two fingers, a soft "poof" tacton is emitted. Then, every time the user moves one step further or closer to the central point, a sharp "click" is triggered (the same tacton used in the colour control). The aim is to help the user to keep track of how far away from "neutral" they are: five clicks to the left followed by five clicks to the right brings the cursor to a halt.

Accept Control

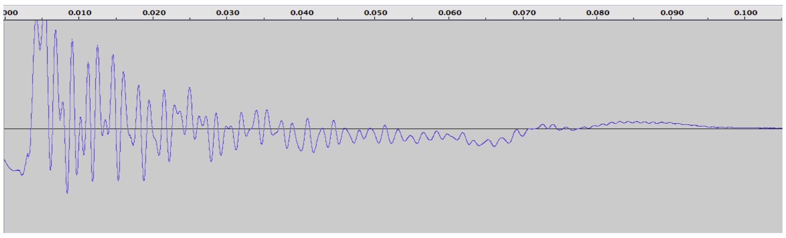
The accept command is mapped to a green button in the lower right corner of the interface. The button's behaviour is identical to typical touch buttons. It is pressed upon touch, but the command is only fired when the finger is released within the boundaries of the button's surface (the finger may leave and re-enter the surface in between). This is the default behaviour of most touch interfaces, such as Apple's iOS and Android, and offers the user a way out of accidentally pressing a button. To make the prototype more realistic, six additional dummy buttons were added to the interface: two at the bottom and four on the left. These buttons react the same way as the accept button, but have no actions attached to them.

The passive tactile feedback for buttons is straightforward: they have a rough texture when rubbed and – like the other controls – a sharp edge surrounding them. The different buttons are spaced slightly apart, with a smooth gap in between them large enough to allow a user to feel the end of one button and the beginning of the next when dragging a finger over them.

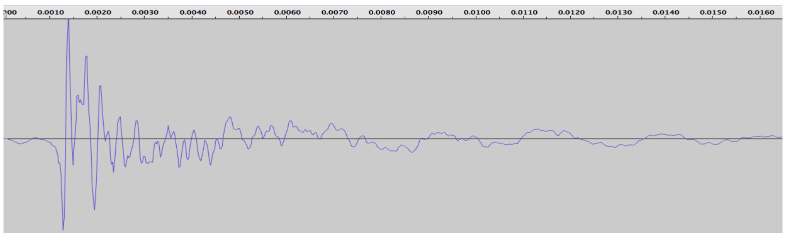
When a button is activated by touching down on its surface, it emits a "key down" tacton. If the user subsequently moves their finger outside of the area, which would result in the button no longer being triggered if the user lifts their finger, a "key up" tacton is emitted. If the user moves their finger back into the active button area after leaving it, the "key down" tacton is played again to indicate the button would fire if the user releases. When rubbing over buttons that are not currently in the process of being pressed (e.g. the touch did not start within their surface area boundaries) these responses are not triggered; only passive feedback is emitted. The "key down" and "key up" tactons were made by recording the pressing and depressing of a keyboard button with a microphone, and subsequently amplifying the low frequencies of that recording (between 0 and 200 Hz).

5.4.4 Test set-up

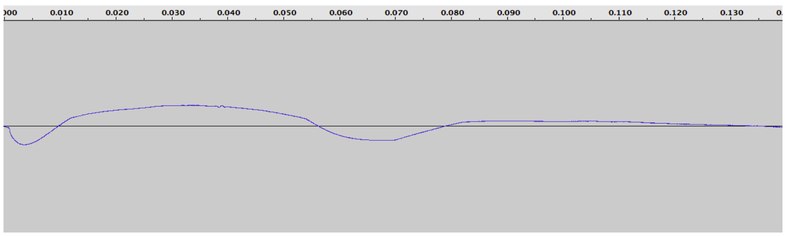
All tests were conducted in a small laboratory space. The room had no windows and everything that might possibly distract users during the experiment was removed. Participants were asked to be seated at a large table, facing an empty wall. The prototype was set-up on top of the table,



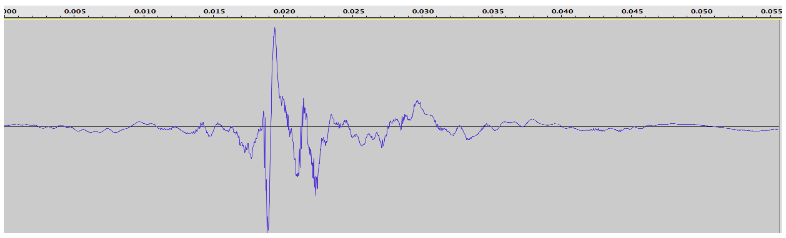
(a) The “pop” tacton used in the shape control.



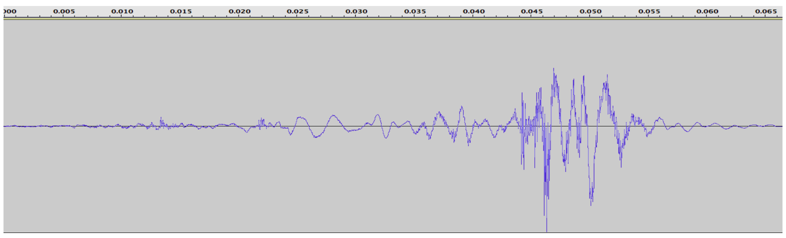
(b) The “click” tacton used in the position and color controls.



(c) The “soft sliding” tacton used in the color control.



(d) The “key down” tacton used for buttons.



(e) The “key up” tacton used for buttons.

Figure 5.9: Waveforms of tactons used in the prototype.

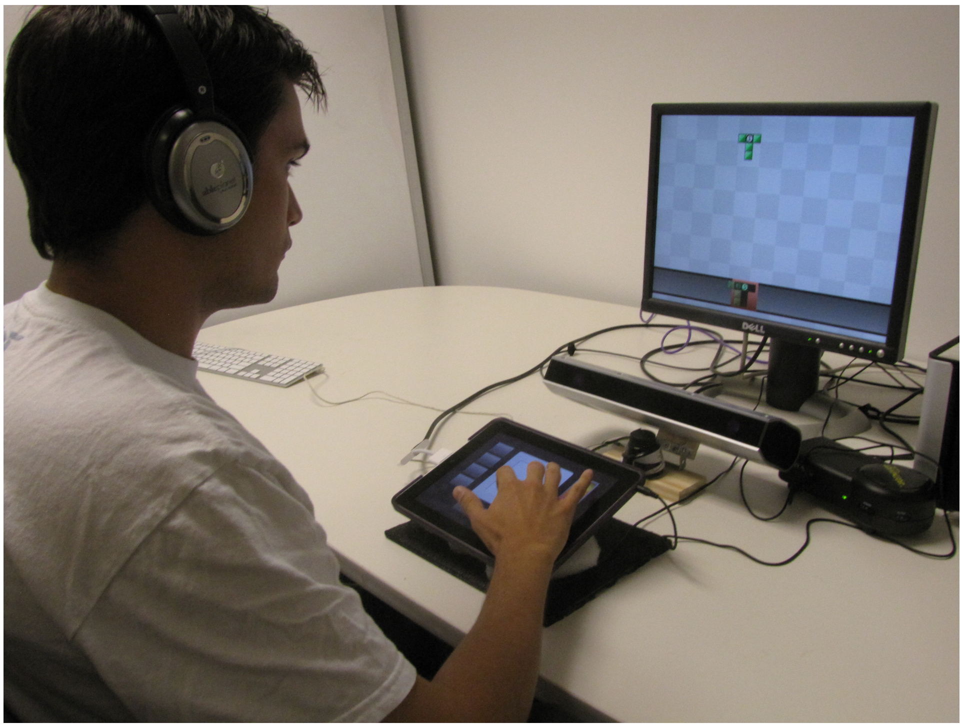


Figure 5.10: The test set-up used during the experiments.

in front of the participant as shown in Figure 5.10. It consisted of a 22 inch TFT display and an iPad running the touch control interface software. The TFT display was placed just out of reach of the user, with the iPad mounted on a stand and placed horizontally (at a slight angle) in front. This was done to convey that the participant was supposed to interact with the touch panel to control the TFT display. The camera element of the eye tracker was placed between the iPad and the TFT display. An iMac was placed out of sight, behind the participant, to record the interaction with its integrated camera.

The experiments were lead and operated by a human facilitator. Throughout the sessions, the facilitator was seated at the end of the table, to the right of the participant. A secondary TFT display allowed the operator to monitor the eye tracker output and record notes. The eye tracker display was turned away from the participant, so it would not distract them or interfere with the test. An overview of the set-up can be seen in Figure 5.11.

A side effect of the vibrotactile actuator used in the prototype, was that it made a lot of noise. To prevent interference and distraction, participants were asked to wear noise cancelling head phones. During pilot testing, instrumental music with constantly playing electrical guitars was found to cancel out the noise the best, so an album of about an hour (longer than the duration of the test) was played in the background.

5.4.5 Pilot Study

A small-scale pilot study was conducted ahead of the full-scale evaluation. The study was conducted among five colleagues from the User Experience group, who were familiar with the purpose of the project and are experienced in designing and evaluating interactive systems. The

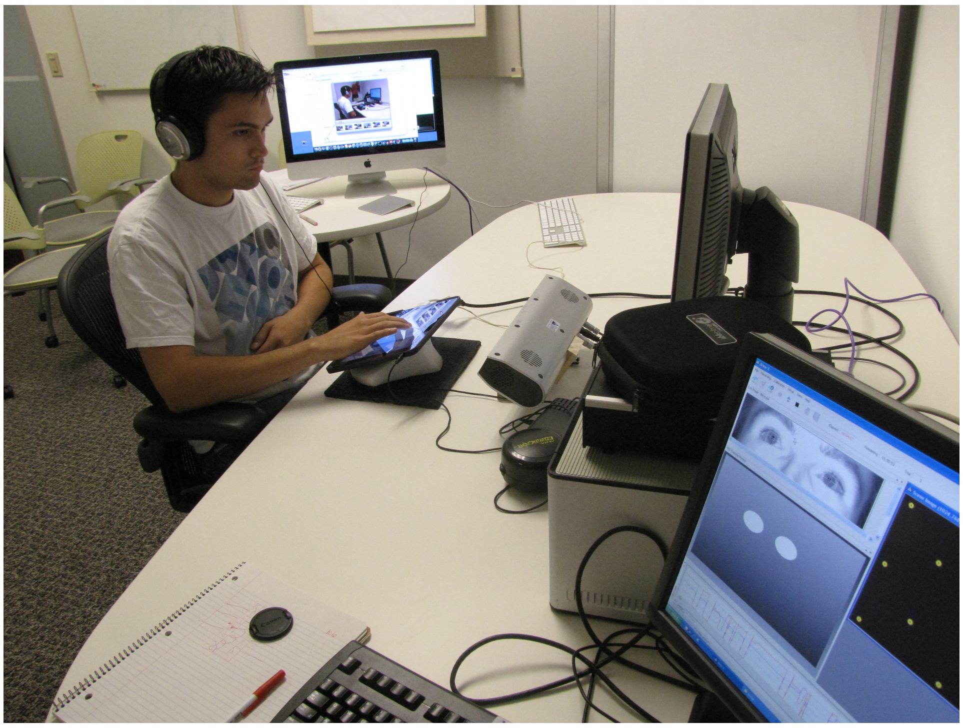


Figure 5.11: Overview of the test set-up, including eye-tracker monitor and recording equipment.

purpose of this pilot study was to test the set-up, check things like the difficulty of the tasks, the duration of the sessions etc. and tweak the design of the prototype.

Additionally, two alternative tactile feedback designs were compared. One design where the buttons were smooth and the background was rough was compared to another design where the buttons were rough and the background smooth. Participants were asked to explore both interfaces on their own while thinking out loud, and perform part of the tasks designed for the final study. 4 out of 5 rated the variant where the buttons have a texture as easier to use and preferred over the alternative.

They remarked that it was easier to identify the buttons by the transition from smooth to rough than vice versa. Also, the fact that there is no feedback for inactive areas of the touch screen matches the fact that there is also no feedback when the user touches the bezel of the iPad (where it is not sensitive to touch). A drawback of having textured buttons that was mentioned by two participants, is that it is harder to feel the difference between response feedback and texture feedback. Other participants, however, stated that they felt it made more sense that the interactive parts (buttons etc.) have all the feedback and the inactive parts have no feedback and that it was too “busy” having feedback for both.

From the results of the pilot test, the version with textured buttons and smooth background was chosen for the final evaluation. Pilot testing and earlier tests with colleague designers and researchers also lead to some other small changes. Most importantly, some participants had trouble matching the colours by similarity. As the intention was to test the controls, not the participants’ ability to compare colours, numbers were assigned to the colours on the touch panel as well as on the display, in order to make them easier to recognise. Although this might mean that some participants perform the task by relying on the colours and others on the numbers, it was expected that participants would adopt the same strategy regardless of the conditions and that this change would therefore not affect the results.

5.4.6 Participants

The study was conducted among 10 participants aged from 19 to 28, with an average age of 24. They were all interns or employees of various departments of Siemens Corporate Research; most of them students with a background in computer science or medicine. All of the participants were male and all but one right handed.

The participants had varied levels of experience playing video games (median 5.5 on a 7-point Likert scale, with 4 participants indicating having "much experience") and average to above average experience using touch devices (median 5 on a 7-point scale). All participants owned or frequently used at least one touch device and 8/10 owned a smart phone.

5.4.7 Procedure

The evaluation procedure consisted of an introduction, followed by several task sessions and finally a short closing discussion with debriefing.

Introduction: Briefing, Exploration and Eye Tracker Calibration

The participants were welcomed into the room by the facilitator and offered some coffee or a soft drink. They were asked to be seated at the table, in front of the prototype set-up. The facilitator explained that the purpose of the study was to learn more about touch screen interaction, specifically the way different kinds of feedback affect performance.

After the briefing, participants were asked to provide some demographic details about themselves, such as their age and experience with touch devices and video games. The form used can be found in Appendix A.

Next, they were given five minutes to freely explore the prototype while thinking aloud. The tactile feedback was turned on and participants were given noise cancelling headphones to wear and were asked to not lift up or move the iPad. If after three minutes, the participant did not find all of the functionality and controls, the facilitator would intervene and explain the undiscovered features, after which the participant would be given two more minutes to try them out. As a result, all participants had the same knowledge about how to operate the controls before starting the tasks.

Because the noise cancelling headphones made it hard for the participant to hear the facilitator, it was not possible to interrupt them while exploring to remind them to think aloud. Therefore, all participants were also given the opportunity to make remarks about their experiences after they were done exploring.

The introduction was concluded with an explanation of the tasks the participant would perform and calibration of the eye tracker. Participants were explained that the purpose of the task was to match the colour, shape and position of cursor to that of the targets and then press "accept". They were asked to do this as good and as fast as possible and were explained that both the time and accuracy of their target matches were recorded.

Task Sessions

After the instructions, users were given some targets to practice. This served not only to give the users some experience using the interface to perform tasks, but also as a last-minute check to confirm that participants remember how to use the controls correctly and test the eye tracking.

The practice trial consisted of a series of tasks that was slightly shorter than those of the actual tests, but enough to get familiar with all the different kinds of adjustments. The series, included in Appendix B, was ran twice: once with and once without tactile feedback. In case any problems arose during practicing, the facilitator would explain or correct the user, so that after the two practice sessions, all participants were sufficiently prepared.

Next, participants were asked to perform two sessions of task trials. Each session consisted of a single trial of tasks that was ran twice: once with and once without tactile feedback. The trials, shown in Apendix B, consisted of series of ten targets followed by a break for the participants to rest their eyes and neck (although the eye tracker followed the user's head, they still had to remain somewhat still, so the breaks were inserted to give them some relief).

The trials had the user first adjust the shape ten times, then the colour and then the position. Then, there were five targets that required both the shape and colour to be changed, followed by five targets that had a different shape and position and five targets with a different colour and position. Finally, there were five targets where all three parameters needed to be changed. For those targets where one or more of the parameters would not need changing, those parameters of the target would automatically be set to whatever value the cursor currently had to ensure that the participant would not get distracted by unnecessary adjustments. The trial used for session 2 was a mirror image of that used in session 1, all adjustments are of a similar order, to ensure that they can be compared.

After each trial run, the participants were asked to fill out a questionnaire. During the first session, forms 2 and 3 of Appendix A were used. One form was filled out after the trial was run with tactile feedback and the other after the trial without tactile feedback (depending on the order used). The forms prompted the user to rate the ease of use and distraction of the interface, as well as how they experienced the workload during the task.

Then the second session was run, again consisting of two trial runs using the same series of tasks, once with and once without tactile feedback. After the first run, the participant would fill out forms 4, 5 and 6 – more detailed versions of the forms used in the first session – and after the second run the identical forms 7, 8 and 9. These forms additionally asked about the perceived speed, control and the workload of the individual controls.

Closing remarks & debriefing

After the last session, the participants were asked some general questions about the effects of tactile feedback and their preferences using a final questionnaire form (form 10). Then, the session was closed with the opportunity for the participant to clarify their answers and provide comments. This had the form of an open-ended interview with no predefined questions.

5.4.8 Results

As outlined in Section 5.4.1, the effects of tactile feedback were studied with respect to task performance – speed and accuracy – distraction, perceived task load and subjective user experience.

Task Performance: Speed

The task duration was measured as the time spent working on the task, minus any breaks taken. On average, it took participants just over 6 minutes to complete a single series of tasks under each condition: 382.5 seconds *without feedback* (SD=57.7) and 379.8 seconds with *tactile feedback* (SD=63.06), This means the difference, just under three seconds, is rather small compared to the variance. as can can be seen in Figure 5.12. A pairwise t-test was conducted and showed no significant difference for total duration (t=0.369, p=0.720).

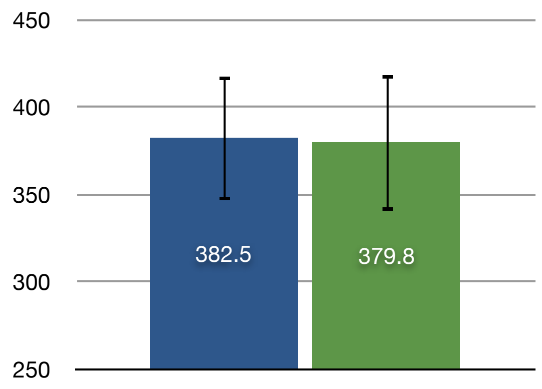


Figure 5.12: Mean duration (in seconds) under the without (blue, left) and with (green, right) tactile feedback conditions. Error bars represent the 95% confidence interval.

To understand the time needed to complete the different kinds of adjustments the mean task completion time was measured for the *shape*, *colour* and *position* tasks, as well as those that required combinations of each to be changed. Figure 5.13 shows how much time participants needed to complete each kind of task in the first and second session.

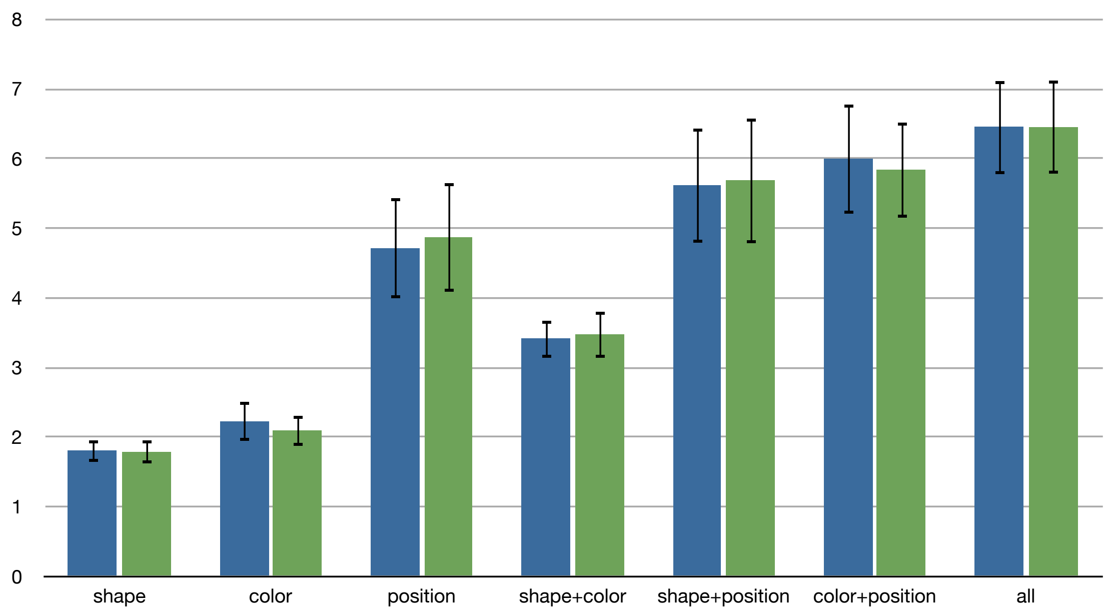
Participants were able to complete tasks that required just the *shape* to be changed fastest, followed by *colour* and the combination of the two. Adjusting the *position* generally took much longer; on average about twice as long as shape or colour. Tasks that required participants to adjust other parameters in addition to the *position* required only minimally more time, especially in the second session.

The differences in task completion times per task between the *no feedback* and *tactile feedback* condition was small for all types of tasks. The biggest difference can be seen in the second session, for tasks that required all parameters to be manipulated. There, participants needed on average 6.7 seconds to complete the task without feedback (SD=0.87) and 6.1 seconds to complete the task with tactile feedback (SD=1.29). A paired samples t-test was performed for each of the task types. It showed that only the difference for *all* in the second session was significant (t=2.297, p<0.05). Only one participant was slower with tactile feedback in that case, all others were faster.

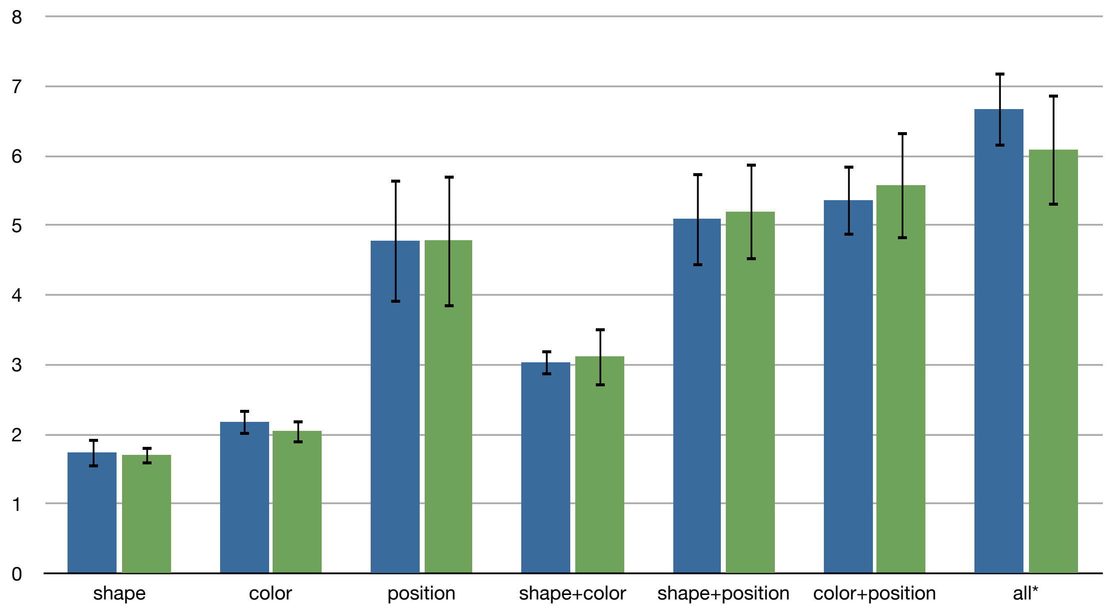
Task Performance: Accuracy

Overall, participants made very few errors. The errors they did make, were generally small. Errors were measured for each of the manipulable parameters: shape, colour and position. On a few occasions, errors were introduced that were not related to the operating of a particular control. For example, participants accidentally pressed the "accept" button twice, therefore skipping over one task entirely without making adjustments. To prevent such outliers from skewing the results, any errors that were more than three times the standard deviation above the mean were taken out and replaced by the mean error value.

There were no significant differences in the number or magnitude of errors made for shape, colour or position. The whole test involved changing the value of *colour* (in isolation or alongside other parameters) 50 times per condition, 25 times in the first session and again 25 times in the second session. Taken over the whole test, the mean total number of errors per participant was 0.8 for the *no feedback* condition and 1.2 for the *tactile feedback* condition with standard deviations of 0.92 and 1.55 respectively. A pairwise t-test did not show this difference to be



(a) Session 1



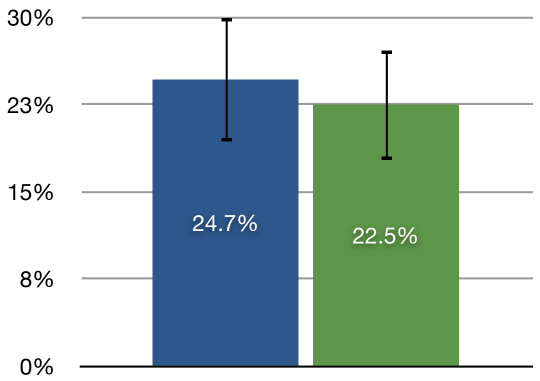
(b) Session 2

Figure 5.13: Mean duration (in seconds) per target of each kind under the without (blue, left) and with (green, right) tactile feedback conditions. Error bars represent the 95% confidence interval.

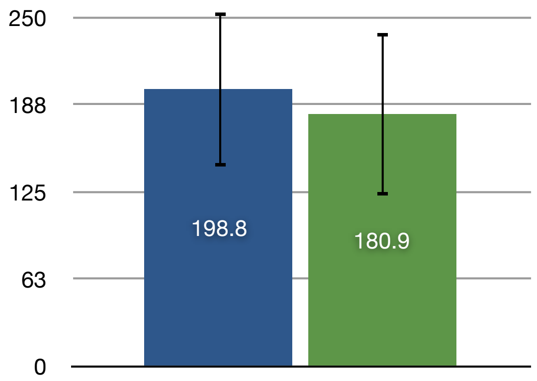
significant (p=0.462). As there were only two errors of magnitude greater than 1, the calculation of the average error magnitude was ommitted for colour.

In adjusting the *shape* parameter, participants made even fewer errors. Taken over the whole test, just 8 shape errors were made by all participants; four in the *no feedback* condition and four with *tactile feedback*. No significant difference was found in the number or magnitude of errors for shape.

Participants made by far the most errors while adjusting the *position* (where even a single pixel off-target was counted as an error), with a mean of 40.5 errors for 50 adjustments made under the *no feedback* condition (SD=6.04) and 42.50 with *tactile feedback* (SD=4.95). The mean magnitude of the errors was low under both conditions: 3.48 pixels for *no feedback* (SD=2.98) and 3.04 for *tactile feedback* (SD=2.47). A pairwise t-test was conducted to determine whether these differences are significant. There was no significant difference for number of errors (p=0.165) and no significant difference for mean magnitude of error (p=0.307).



(a) Mean time distracted (in % of total)



(b) Mean total distractions*

Figure 5.14: Distraction under the without (blue, left) and with (green, right) tactile feedback conditions. Error bars represent the 95% confidence interval.

Distraction

Visual distraction was measured using an eye tracker in two ways: how many times a participant looked away and for how long. Both were measured while participants were performing tasks.

The total amount of time participants were distracted, in other words: looking away from the task display, was measured as a percentage of the total task duration. Figure 5.14a displays the results: participants were looking away from the screen 24.7% of the time under the *no feedback* condition (SD=8.63) whereas they were distracted on average 22.5% of the time with *tactile feedback* (SD=7.75). A pairwise t-test did not show a significant difference for *mean time distracted* (t=1.603, p=0.143).

Taken over the whole test, e.g. two sessions (100 targets), participants glanced away from the task display on average 198.8 times (SD=89.4) under the *without feedback* condition, compared to 180.9 times under the *tactile feedback* condition (SD=93.7). Figure 5.14b contains a graph of this difference. A pairwise t-test showed a significant difference for *number of distractions* (t=2.241, p<0.05).

Another way to look at the number of distraction events, is to consider them per task. Knowing that participants completed 100 tasks per condition, this means that they glanced away from the controls (on average) 1.99 times per task with *no feedback* and 1.81 times per task with *tactile feedback*. The normalised *distractions per task* is easier to compare and allows us to take a closer look.

Figure 5.15 shows the distractions per task for both conditions in the first and second session of tasks. In the first session, participants looked away an average of 2 times per task under the *no feedback* condition (SD=0.81) and 1.9 times per task under the *tactile feedback* condition (SD=0.99). In the second session, the differences were bigger. The mean number of distractions per task was 1.9 under the *no feedback* condition (SD=0.99) compared to 1.7 times under the *tactile feedback* condition (SD=0.94). A pairwise t-test did not show a significant difference for session 1 (t=0.773, p=0.46). The difference for session 2, on the other hand, was significant (t=3.830, p< 0.01).

To further understand where the differences in number of distractions, pairwise t-tests were performed for the mean number of distractions for the *shape*, *colour* and *position* tasks and combinations thereof. The results are shown in Figure 5.16. As we found before, participants

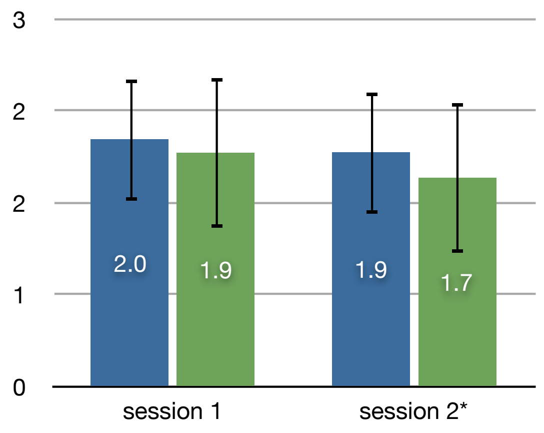


Figure 5.15: Mean number of distractions per task in each session under the without (blue, left) and with (green, right) tactile feedback conditions. Error bars represent the 95% confidence interval.

were on average equally often distracted for each of the parameters in the first session. Only for those tasks that required them to adjust all parameters, a significant difference was found: a mean of 3.34 distractions was measured in the *no feedback* condition (SD=1.19), whereas under the *tactile feedback*, 2.62 distractions per task were seen on average (SD=1.37). This difference of 0.72 distractions per task was found to be significant (t=3.28, p<0.01).

In the second session, the mean number of distractions was lower for nearly all the task types under both conditions. On average, the difference was bigger for the *tactile feedback* condition. The variances were high, so most differences are not significant. The biggest difference was found for *colour*, which counted a mean of 1.59 distractions for *no feedback* (SD=0.77) compared to 1.14 distractions for *tactile feedback* (SD=0.68). This difference was found to be significant (t=3.48, p<0.05). Other differences that were large, but not significant, are *position* (1.76 vs 1.60, p=0.07 and *color+pos* (2.48 vs 2.02, p=0.11).

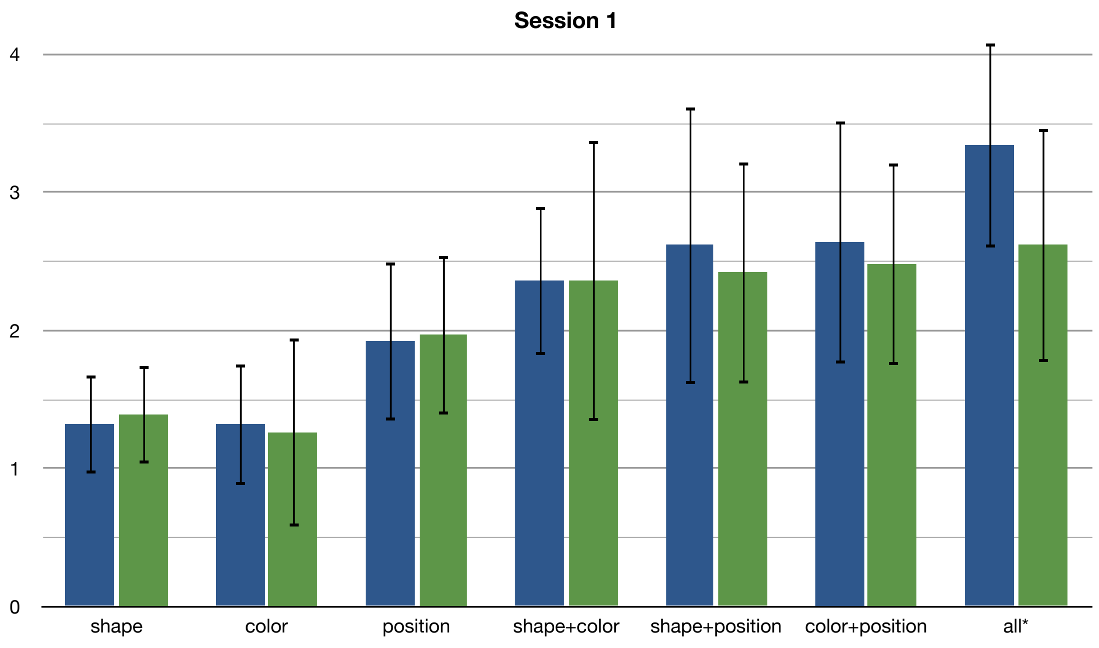
Task Load

Subjective task load was measured by asking people to provide ratings on open 20-point scales taken from the NASA Task Load Index. Figure 5.17 shows the mean ratings per factor under the *no feedback* and *tactile feedback* conditions in the first and second session.

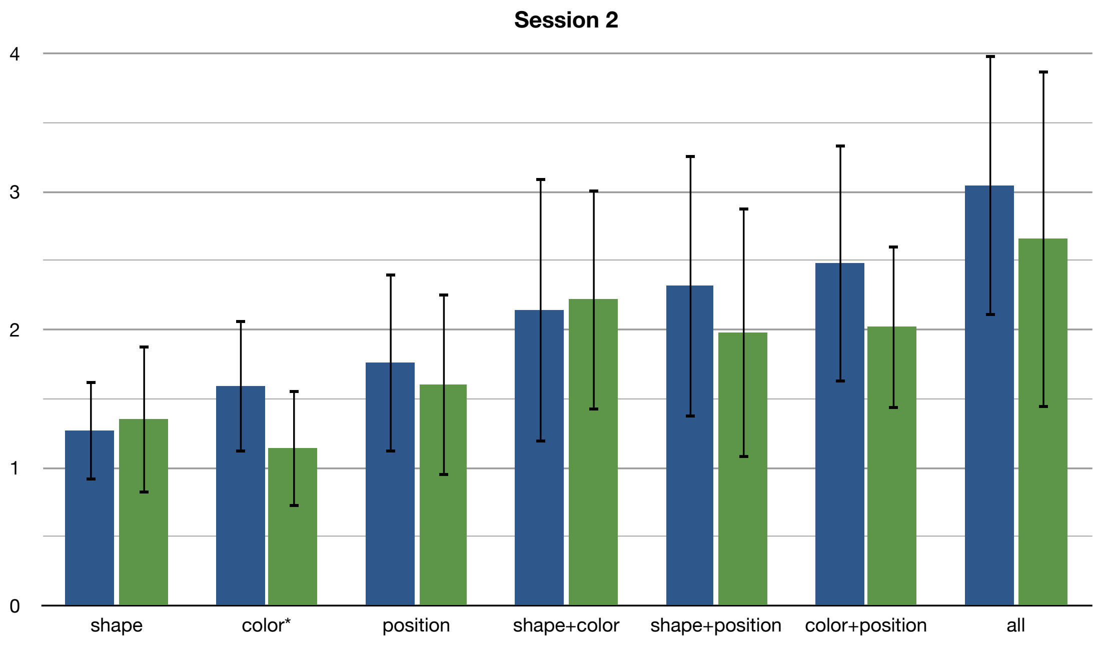
Overall, the perceived task load was not very high (in the lower half of the scale) and the differences in ratings between the conditions were small. In the second session, some bigger differences are visible. Participants indicated a slightly lower task load in the *without feedback* condition than when using *tactile feedback*. A pairwise t-test was performed to determine whether these differences were significant. Only for *performance* in the second session, a significant difference was found (t=2.362, p<0.05).

Participants rated their performance with a mean of 14.6 (SD=3.3) under the *no feedback* condition, compared to 12.8 (SD=3.5) with *tactile feedback*.

Although time considerations made it impractical to obtain task load ratings for the different types of tasks, participants were asked to rate the effort of operating each of the controls after the test on a similar scale. For the controls for shape, colour and accept, ratings were obtained about the effort to *find* and *change* the control. The position control used a screen-wide gesture, so participants were only asked about the effort to change the value for this control.

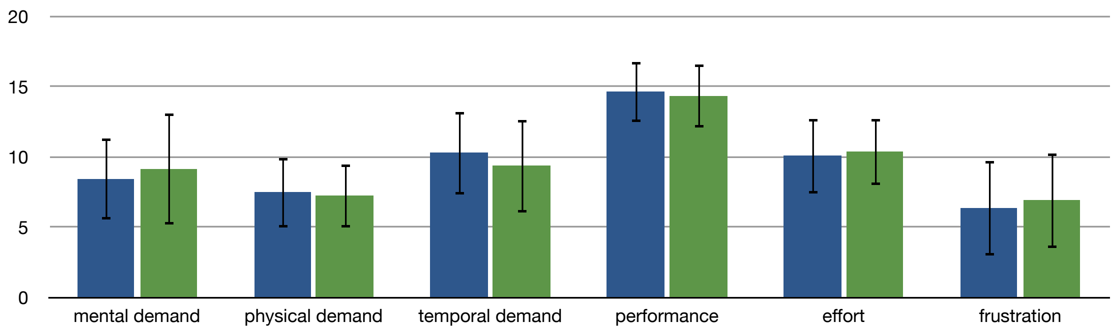


(a) Session 1

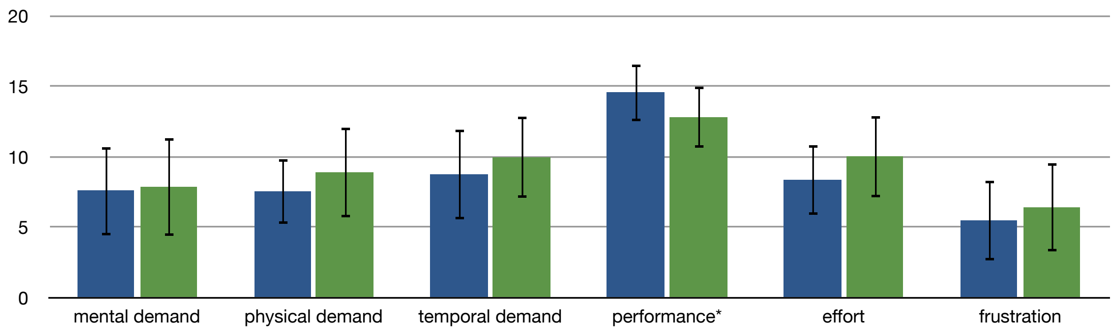


(b) Session 2

Figure 5.16: Mean number of distractions per target of each kind under the without (blue, left) and with (green, right) tactile feedback conditions. Error bars represent the 95% confidence interval.



(a) Session 1



(b) Session 2

Figure 5.17: Ratings for the Task Load Factors in each condition. Ratings range from 0 (low) to 20 (high).

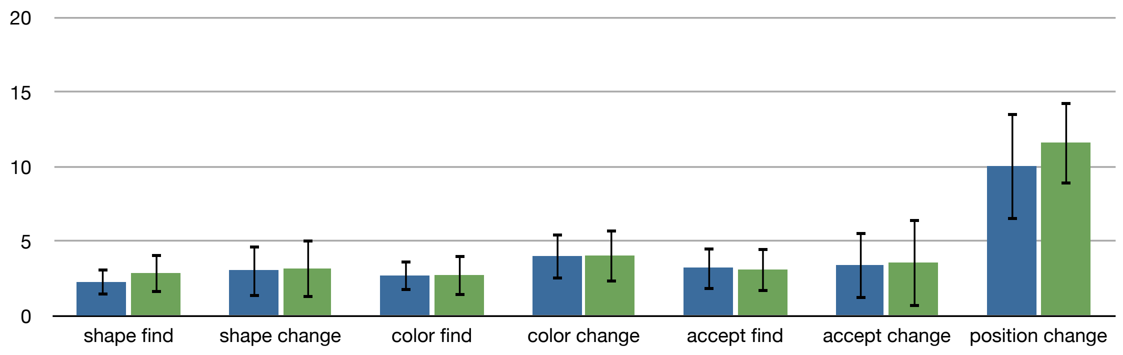


Figure 5.18: Mean perceived effort to operate each control in each condition.
A rating of 0 means "low" and 20 means "high".

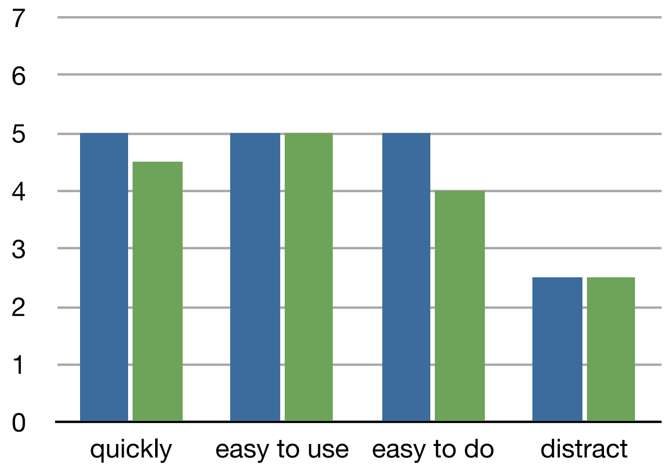


Figure 5.19: Median ratings for the user experience questions after session 2. Answers are on a 7-point likert scale.

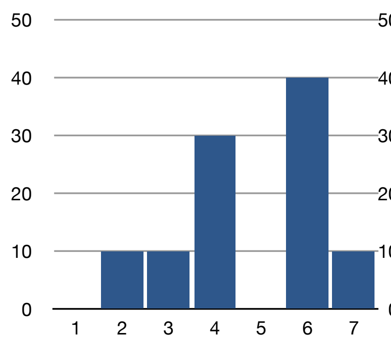
Figure 5.18 shows the mean ratings for the perceived operation load for each control. Overall, the ratings for all of the controls were very low, with the exception of position. The differences between the condition with or without tactile were small and no significant differences were found.

User Experience Ratings

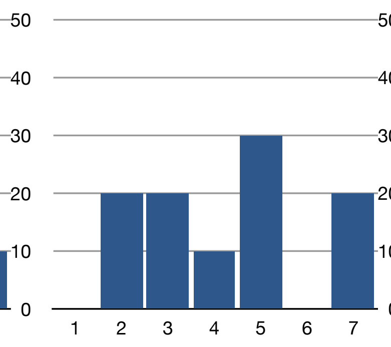
The rest of the subjective user experience data was obtained from the likert items in the questionnaire, as wel as post-hoc interviews and remarks.

After each session, participants were asked to rate their experience with each prototype on a number of factors, such as how quickly the controls helped them accomplish their tasks, how easy they were to use, how much they distracted from the task and how easy it was in general to get a system with these controls to do what they want (see Appendix A).

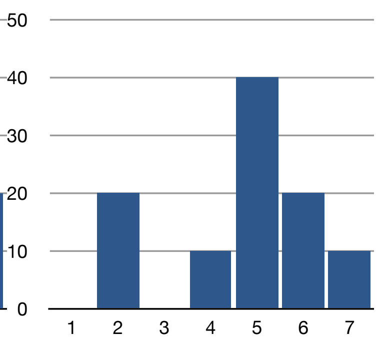
Figure 5.19 shows the median of all 10 responses for each item. Participants mostly rated the controls equally under both conditions. Only in the second session were there any real differences,



(a) improves ease of use



(b) improves focus



(c) improves performance

Figure 5.20: Frequency of responses (in % of participants) for the questions whether tactile feedback improves ease of use, focus and performance. Answers are coded from 1 (strongly disagree) to 7 (strongly agree).

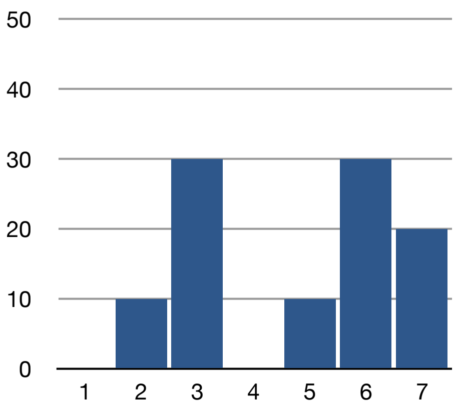


Figure 5.21: Frequency of responses (in % of participants) for the question "I prefer the system with tactile feedback".

but a Wilcoxon Signed Rank test did not indicate that any of them were significant.

After the test, participants were also asked some very specific questions about the effects tactile feedback had on their experiences with respect to *ease of use*, *focus* and *performance*. As can be seen in Figure 5.20, participants were divided on whether tactile feedback made the interface easier to use. 30% of participants neither agreed or disagreed that it improved anything, whereas 20% indicated they disagreed. The remaining 50% agreed strongly that tactile feedback improved ease of use.

The responses to the statement that tactile feedback improved focus were more divided. Half of the participants indicated agreeing with the statement, compared to 40% that disagreed.

On the statement that Tactile Feedback improves performance, participants were the most in agreement. Just 20% of participants disagreed, whereas the majority (70%) indicated that they felt that tactile feedback indeed improved performance.

When asked if they preferred the system with tactile feedback of the one without, participants visibly fell into two camps (as seen in Figure 5.21). 60% of participants preferred the system with tactile feedback (most of them quite strongly), whereas the remaining 40% did not.

User Experience Remarks

At the end of each test, participants were asked to provide remarks about the tasks, the experiment, the prototypes or their experiences in general. The number of remarks was small enough to list them verbatim, organised by topic.

Inattentive Operation Participants made some general remarks about dividing their attention while performing the tasks and the effects of the design of the controls and feedback.

When it came to operating the controls without looking, one participant indicated that the accept button was the cause of the most frustration: “It’s a bit small. If I was able to look at the control panel it wasn’t a big problem, but if I was trying to perform the task without looking at the controls it was a bit more difficult.”.

About the colour control, one participant stated: “This suits the application the most. It allows you to select the colour without looking down.”.

About the effects of tactile feedback on attention and distraction, participants said the following: “The tactile feedback was really helpful. Especially as you are trying to not look at the touch panel. When you are switching up and down (colour) and changing the shape. It was also helpful for finding the button, not to have to look at the screen all the time.” and that it is useful “so you know when it changes and you don’t have to look at the device all the time”.

One participant said he would like to see a miniature of the touch panel layout on the large screen: “That way my eyes don’t need to move so much. I don’t need to be able to touch the main screen, but just so I can see it. It would help me press things on the touch screen better.”

Suitability of Tactile Feedback A lot of the remarks made by participants at the end of the study were about their opinion on the suitability of tactile feedback in general or the way it was implemented in the prototype for the different types of controls.

Three participants stressed that they found tactile feedback most beneficial when operating the accept button. They said it helpted them recognising whether they had successfully pressed the button and detect when they mispressed it.

About the suitability of the tactile feedback for shape selection, participants had different opinions. One participant said: “It helped me recognise when the shape changed. If it didn’t I could not trust that it changed.”. Another participant agreed, but thought it could be improved to be more thrustworthy: “If the feedback was different it could be easier for other people. For me it was fine. I expected richer feedback. For example, the feedback for all shape changes is the same. It does not say which shape you are changing to. If you go diagonally, you can change two times and then if you rely on it too much, you can betray yourself. I trusted the controls a bit less for shape because of this. This problem does not exist for colour or the accept button.”. Yet another participant said the feedback did not add anything for shape: “Whether there is tactile feedback with [shape] does not matter. I can just visually compare.”.

Regarding the colour control, three participants explicitly stated that tactile feedback made it easier to use and two participants said it was not really necessary or even undesirable.

The position control was universally considered the most difficult. Participants called it “really hard” and “the most difficult and exhausing part”. They were divided about the benefits of tactile feedback in controlling position. About the feedback, they said that it “helps determining the amount of acceleration.”, making it easier to stop in time and that it was “really good, because you feel the speed more and you can ‘brake’ better”. Another participant disagreed, saying “tactile may have made it harder. It was a bit distracting. I was trying to focus on making the cursor go somewhere visually and the tactile was not giving me any more information.”.

Yet another participant suggested an alternative implementation of the tactile feedback for position. Rather than feeling 'clicks' based on how far he moved his fingers away from the centre point, he would like to feel a click for every few pixels the cursor on the screen moves.

One participant who did not like the tactile feedback for both the colour and the position controls explained his preference as follows: "I don't like feedback with scrolling. Ever. The idea of scrolling is as if you are sliding it over a slippery surface. By adding tactile feedback it is no longer a slippery surface but like a rocky surface. I like it to be smooth.".

Motivation for preferences When asked whether they preferred the system with tactile feedback on the questionnaire, six participants indicated they did, whereas four participants said they did not prefer to have tactile feedback (see Section 5.4.8 for details). In the post-evaluation discussion, they were given the opportunity to elaborate on their preferences.

Participants who preferred tactile feedback generally explained their preferences on a per-control basis (and therefore their remarks have mostly been included in the previous section). More interesting are the motivations for people to prefer a system without tactile feedback.

One participant said the feedback was "just a bit too heavy. . . a bit too strong. If it was less it would have been better.". Another participant said "Tactile feedback did not help me that much. I think it is because not enough of my hand touches the screen, so I cannot really feel the different kinds of feedback. I was so focused on the screen that I did not really pay attention to the shaking.".

The remaining two participants explained their preference in relation to the tasks used in the experiment, with one stating: "I was surprised that I liked it without tactile feedback better. If this would have been a test that involved typing on a keyboard I would have said let's have tactile feedback all the time, because with typing I like the feedback a lot. In a professsional context I would prefer to have feedback, but it depends. For things that involve motion, I would prefer it to be smooth and effortless. If it's something that is discontinuous, like a box that has clear borders or edges, you want that feedback to make sure that you hit the surface that you intended. Also, when you have this tactile feedback in a scrolling type of scenario, I think our natural response is to tense up our hands. I think this tension is not good in the long run. I think it is best to stick to natural metaphors. If in the real world you have keys, they have tactile feedback, whereas you slide something there is not."

The other participant explained why he preferred the interface without tactile feedback as follows: "Tactile feedback is really good. It was easier for me with the tactile feedback, but if I would be playing a game, I would sometimes prefer to do it without the tactile feedback, because then it is harder to play it."

5.4.9 Discussion

In the previous sections, the results of the laboratory experiment have been described in detail. The purpose of the experiment was to better understand the effects of tactile feedback on inattentive operation of touch screen controls. Using a prototype interface and the block matching test specifically designed for this purpose, effects on *performance*, *attention* and *user experience* have been tested in relative isolation, for several types of controls and across the different phases of control interaction.

Performance

Performance on the block matching task was measured in two ways: speed and accuracy. It was expected (Hypothesis I) that tactile feedback would improve the overall performance; reducing

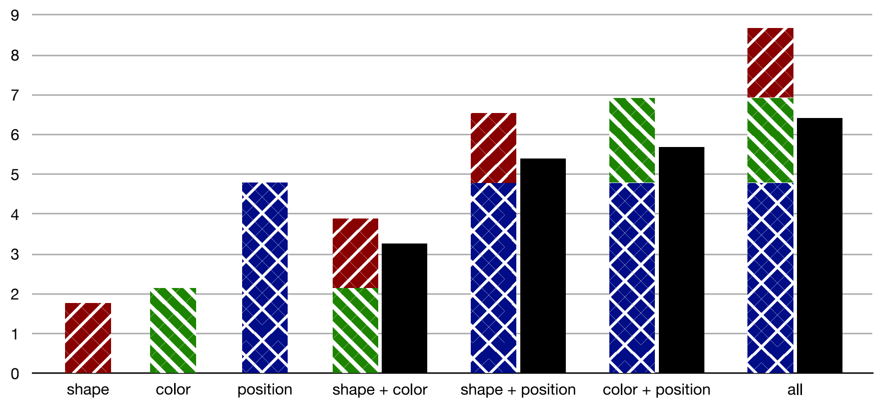


Figure 5.22: Mean duration in seconds (calculated from both sessions) of all task types. For each task that involved adjusting multiple parameters, the black bar on the right represents the time needed to complete that task and the bar on the left shows how much time it took to adjust each parameter individually for comparison.

the time needed to complete the tasks, as well as the amount and magnitude of errors made.

Taken over the whole test, which took participants on average a little under six and a half minutes, participants were less than three seconds faster when tactile feedback was present than when it was not – not a significant difference. A closer look at the durations per task, shows that this remarkably small difference can almost completely be attributed to participants performing slightly faster in the tasks that required all three parameters to be manipulated. The second time they performed these tasks, it took them on average 581 ms per task less with tactile feedback than without, which was significantly faster (by about 9%).

Perhaps the most remarkable about the results of the speed measurements, is the apparent lack of any difference in all the other tasks. Participants perform virtually exactly the same with or without tactile feedback. The three second difference in average time needed to complete the whole test appears to stem solely from participants performing almost 600 ms per task faster in the last five tasks of the second session.

Participants also made very few errors in general. As such, there were no significant differences in the number of errors or their magnitude.

Overall, tactile feedback did not seem to have a big impact on task performance. Only when repeating the task session a second time was there any noteworthy difference, and even then only for the most complicated task that required all the parameters to be adjusted at once.

A partial explanation for this can be found in the way participants operated the controls. Very rarely were participants observed to rely on tactile feedback alone to find the controls. Even though the virtual textures of the touch screen components would have allowed users to find them by moving their fingers over the panel towards the general area of a control until they felt it, participants generally just quickly glanced down.

There are several factors that likely contributed to this. For one, the task was too forgiving of visual distraction for most parameters. Participants could memorise the target at the beginning of the task and then complete it without keeping an eye on it – the target would keep falling, but that essentially did not change the task.

At the same time, most of the visual feedback given by the system was very direct. The value changes of the shape and colour parameters, as well as the accept function, were visualised very clearly and responsively on the display, making tactile feedback as an extra confirmation a bit redundant. In a more realistic environment there would likely be more lag between activating a control and being able to visually see the effects, as the position control intended to simulate.

The controls were also very forgiving. There were, for example, no negative consequences for leaving the shape or colour control areas while making adjustments. The texture applied around those controls may have helped users prevent leaving the control area, but there was no reason to do so. Real interfaces often cannot avoid having controls closer to the edges of the panel, where the cost of drifting too far can be having to reacquire the control.

Participants were also very good at switching between controls, even relying on vision alone. As Figure 5.22 shows, it took them only little more time to complete tasks where they adjusted all parameters than it took to complete the tasks where only the position needed to be changed.

Finally, the implementation of tactile feedback preferred by participants in the pilot study was such that actually large portions of the control panel had no texture (e.g. they were smooth). With only the buttons being rough, the textures would almost never be felt spontaneously, thus not reminding the user of their existence.

Whatever the reasons, tactile feedback did not make participants complete the tasks significantly faster or with less errors, and it also did not result in a higher perceived performance. Therefore, Hypothesis I (tactile feedback improves task performance) cannot be confirmed.

Attention

One of the goals of this study was to try to improve inattentive touch operation. A touch screen that can be operated blindly, without requiring the user to look at the controls, allows more attention to be spent on the task they are performing using those controls. Tactile feedback was expected to improve the extent to which the touch screen in the experiment could be operated without looking. As a result, it was expected that both the total relative time spent looking away from the primary display and the number of glances would be lowered by the presence of tactile feedback. The mean time distracted was not significantly lower, but the participants in the study were found to look significantly less often away from the screen.

In 100 tasks, they glanced away on average 199 times without feedback, compared to 181 times with tactile feedback. In other words, respectively 1.99 and 1.81 times per task. The difference was biggest in the second session, where participants looked away 1.9 times without and 1.7 times with tactile feedback. This suggests that there is a learning effect for distractions and that this effect is stronger when tactile feedback is present. With a slight but not significant decrease in time looked away and a significant reduction in the number of times distracted, it appears that we can confirm Hypothesis II: Tactile feedback decreases distraction from the task.

Still, very few participants completed the tasks completely without looking. Although the Block Matching Test was designed to seemingly require constant visual attention, the blocks were falling slowly enough that a brief glance at the controls would go unpunished. They were very rarely observed to rely on touch alone to acquire controls. It is likely that participants estimated that feeling around for the controls would take longer than briefly looking down. Looking at the data obtained from the logs of the touch panel and eye tracker, which is summarised in full for each individual participant in Appendix C, brief glances at the beginning and end of tasks (before and after pressing "accept") were still very common, whereas feeling around the touch panel to find controls was not. The benefits of tactile feedback in reducing visual distraction appeared mostly in cases where tactile feedback helped the user detect successful control acquisition and value changes, not in finding the controls themselves.

To further understand how tactile feedback affected different types of controls and tasks, we can look at a breakdown of the averages, shown in Figure 5.16. Adjusting the shape caused the fewest distractions, for example: about 1.3 regardless of whether there was tactile feedback or not. Adjusting colour caused about the same number of glances away from the screen, but the second time, with the help of tactile feedback, only 1.14 distractions were measured. The most distractions were measured for those tasks that required all parameters to be ajusted: participants looked down more than 3 times without and about 2.6 times with tactile feedback.

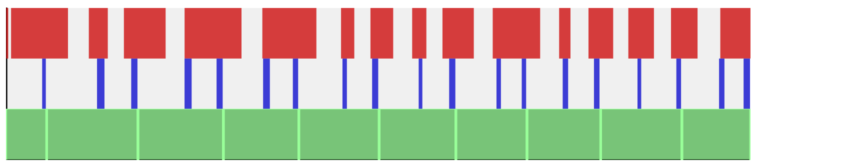
To make sense of these numbers, it is useful to look at the different tasks in the light of the Control Interaction Model of Section 2. Completing any task required at least two controls: one or more parameter controls and the accept button. Both controls needed to be acquired and operated in order to complete the task.

Most participants kept their hands on or near the touch panel throughout the test, so they generally did not have to (re)acquire the control area. The test-setup was such that the same could not be said for the individual controls: participants were only allowed to use one hand, so they could not keep a hand on a control and were forced to go through the "acquire control" phase for every first adjustment in each new task (having last pressed the "accept"-button for the previous task). This was a first opportunity for distraction. Next up was the process of adjusting the control's value: another opportunity to look down. None of the controls was at risk for accidental changes upon being released, so this control phase did not play a role in distraction. Finally, each task was concluded by pressing the "accept"-button. This too needed to be acquired and activated, although this would generally not incur more than one distraction (once the button had been properly acquired, it was a logical consequence that it would also be pressed successfully).

Assuming that participants are able to judge the adjustments based on visual inspection of the primary display alone, this leads to the expectation that participants would look down once for every control they needed to acquire: once for each parameter plus one time extra for the "accept"-button. In other words, one would expect an average of 2 distractions for shape, colour and position; 3 for each of the combinations shape+colour, shape+positon and color+position and 4 for the tasks that required all parameters to be adjusted. In this light, we can look back at the results per task type in Figure 5.16 and see that participants generally managed to complete the tasks with less distractions than expected: on average around 1 distraction per task less without feedback and up to 1.5 with tactile feedback.

Shape Adding tactile feedback to the touch interface was thought to reduce the need to glance down at the controls. After all, tactile feedback allowed the user to feel the boundaries of the control, so the user did not have to worry about accidentally leaving the control area or missing it on their first touch. As a result, it was expected that participants would rely more on their tactile senses than on their vision. In other words, the results were expected to look like in Figure 5.23: without feedback, the participant looks down to find the shape control, then makes adjustments, looks down again for the "accept"-button and presses it. With tactile feedback, there is no need to look down to find the shape control; looking is replaced with feeling. Unfortunately, this was not a common strategy. Most participants adopted a "peck and hunt" strategy and did not drag their fingers across the screen to feel the texture feedback. As a result, they could only feel the tactile feedback from the response layer.

Colour Of all the adjustable parameters, colour was the simplest. It had only five values, placed in order on a linear scale and numbered for reference. Unlike with shape and position, the touch screen provided a visual cue to the current value for colour. This means that for tasks where colour needed to be changed in isolation, the task could be completed looking only at the touch panel after memorising the target value from the task display. Four out of ten participants adopted this as a strategy, regardless of whether tactile feedback was present.


(a) No feedback

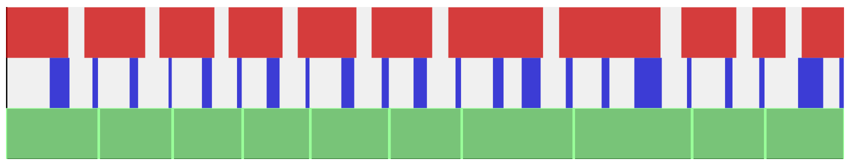

(b) Tactile feedback

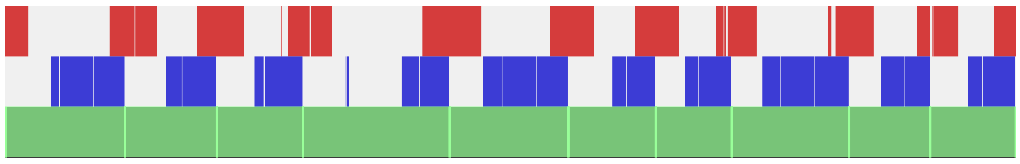
Figure 5.23: Data timeline of a participant completing a series of "shape" tasks under both conditions, showing the effects of relying on tactile feedback on visual distraction. Task boundaries are shown in green at the the bottom, touch contact in the middle (blue) and visual attention at the top in red. Data is from participant 1, session 2 (Appendix C).

Figure 5.24 shows that this strategy lead to long distractions in both conditions. Only one participant (Participant 9) adopted this strategy for the "no feedback", but not the "tactile feedback" sessions. This resulted in significantly less distraction and faster task completion times, as can be seen in Figure 5.25.

The remaining participants were all able to adjust the colour with very few distractions. This may partly be explained by the fact that all but one participant used their right hand to operate the touch screen. This meant that, given the position of the control, the colour could be adjusted with the index finger while keeping the thumb above the "accept"-button (see Figure 5.8). As a result, finding and pressing the button was much easier to do without looking down; once the colour control had been found, the accept button was literally within reach. The majority of participants, including those that were looking at the control panel while changing the colour, was observed to be looking at the screen while pressing the button. In those cases where they mis-pressed the button on their first try, the presence of a tactile confirmation was seen to prevent an extra glance down, which might explain why there were less distractions with tactile feedback than without.

Position The position control was by far the most difficult to control. It took the longest, distracted the most and took the most effort to get right. The position control was almost the opposite of the colour control: it was a gesture, and therefore was not located anywhere on the screen. The gesture could start anywhere on the screen, so there was no need to find a control widget first and here was no way to see the current value on the touch panel. There was really no reason to look at the control panel, as there was simply nothing to see. If anything, the position control was the least forgiving when it came to looking away from the task display (as described in Section 5.4.2 it was specifically designed to require constant visual monitoring).

Looking at the eye-tracker data, an interesting pattern emerges. As expected, most participants do not look at the control panel at the beginning of each task – after all, they have no need to find a specific place to start touching the screen. In line with the other controls, participants also frequently looked down after adjusting the position, in order to find the accept button. Unexpectedly, the position control distracted more than the shape and colour controls. There are two possible explanations for this. It is possible that – while performing the gesture –


(a) No feedback

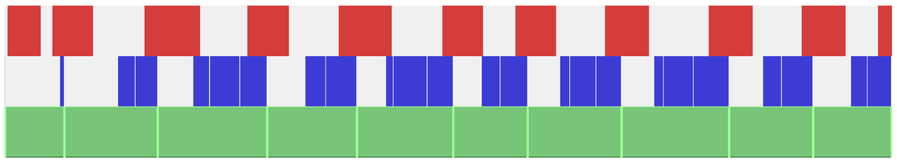
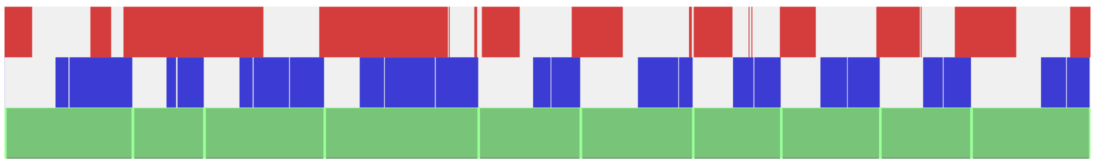

(b) Tactile feedback

Figure 5.24: Data timeline of a participant completing a series of “colour” tasks under both conditions while looking at the control panel in both cases. Task boundaries are shown in green at the the bottom, touch contact in the middle (blue) and visual attention at the top in red. Data is from participant 2, session 2 (Appendix C).


(a) No feedback

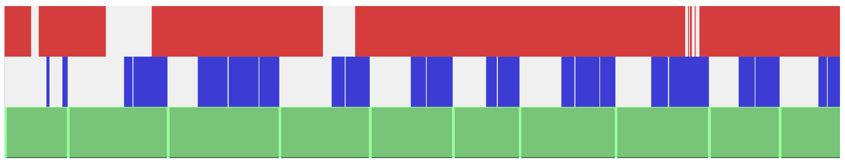

(b) Tactile feedback

Figure 5.25: Relying on tactile feedback for colour reduced distraction. Data is from participant 9, session 2 (Appendix C).

participants moved their fingers outside of the touch sensitive area of the panel and lost control. This would have been easier to recognise with feedback than without, but appeared to rarely happen. A more likely explanation is that upon starting or finishing the gesture, participants inadvertently triggered changes in other controls. Due to the way the prototype was implemented, it was especially sensitive for errors caused by a less-than-perfect initiation of the gesture[4]. This happened on several occasions resulted in the participant having to correct the other value before proceeding. The frequency of this happening was not sensitive to the presence of tactile feedback, but tactile feedback did make it a bit easier to detect.

User Experience

The user experience of touch screen systems is a hard thing to quantify. In this study, information on perceived task load and performance (actual and expected) were collected along with the perceived ease of use and preferences.

Regarding task load, although the perceived load was low in both cases, there unexpectedly seemed to be a slight trend towards participants perceiving the controls with tactile feedback to be slightly more demanding and effortful than those without feedback. They also rated their performance a bit lower with tactile feedback than without, even though they actually did not perform any worse.

A possible explanation for this is that by adding an extra sensory channel, the user is given extra (tactile) information to process, which in turn leads to an increase in perceived demand and effort. However, the one factor that did not differ at all was "mental demand". Another explanation for a possibly higher perceived effort and (mostly physical) demand, is that the tactile feedback creates friction, which makes the touch screen feel less "smooth", as one participant remarked. Several participants commented on the strength of the feedback, which could indicate that more subtle feedback might be perceived as less demanding.

As expected, using the position gesture to control the cursor was perceived as requiring significantly more effort than finding or changing the other controls. Although several participants remarked that they liked having tactile feedback with the position control, the average ratings suggest that participants overall perceived the control to require slightly more effort to operate, although the difference was not significant. One participant indicated that the feedback was distracting and another preferred the screen to be smooth and frictionless. The other control types were all rated roughly the same in both conditions.

The trend of tactile feedback being perceived as slightly harder to use can also be seen in the other ratings. Although there were again no significant differences, it appears that participants found it slightly harder to get the control interface with tactile feedback to do what they wanted than the one without feedback. Overall, the controls were universally rated as easy and quick, but not distracting. Interestingly, participants did not rate the tactile controls as being less distracting, even though they were significantly less distracted when using them.

An interesting contradiction arises when participants are asked directly about the effects tactile feedback had. As described in Section 5.4.8, participants leaned slightly towards answering that tactile feedback improves ease of use, even though they rated the controls on average slightly lower under the tactile condition than without feedback. Half of participants felt strongly that it did improve, whereas 20% felt that it did not. When asked about whether it improved performance – the one factor where they rated the controls significantly lower with tactile feedback

[4]Normally, when a user starts touching the screen, there is a short delay before the touch is recognised, so the system has time to determine whether the user is performing a gesture or interacting with an on-screen control. In adapting the system to generate real-time feedback, this behaviour was lost, resulting in an increase in the likelyhood that performing a gesture triggered other controls.

than without – participants answered overwelmingly affirmative, with 70% saying it did and only 20% saying it did not improve.

When asked whether tactile feedback improved focus, which according to the eye tracker data it did, participants were again divided, with equal parts responding it did or it didn't. It appears that when it comes to tactile feedback, participants either liked it (and 6 of 10 participants did, quite strongly), or they didn't.

Participants who liked tactile feedback, said it helped them operate the controls without looking, recognise whether they touched in the right place – especially for buttons. About the helpfulness of tactile feedback while adjusting the controls, participants were more divided. Some participants said that the feedback helped them stop at the right value, whereas others found the feedback distracting and unneccessary. Participants who disliked tactile feedback in general said that they found the feedback too strong or unpleasant, or simply preferred the touch surface to feel smooth, rather than textured. One participant said he really liked tactile feedback, but because it made the task too easy, he preferred to play the 'game' without it. With a better formulation of the question, this participant's opinion could also have been counted as being in favour of tactile feedback.

All in all it is not crystal clear what the effects of tactile feedback are on user experience. The results suggest that adding tactile feedback may lead to a slight increase in physical demand and effort, that might cause participants to perceive the controls as being a bit harder to operate. At the same time, participants appeared to find the tasks easier to complete with the tactile feedback present and, although they are divided on some issues, the majority clearly leaned towards agreeing that tactile feedback did improve ease of use, focus and especially performance. With 60% of participants quite strongly agreeing to prefer the system with tactile feedback, and 40% not so much, Hypothesis III can be partly accepted. For some users tactile feedback improves the user experience, whereas others prefer more subtle feedback or no tactile feedback at all.

5.4.10 Conclusions

In an attempt to improve the inattentive operation of the specific class of indirect touch screen controls, this study considered tactile feedback as a potential solution. As such, the goal was to understand the effects of tactile feedback on operation those controls, with respect to performance, attention and user experience. A prototype system consisting of a touch panel with a custom user interface and a separate task display, was used in conjunction with the specially designed Block Matching Test to evaluate these effects in a within-subject repeated measures experiment. Participants performed the task once with and once without tactile feedback and reported their perceived task load, ease of use and preferences.

Participants were able to complete the tasks very quickly and with few errors. The study included three different types of controls, each representative of a class of touch screen controls. Participants took the most time adjusting the position, which despite being controlled using a full-screen gesture, was found very difficult because of the indirect nature of control. Under both conditions, with or without feedback, it took very little more time to adjust one, two or three controls, suggesting that there is very little performance overhead overhead caused by switching between controls. Only at the very end of the test, after particpants had been using the controls for a longer time, was there any difference in how fast they performed with tactile feedback and even then only for the most difficult combination of adjustments. Otherwise, participants performed virtually identical whether or not tactile feedback was present.

When it came to visual attention, the data collected using the eye tracker showed several differences. Overall, participants glanced away from the task display signiticantly less often when tactile feedback was present. This effect was strongest in the second series of tasks, which may indicate a learning effect for tactile feedback, and most consistent for tasks involving multiple adjustments. The presence of tactile feedback led some participants to adopt different strategies, where they hardly ever needed to look at the controls. However, even with tactile feedback, participants generally still briefly glanced down at least once per task; especially at the start and end of tasks. In addition to being distracted less times, participants on average also appeared to spend a smaller proportion of time looking away from the screen. Although this difference was not large enough to be significant, it is in line with the other findings that tactile feedback reduced distraction.

Although tactile feedback seemed to slightly increase the perceived effort needed to operate the controls, most participants found that it did improve ease of use, focus and performance. They were divided on whether they liked the feedback or not. Some of them did not like that the feedback made the otherwise smooth glass touch panel feel rough and textured, whereas others indicated the feedback was simply too strong for them. The participants who preferred tactile feedback did so quite strongly, whereas those that did not were generally able to indicate how the feedback could be changed to better suit their liking.

There were some limitations in the design of the task and prototype used in the study. First of all, the block matching task, although specifically designed to require constant visual attention, was too forgiving of brief glances away from the primary display. At the same time, the simplified interface of the control panel did not benefit as much from the possibilities of tactile feedback as would have been possible. Combined with the fact that the participants were new to both the task, the interface and the tactile feedback, and the fact that they had only brief time to get acquinted with the interface, they did not rely on tactile feedback as much as hoped. It is possible that with more practice, tactile feedback would start to improve performance, but such an effect was not seen in this study. Tactile feedback did significantly reduce distraction and allowed participants to focus more on the task; especially after having completed a first series of tasks. More research is needed to study the long term learning effects of tactile feedback to see if this trend persists for attention and whether similar effects exist for performance.

Even in this preliminary form, with only very crude and cheap hardware, tactile feedback was already found to be beneficial for inattentive operation and preferred by the majority of participants. Tactile feedback seems a promising direction for improving indirect touch screen operation. Better tactile actuators, richer feedback and user-controllable feedback strength can further improve the user experience and increase performance.

Chapter 6

Overall Conclusions & Future Work

Touch screens are powerful input devices. The ability to directly interact with a surface that can be changed freely and on the fly by the system opens up all kinds of possibilities. For manufacturers of machines and other systems, a touch screen is an attractive blank slate that can be adapted to their needs. Moreover, because the contents and layout of touch interfaces can be changed, the same space can be re-used for different functions and modes. As a result, touch screens are now found in cars, remote controls and in other places where they are expected to be used while the user is doing something else. Unfortunately, touch screens suffer from some problems that make them less than ideal to be used in such inattentive contexts. The touch screen's greatest strenth is also its greatest weakness. The fact that every touch screen is just a uniform flat surface until it is given meaning by changing the interface displayed on the screen, means that the only way to distinguish between interface elements on the screen is by looking at it. This study aimed to better understand the problems of inattentive operation of peripheral touch screen interfaces and see how it can be improved.

Compared to the other controls and input methods they are often used to replace, touch screen controls lack physicality. Whereas hardware buttons, knobs and toggles have a distinct threedimensional form, their virtual counterparts can only be embodied as pixels on a flat screen. They can only be seen, not felt. As a result, while touch screens can be quite effective for direct input, they are more difficult to use when controlling something indirectly.

The problems with touch screens in an inattentional setting are caused by some of their fundamental properties. Firstly, touch panels are only sensitive to touch. This means that the touch controls that live on a touch panel are also only sensitive to touch. There is no hover state, where the system is aware that the user has positioned their pointer (in this case a finger) over a control, ready to touch it. The flip side of this is that there is also no way to distinguish between a successful and an accidental touch. As a result, the user cannot rest their hands on a touch panel for easy relative movement without causing all kinds of unintended side effects. Touch controls also only respond to being touched directly. The user cannot move over them while touching the screen and activate them that way. The worst problem for inattentive operation, finally, is touch interfaces' reliance on vision and proprioception. Controls only have a visual representation, so no other sensory input is available to help guide the user's movements in the early phases of control interaction. These problems are worst for controls that are embodied as on-screen widgets that require some degree of precision. Gestures, though having problems of their own, sidestep the problems with the acquisition of embodied controls by not being bound to a small area on the screen.

Using global gestures instead of embodied controls potentially reduces the need for visual attention by requiring less precision on the user's part to acquire a specific control. The gesture used determines what it controls, rather than the position on the screen that is touched. These less accurate movements are easier to make using just an initial glance and proprioception. Another approach is to restore a tracking or hover state to the touch interface. This allows the system to start providing users with feedback earlier on, which they can then use to make precise movements. A final solution is to add nonvisual feedback, such as audio or tactile feedback, so the user has an extra sensory channel to rely on.

As part of this study, a prototype system was designed that employed rich vibrotactile feedback to allow users to explore the interface by touch in addition to vision. By giving the controls a tactile representation, consisting of the passive properties texture and height in addition to active tactile responses, the spatial location, separation and behaviour of controls could be perceived by touch. It was expected that because it makes touch controls more like physical controls, this richer tactile feedback would have positive effects on the performance and the amount of visual attention needed for their operation, as well as on their ease of use and other user experience factors.

Using the prototype, participants were asked to perform series of Block Matching Tasks, designed specifically to simulate an environment where visual distraction has negative consequences. In a laboratory experiment, participants were seen to perform equally well on these tasks regardless of whether tactile feedback was present or not (with the exception of tasks that involved operating all the controls at once, which were operated faster with feedback). Participants did however require significantly less glances at the controls when tactile feedback was present. The effects on visual distraction were strongest for the embodied continuous color picker control, and tasks that required several controls to be operated in succession. This suggests that this type of tactile feedback is most beneficial when switching between controls. Additionally, for most types of controls, the effects were stronger after having worked with them longer, which suggests there may be a learning effect for tactile feedback that allows users to rely on it more after becoming more familiar with the layout of the controls. Altough the passive feedback added to the interface allowed users to find controls by touch alone, not many participants did so. More research is needed to find out whether longer exposure to tactile feedback under more demanding conditions would result in users being able to fully blindly operate the touch controls, just as experienced touch typists can type blindly.

Overall, the tactile feedback evaluated in this study was received favourably. Although it seemed to result in a slight increase in the perceived effort needed to operate the controls, it also increased their ease of use and made it easier to focus on the task at hand. Not all participants preferred the tactile feedback, finding the feedback too strong, or preferring it to be off for some controls. Future studies should take preferences of individual users regarding the strenth and nature of feedback into account. It is also likely that better actuators, such as those using electrical rather than vibrotactile stimulation, are able to produce more subtle feedback than the crude linear actuator used for the prototype. Additionally, the possibilities of combining tactile feedback with other solutions should be explored. There are still many other types of controls (some specifically designed for eyes-free operation) and combinations of controls for which the effects of rich dynamic feedback are unknown. In this preliminary explanation, however, adding tactile feedback has shown to be a promising direction for reducing the visual attention needed to operate indirect touch controls.

Acknowledgements

Over the past months as I have been working on this thesis – and it really were plenty of months – I have relied on countless people. From deep discussions about the problems and solutions in this study, to the well-needed moral support and the occasional push, I could always count on my friends, family, fellow students, researchers, interns and colleagues. It's pretty safe to say that if without all of you I would have managed to finish this work at all, it would have been of significantly lower quality. I am indebted to you all.

I would especially like to thank James Lin and Siemens, for generously allowing me the space and freedom to work on a problem I genuinely cared about and providing me with the resources needed to realise the prototype and test-setup needed for the experiment. I would also like to thank Dirk Heylen, Mariet Theune and Betsy van Dijk, for their flexibility, their invaluable feedback and above all their patience. Finally, my thanks go out to Lis, without whom this thesis would have looked different in more than one way. Thanks for putting up with me.

– Michel

Bibliography

[1] T. Ahmaniemi, J. Marila, and V. Lantz. Design of dynamic vibrotactile textures. *IEEE Transactions on Haptics*, 3(4):245–256, 2010.

[2] M. Bassolino, A. Serino, S. Ubaldi, and E. Làdavas. Everyday use of the computer mouse extends peripersonal space representation. *Neuropsychologia*, 48(3):803–811, 2010.

[3] O. Bau and W. E. Mackay. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 37–46, New York, NY, USA, 2008. ACM.

[4] O. Bau, I. Poupyrev, A. Israr, and C. Harrison. TeslaTouch: electrovibration for touch surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 283–292. ACM, 2010.

[5] H. Benko, S. Izadi, A. Wilson, X. Cao, D. Rosenfeld, and K. Hinckley. Design and evaluation of interaction models for multi-touch mice. In *Proceedings of Graphics Interface 2010 on Proceedings of Graphics Interface 2010*, number Figure 1, pages 253–260. Canadian Information Processing Society, 2010.

[6] D. Birnbaum. The touch flute: Exploring roles of vibrotactile feedback in musical performance. *Project report, McGill University*, 2003.

[7] B. Bongers and G. Van Der Veer. Tactual Articulatory Feedback and Gestural Input. *... available at: http://www. cs. vu. ...*, 2008.

[8] S. Brewster, F. Chohan, and L. Brown. Tactile feedback for mobile interactions. *Conference on Human Factors in Computing Systems - Proceedings*, pages 159–162, 2007.

[9] W. Buxton. Lexical and Pragmatic Considerations of Input Structures. *ACM SIGGRAPH Computer Graphics*, (January):31–37, 1983.

[10] W. Buxton. A Three-State Model of Graphical Input. *Human-computer interaction-INTERACT*, pages 449–456, 1990.

[11] S. K. Card, J. D. Mackinlay, and G. G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*, 9(2):99–122, Apr. 1991.

[12] S. K. Card, J. D. Muckinlay, and G. G. Robertson. The design space of input devices. *Proceedings of Chi'90*, (April):117–124, 1990.

[13] A. Crossan, J. Williamson, and S. Brewster. Artex : Artificial Textures from Everyday Surfaces for Touchscreens. *CHI 2010*, pages 4081–4086, 2010.

[14] F. Echtler, M. Huber, and G. Klinker. Shadow tracking on multi-touch tables. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '08, pages 388–391, New York, NY, USA, 2008. ACM.

[15] R. Ecker, V. Broy, A. Butz, and A. De Luca. pieTouch: a direct touch gesture interface for interacting with in-vehicle information systems. In *Proceedings of the 11th international Conference on Human-Computer interaction with Mobile Devices and Services*, pages 1–10. ACM, 2009.

[16] D. Elliott, W. F. Helsen, and R. Chua. A century later: Woodworth's (1899) two-component model of goal-directed aiming. *Psychological Bulletin*, 127(3):342–357, 2001.

[17] G. Faconti and M. Massink. Two Formal Approaches to Modelling Cognitive Aspects of HCI.

[18] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 121(3):262–9, Sept. 1954.

[19] C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–656. ACM, 2007.

[20] D. Freeman, H. Benko, M. R. Morris, and D. Wigdor. Shadowguides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 165–172, New York, NY, USA, 2009. ACM.

[21] Google. *Android 2.3 Compatibility Definition*, 2010.

[22] S. Graf, W. Spiessl, A. Schmidt, A. Winter, and G. Rigoll. In-car interaction using search-based user interfaces. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1685, 2008.

[23] S. Han and J. Park. A study on touch &#38; hover based interaction for zooming. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12, pages 2183–2188, New York, NY, USA, 2012. ACM.

[24] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: design and evaluation of one-, two-and three-touch techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1147–1156. ACM, 2007.

[25] C. Harrison. Texture displays: a passive approach to tactile presentation. *Proceedings of CHI 2009*, pages 2261–2264, 2009.

[26] C. Harrison and S. E. Hudson. Providing dynamically changeable physical buttons on a visual display. *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, page 299, 2009.

[27] S. Hart. NASA-task load index (NASA-TLX); 20 years later. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 50, pages 904–908. Human Factors and Ergonomics Society, 2006.

[28] S. G. Hart, M. F. California, and L. E. Staveland. *Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research.* 1988.

[29] M. Hirsch, D. Lanman, and H. Holtzman. BiDi screen: a thin, depth-sensing LCD for 3D interaction using light fields. *ACM Transactions on Graphics*, 28(5):1–10, 2009.

[30] K. Hofmeester and J. Wolfe. Self-Revealing Gestures: Teaching New Touch Interactions in Windows 8. Microsoft Techdays 2012, The Netherlands, `http://channel9.msdn.com/Events/TechDays/Techdays-2012-the-Netherlands/2373`, February 2012.

[31] E. Hoggan, S. a. Brewster, and J. Johnston. Investigating the effectiveness of tactile feedback for mobile touchscreens. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1573, 2008.

[32] R. Howe and M. Cutkosky. Sensing skin acceleration for slip and texture perception. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 145–150. IEEE, 1989.

[33] T. Kaaresoja, L. Brown, and J. Linjama. Snap-Crackle-Pop: Tactile feedback for mobile touch screens. In *Proc Eurohaptics*, volume 6, pages 565–566. Citeseer, 2006.

[34] M. Karam and M. Schraefel. A taxonomy of Gestures in Human Computer Interaction. *ACM Transactions on Computer-Human Interactions*, pages 1–45, 2005.

[35] K. Kin and M. Agrawala. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. *Proceedings of Graphics Interface*, 2009.

[36] M. Konyo, S. Tadokoro, A. Yoshida, and N. Saiwaki. A tactile synthesis method using multiple frequency vibrations for representing virtual touch. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3965–3971, 2005.

[37] E. Koskinen and T. Kaaresoja. Feel-good touch: finding the most pleasant tactile feedback for a mobile touch screen button. *Proceedings of the 10th International Conference on Multimodal Interfaces (ICMI 2008)*, pages 297–304, 2008.

[38] A. Kun, T. Paek, v. Medenica, N. Memarović, and O. Palinko. Glancing at personal navigation devices can affect driving: experimental results and design implications. In *Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI)*, pages 129–136. ACM, 2009.

[39] K.-U. Kyung, J.-Y. Lee, and M. A. Srinivasan. Precise manipulation of GUI on a touch screen with haptic cues. In *Proceedings - 3rd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, World Haptics 2009*, 3rd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, World Haptics 2009, pages 202–207, IT Technology Convergence Lab, Electronics and Telecommunications Research Institute(ETRI), 161 Gajeong-dong, Yuseong-gu, Daejeon, South Korea, 2009.

[40] S. Lao and P. Wang. A Gestural Interaction Design Model for Multi-touch Displays. *People and Computers*, pages 440–446, 2009.

[41] S. Lee and S. Zhai. The performance of touch screen soft buttons. *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, page 309, 2009.

[42] B. Lippincott, J. Morris, and J. Mueller. Keeping in touch: Smartphone touchscreens and customers with disabilities. *Archves of the Rehabilitation Engineering Research Center for Wireless Technologies (Wireless RERC)*, 2009.

[43] M. H. Lopez, S. Castelluci, and I. S. Mackenzie. Text Entry with the Apple iPhone and the Nintendo Wii. *Proceedings of CHI 2009*, 2009.

[44] I. S. MacKenzie and W. Buxton. Extending Fitts' law to two-dimensional tasks. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '92*, pages 219–226, 1992.

[45] J. Mackinlay and S. Card. A semantic analysis of the design space of input devices. *Human-Computer Interaction*, 5(1984):145–190, 1990.

[46] S. Malik. An Exploration of Multi-finger Interaction on Multi-touch Surfaces. *Computer*, 2007.

[47] A. Martinet, G. Casiez, and L. Grisoni. 3D positioning techniques for multi-touch displays. *of the 16th ACM Symposium on*, pages 227–228, 2009.

[48] A. Martinet, G. Casiez, and L. Grisoni. The effect of DOF separation in 3D manipulation tasks with multi-touch displays. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, pages 111–118. ACM, 2010.

[49] D. McGookin, S. Brewster, and W. Jiang. Investigating touchscreen accessibility for people with visual impairments. *Proceedings of the 5th Nordic conference on Human-computer interaction building bridges - NordiCHI '08*, page 298, 2008.

[50] B. Mcgrath, A. Mckinley, M. Duistermaat, O. Carlander, C. Brill, G. Zets, and J. B. F. V. Erp. Tactile displays for orientation, navigation and communication in air, sea and land environments. Technical Report Chapter 4: Tactile Actuator Technology, North Atlantic Treaty Organisation Research and Technology Organisation (NATO RTO), August 2008.

[51] M. Micire, J. Drury, B. Keyes, and H. Yanco. Multi-touch interaction for robot control. *Proceedings of the 14th ...*, pages 425–428, 2009.

[52] K. Myles and M. Binseel. The Tactile Modality: A Review of Tactile Sensitivity and Human Tactile Interfaces. Technical Report May, U.S. Army Research Laboratory Human Research and Engineering Directorate, 2007.

[53] M. A. Nacenta, P. Baudisch, H. Benko, and A. Wilson. Separability of spatial manipulations in multi-touch interfaces. In *Proceedings of Graphics Interface 2009*, GI '09, pages 175–182, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[54] R. Olson, R. Hanowski, and J. Hickman. Driver distraction in commercial vehicle operations. (September), 2009.

[55] A. Olwal and S. Feiner. Rubbing the Fisheye: precise touch-screen interaction with gestures and fisheye views. In *In Conf. Supp. of UIST'03*, volume 03, pages 83–84. Citeseer, 2003.

[56] A. Olwal, S. Feiner, and S. Heyman. Rubbing and tapping for precise and rapid selection on touch-screen displays. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, 2008:295, 2008.

[57] Opera. Press release: Opera 5.12 for windows in korean. `http://www.opera.com/press/releases/2001/09/10/`, September 2001.

[58] P. Parente and G. Bishop. BATS: the blind audio tactile mapping system. *Proc. of the ACM Southeast Regional Conference*, 2003.

[59] M. J. Pitts, M. a. Williams, T. Wellings, and A. Attridge. Assessing subjective response to haptic feedback in automotive touchscreens. *Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications - AutomotiveUI '09*, (AutomotiveUI):11, 2009.

[60] I. Poupyrev and S. Maruyama. Tactile interfaces for small touch screens. *Proceedings of the 16th annual ACM symposium on User interface software and technology - UIST '03*, pages 217–220, 2003.

[61] I. Poupyrev, S. Maruyama, and J. Rekimoto. Ambient touch: designing tactile interfaces for handheld devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, volume 4, pages 51–60. ACM, 2002.

[62] Research In Motion. Blackberry Storm 9500 User Guide, 2009.

[63] A. Richards, E. M Hannon, and M. Vitkovitch. Distracted by distractors: eye movements in a dynamic Inattentional Blindness task. *Consciousness and cognition*, 21(1):170–6, Mar. 2012.

[64] H. Richter, R. Ecker, and C. Deisler. HapTouch and the 2+ 1 state model: potentials of haptic feedback on touch based in-vehicle information systems. *Proceedings of AutomotiveUI'10*, 2010.

[65] W. a. Rogers, A. D. Fisk, A. C. McLaughlin, and R. Pak. Touch a Screen or Turn a Knob: Choosing the Best Device for the Job. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 47(2):271–288, Apr. 2005.

[66] J. Rovan and V. Hayward. Typology of tactile sounds and their synthesis in gesture-driven computer music performance. *Trends in Gestural Control of Music*, pages 297–320, 2000.

[67] K. Salisbury, D. Brock, T. Massie, and N. Swarup. Haptic rendering: Programming touch interaction with virtual objects. *Proceedings of the*, pages 123–130, 1995.

[68] M. Schedlbauer. A Survey of Manual Input Devices. Technical report, Citeseer, 2007.

[69] J. Schöning, J. Hook, N. Motamedi, P. Olivier, F. Echtler, P. Brandl, L. Muller, F. Daiber, O. Hilliges, M. Loechtefeld, T. Roth, and D. Schmidt. Building Interactive Multi-Touch Surfaces. In C. Mueller-Tomfelde, editor, *Tabletops - Horizontal Interactive Displays*, volume 14, chapter 2, pages 35–55. 2009.

[70] Senseg. E-sense. http://senseg.com.

[71] R. W. Soukoreff and I. S. MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts law research in HCI. *International Journal of Human-Computer Studies*, 61(6):751–789, Dec. 2004.

[72] V. Venkatesh and F. D. Davis. A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science*, 46(2):186–204, 2000.

[73] V. Venkatesh, M. Morris, G. Davis, and F. Davis. User acceptance of information technology: Toward a unified view. *MIS quarterly*, 27(3):425–478, 2003.

[74] M. Weiss and R. Jennings. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. *Proceedings of the 27th ...*, pages 481–490, 2009.

[75] R. Woodworth. *The accuracy of voluntary movement.* Columbia University., New York, July 1899.

[76] T. Yamauchi, M. Konyo, S. Okamoto, and S. Tadokoro. Virtual active touch: Perceived roughness through a pointing-stick-type tactile interface. In *Proceedings - 3rd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, World Haptics 2009*, 3rd Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, World Haptics 2009, pages 605–610, Tohoku University, 2009.

[77] K. Yatani and K. Truong. SemFeel: a user interface with semantic tactile feedback for mobile touch-screen devices. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 111–120. ACM, 2009.

[78] L. Zaman, D. Natapov, and R. Teather. Touchscreens vs. Traditional Controllers in Handheld Gaming. In *Proceedings of the 2010 Conference on FuturePlay*, pages 207–214, 2010.

[79] S. Zhai, S. Conversy, M. Beaudouin-Lafon, and Y. Guiard. Human on-line response to target expansion. *Proceedings of the conference on Human factors in computing systems - CHI '03*, page 177, 2003.

[80] S. Zhai and P. Milgram. Quantifying coordination in multiple DOF movement and its application to evaluating 6 DOF input devices. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '98*, pages 320–327, 1998.

Part III

Appendix

Appendix A

Evaluation Forms

Participant Information

The purpose of the study is to research the effect of feedback on touch screen operation and attention. You will get to try out a prototype interface and give your opinion. During the evaluation, you will also be asked to perform some tasks to gain insight into the way tactile feedback affects performance. Note that the purpose of the study is not to evaluate you or your performance, but that of the prototype. All identifiable data will be treated confidentially.

During the experiment, data is collected for analysis. This includes video, eye-tracking, and touch data. This too will be treated confidentially. All data from individual participants will be coded so that their anonymity will be protected in any reports, research papers, thesis documents, and presentations that result from this work.

There is no compensation for participating in the study. Your participation is entirely voluntary and you may refuse to participate or withdraw from the study at any time, without any obligations. By proceeding with the experiment, your consent is assumed.

The evaluation will take about an hour, first we ask you to provide some demographic information, after which you get to explore the prototype on your own. Then we will have a short training before we give you some tasks to complete. Finally, we will ask you to fill out a questionnaire and give your opinion.

1. Age

   ____

2. Gender

   ____

3. Handedness (left/right)

   ____

4. Which touch devices, such as smart phones, tablets, car navigation systems, remote controls etc. do you own or use regularly?

   ____

   ____

5. How much experienced do you have using touch devices?

   no experience ☐ ☐ ☐ ☐ ☐ ☐ ☐ much experience

6. How much experience do you have playing video games?

   no experience ☐ ☐ ☐ ☐ ☐ ☐ ☐ much experience

Overall Workload – After Session 1a

1. Overall, I believe that the controls are easy to use.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

2. The controls distract me from my primary task.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

Overall Workload – After Session 1a

Please rate your overall experience of performing the task on the following factors by marking the scale. You can mark anywhere on the line from low to high.

Mental Demand

Low High

How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?

Physical Demand

Low High

How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Temporal Demand

Low High

How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Performance

Low High

How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

Effort

Low High

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration

Low High

How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Overall Workload – After Session 1b

1. Overall, I believe that the controls are easy to use.

   strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

2. The controls distract me from my primary task.

   strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

Overall Workload – After Session 1b

Please rate your overall experience of performing the task on the following factors by marking the scale. You can mark anywhere on the line from low to high.

Mental Demand

Low High

How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?

Physical Demand

Low High

How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Temporal Demand

Low High

How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Performance

Low High

How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

Effort

Low High

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration

Low High

How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Overall Experience – After Session 2a
Please answer the following questions regarding your overall experience with the controls. Mark any box between "strongly disagree" or "strongly agree".

1. The controls enable me to accomplish tasks quickly.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

2. Overall, I believe that the controls are easy to use.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

3. I would find it easy to get a system with these controls to do what I want it to do.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

4. The controls distract me from my primary task.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

Overall Workload – After Session 2a

Please rate your overall experience of performing the task on the following factors by marking the scale. You can mark anywhere on the line from low to high.

Mental Demand

Low High

How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?

Physical Demand

Low High

How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Temporal Demand

Low High

How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Performance

Low High

How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

Effort

Low High

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration

Low High

How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Individual controls – After Session 2a

Please rate your experience with the controls for each parameter. You can mark anywhere from low to high.

Shape

Effort to find the control for shape

Low High

Effort to change the shape

Low High

Color

Effort to find the control for color

Low High

Effort to change the color

Low High

Accept

Effort to find the button to accept

Low High

Effort to use the button to accept

Low High

Position

Effort to change the position

Low High

Overall Experience – After Session 2b

Please answer the following questions regarding your overall experience with the controls. Mark any box between “strongly disagree” or “strongly agree”.

1. The controls enable me to accomplish tasks quickly.

   strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

2. Overall, I believe that the controls are easy to use.

   strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

3. I would find it easy to get a system with these controls to do what I want it to do.

   strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

4. The controls distract me from my primary task.

   strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

Overall Workload – After Session 2b

Please rate your overall experience of performing the task on the following factors by marking the scale. You can mark anywhere on the line from low to high.

Mental Demand

Low High

How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?

Physical Demand

Low High

How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Temporal Demand

Low High

How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Performance

Low High

How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

Effort

Low High

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Frustration

Low High

How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Individual controls – After Session 2b

Please rate your experience with the controls for each parameter. You can mark anywhere from low to high.

Shape

Effort to find the control for shape

Low High

Effort to change the shape

Low High

Color

Effort to find the control for color

Low High

Effort to change the color

Low High

Accept

Effort to find the button to accept

Low High

Effort to use the button to accept

Low High

Position

Effort to change the position

Low High

Compare - After session 2
Please answer the following questions regarding your overall experience with the controls. Mark any box between “strongly disagree” or “strongly agree”.

1. Tactile feedback improved my performance.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

2. Tactile feedback made the controls easier to use.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

3. Tactile feedback allowed me to focus more on my primary task

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

4. I prefer the system with tactile feedback over the system without tactile feedback.

strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ strongly agree

Appendix B

Evaluation task script

B.1 Practice

1. shape: 7
2. shape: 1
3. shape: 4
4. shape: 5
5. shape: 7
6. break
7. color: 0
8. color: 2
9. color: 4
10. color: 1
11. color: 3
12. break
13. position: 0
14. position: 14
15. position: 8
16. position: 1
17. position: 2
18. break
19. shape: 1
20. shape: 5
21. shape: 4
22. shape: 7
23. shape: 3
24. break
25. color: 3
26. color: 1
27. color: 4
28. color: 2
29. color: 0
30. break
31. position: 3
32. position: 13

33. position: 4
34. position: 24
35. position: 23

B.2 Session 1

1. shape: 5
2. shape: 1
3. shape: 3
4. shape: 9
5. shape: 2
6. shape: 7
7. shape: 5
8. shape: 4
9. shape: 1
10. shape: 6
11. break
12. color: 0
13. color: 1
14. color: 4
15. color: 3
16. color: 0
17. color: 1
18. color: 4
19. color: 0
20. color: 2
21. color: 3
22. break
23. position: 12
24. position: 13
25. position: 22
26. position: 21
27. position: 10
28. position: 9
29. position: 8
30. position: 3
31. position: 4
32. position: 15
33. break
34. shape: 1, color: 4
35. shape: 5, color: 3
36. shape: 2, color: 0
37. shape: 3, color: 1
38. shape: 8, color: 4
39. break
40. shape: 5, position: 12
41. shape: 1, position: 13
42. shape: 3, position: 22

43. shape: 9, position: 21
44. shape: 2, position: 10
45. break
46. color: 0, position: 9
47. color: 1, position: 8
48. color: 4, position: 3
49. color: 3, position: 4
50. color: 0, position: 15
51. break
52. shape: 2, color: 0, position: 10
53. shape: 9, color: 3, position: 21
54. shape: 3, color: 4, position: 22
55. shape: 1, color: 1, position: 13
56. shape: 5, color: 0, position: 12

B.3 Session 2

1. shape: 5
2. shape: 3
3. shape: 9
4. shape: 7
5. shape: 6
6. shape: 1
7. shape: 5
8. shape: 2
9. shape: 3
10. shape: 8
11. break
12. color: 4
13. color: 3
14. color: 0
15. color: 1
16. color: 4
17. color: 3
18. color: 0
19. color: 4
20. color: 2
21. color: 1
22. break
23. position: 10
24. position: 15
25. position: 20
26. position: 19
27. position: 8
28. position: 7
29. position: 6
30. position: 1
31. position: 2

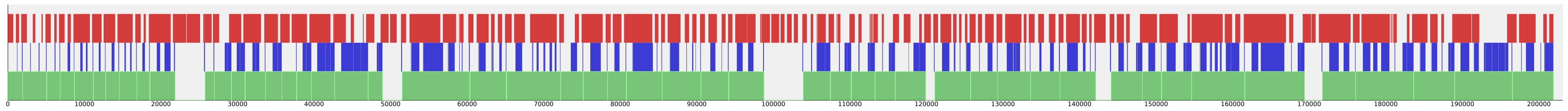
32. position: 13
33. break
34. shape: 7, color: 0
35. shape: 5, color: 1
36. shape: 4, color: 4
37. shape: 1, color: 3
38. shape: 6, color: 0
39. break,
40. shape: 5, position: 10
41. shape: 3, position: 15
42. shape: 9, position: 20
43. shape: 7, position: 19
44. shape: 6, position: 8
45. break
46. color: 3, position: 7
47. color: 0, position: 6
48. color: 4, position: 1
49. color: 2, position: 2
50. color: 1, position: 13
51. break
52. shape: 7, color: 4, position: 19
53. shape: 9, color: 1, position: 20
54. shape: 3, color: 0, position: 15
55. shape: 5, color: 3, position: 10
56. shape: 6, color: 4, position: 8

Appendix C

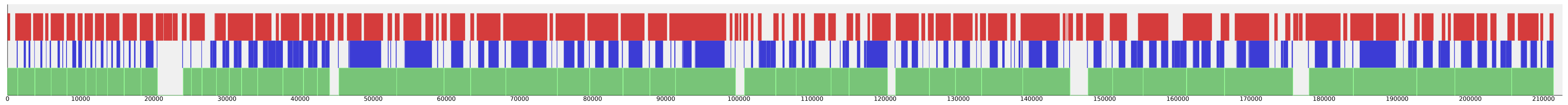
Evaluation Session Logs

During the task sessions, information about visual and touch activity was recorded using an eye tracker. This appendix contains the data from those logs summarised in the form of timelines. The red bars at the top indicate when the user was looking at the task display. A gap means the user looked away. The blue bars in the middle represent touch activity, e.g. when the user was touching the screen or not. The bars at the bottom of the screen represent the individual tasks. The timelines are not corrected for any breaks or gaps, they run from the beginning of the session to the end.

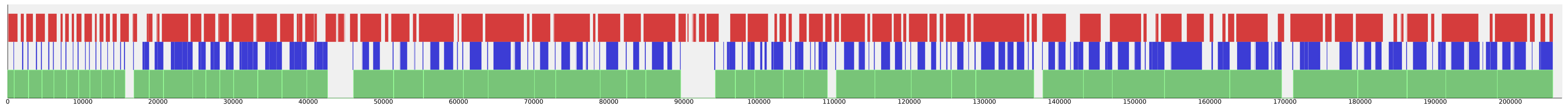
Session Log Participant 1



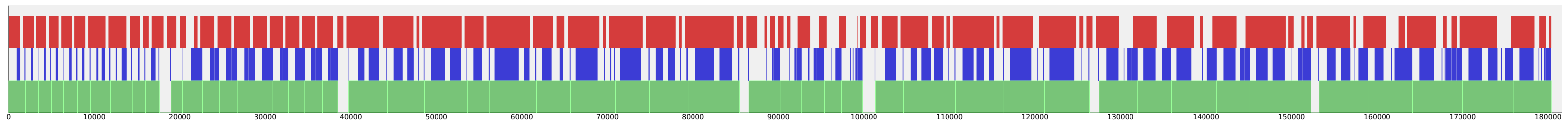
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

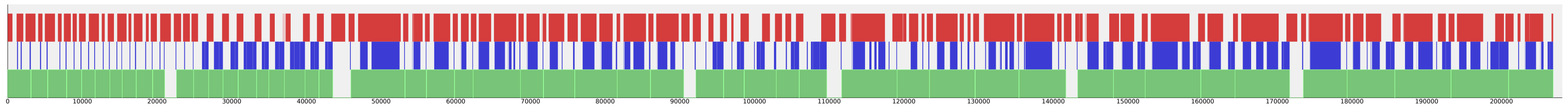


(c) Session 2 without feedback

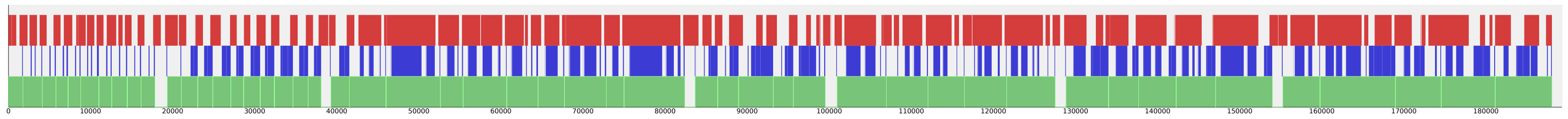


(d) Session 2 with tactile feedback

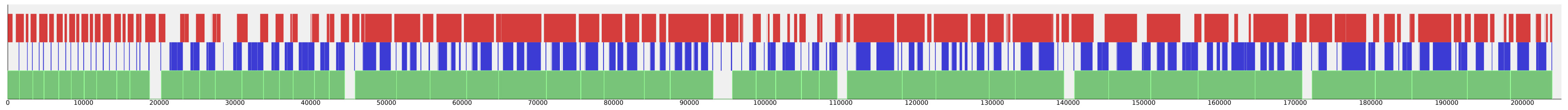
Session Log Participant 2



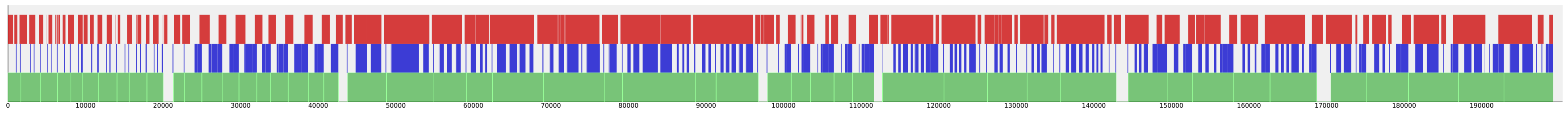
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

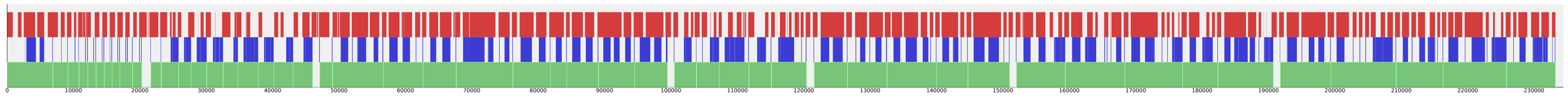


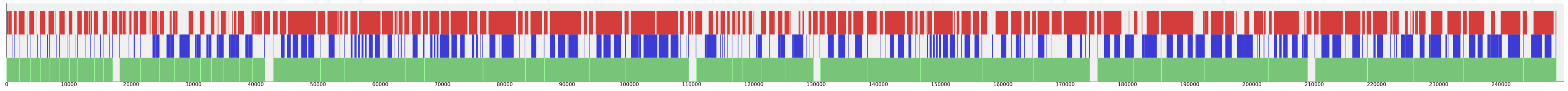
(c) Session 2 without feedback

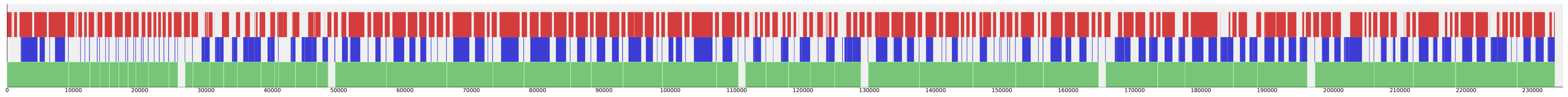


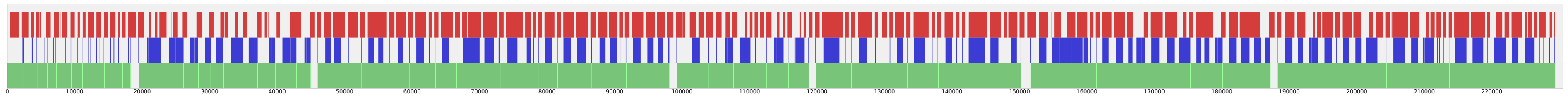
(d) Session 2 with tactile feedback

Session Log Participant 3

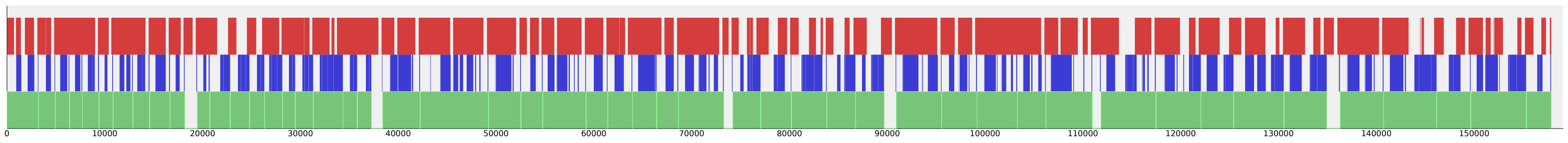

(a) Session 1 without feedback


(b) Session 1 with tactile feedback

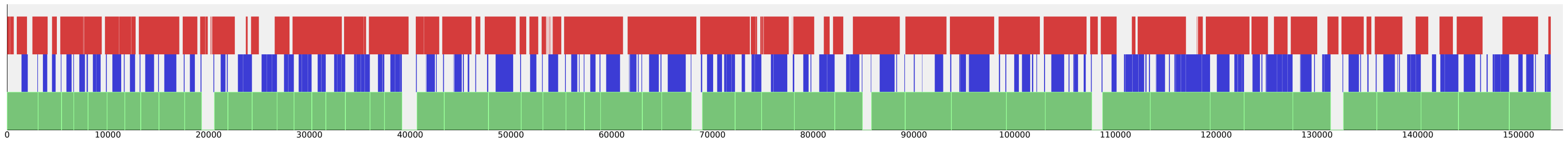

(c) Session 2 without feedback


(d) Session 2 with tactile feedback

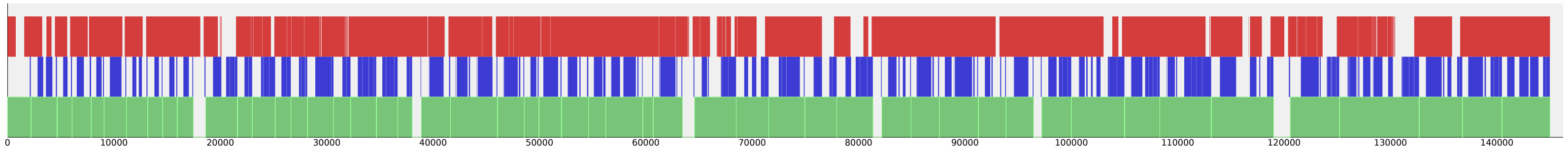
Session Log Participant 4



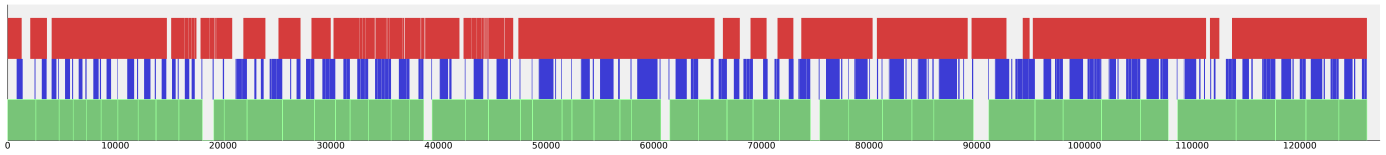
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

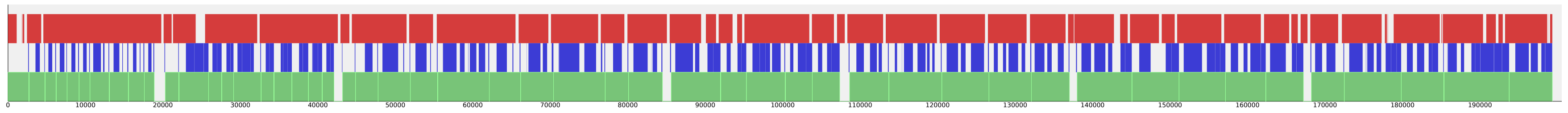


(c) Session 2 without feedback

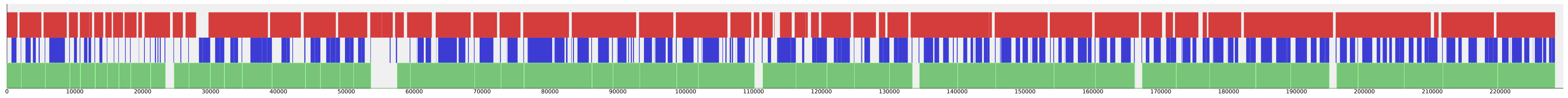


(d) Session 2 with tactile feedback

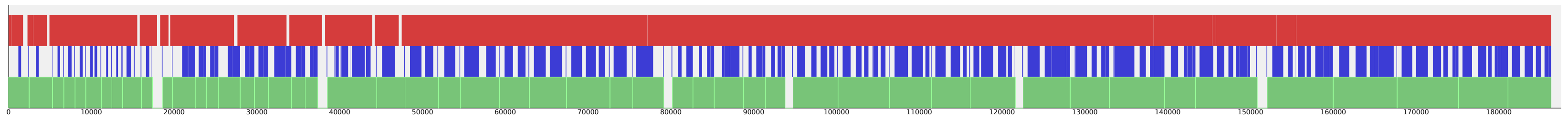
Session Log Participant 5



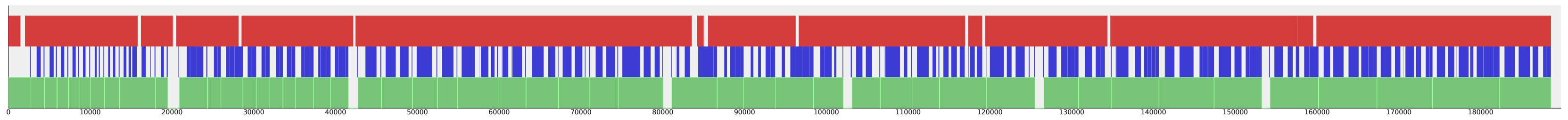
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

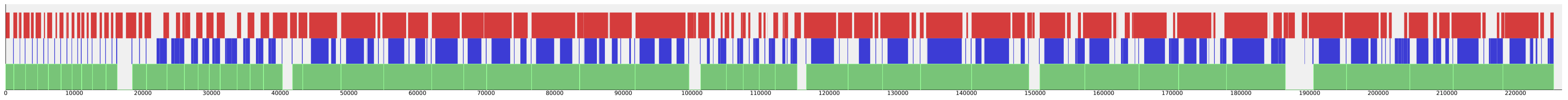


(c) Session 2 without feedback

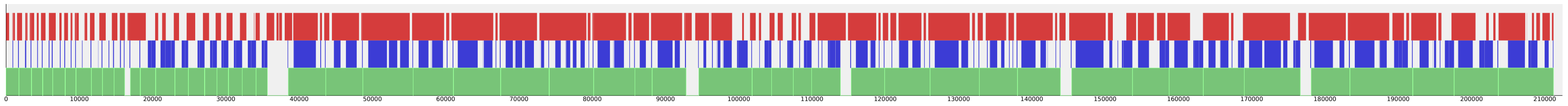


(d) Session 2 with tactile feedback

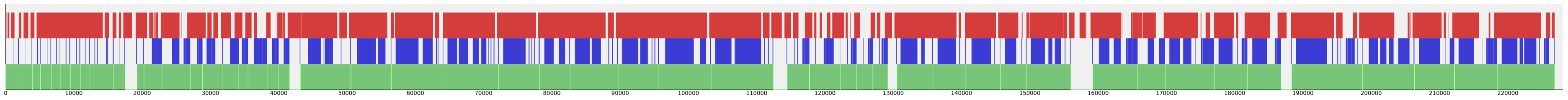
Session Log Participant 6



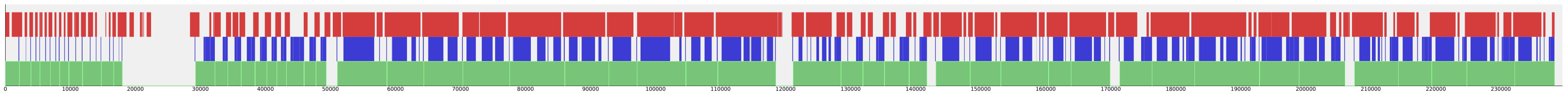
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

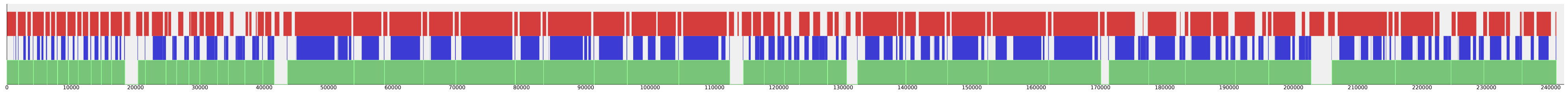


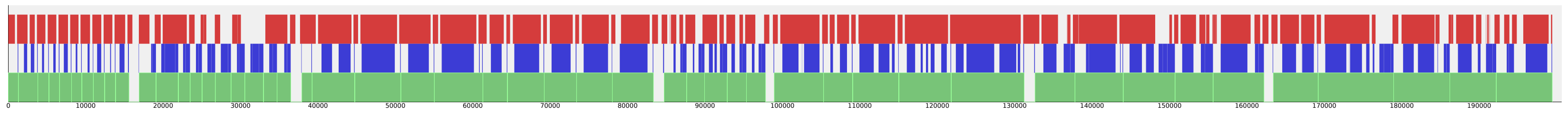
(c) Session 2 without feedback

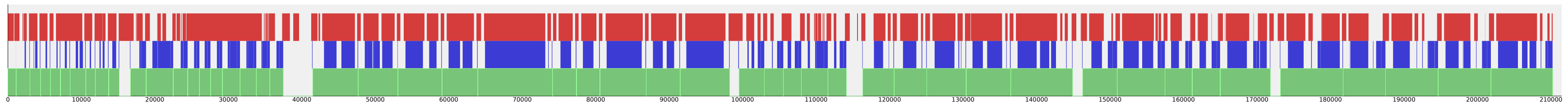


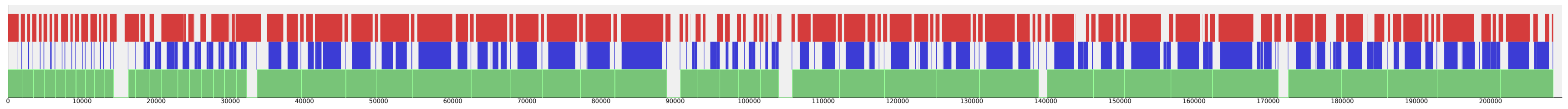
(d) Session 2 with tactile feedback

Session Log Participant 7

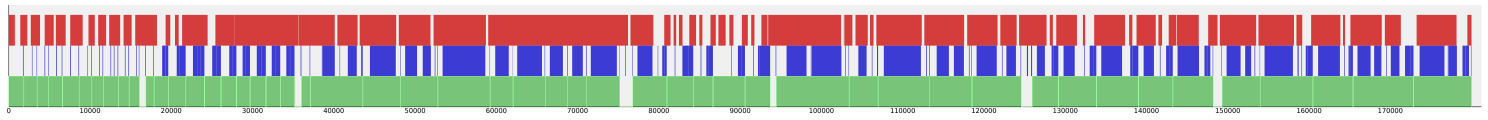

(a) Session 1 without feedback


(b) Session 1 with tactile feedback

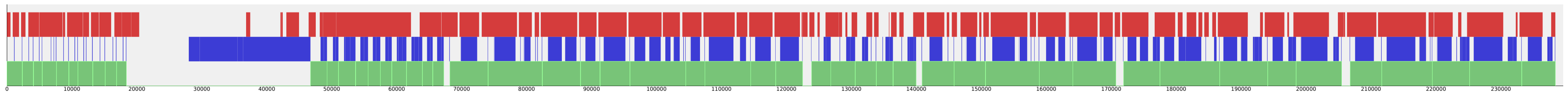

(c) Session 2 without feedback


(d) Session 2 with tactile feedback

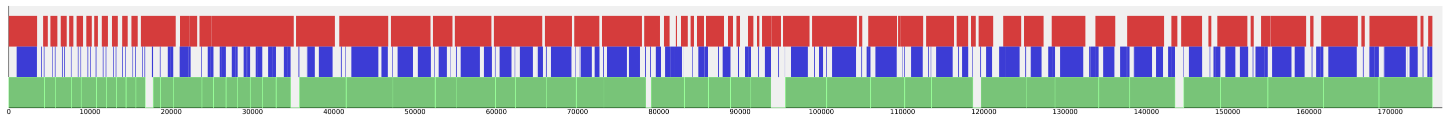
Session Log Participant 8



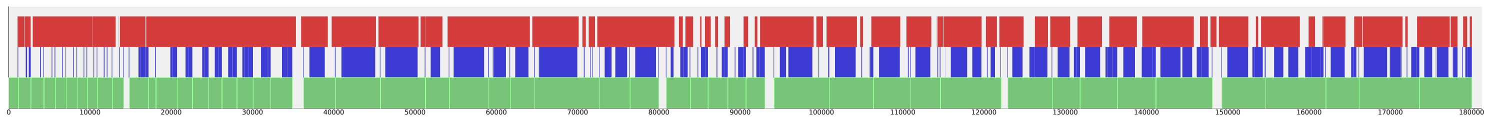
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

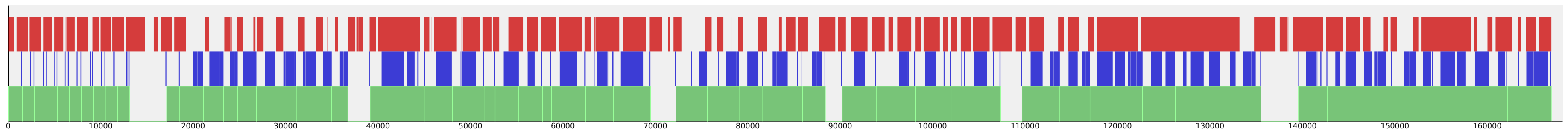


(c) Session 2 without feedback

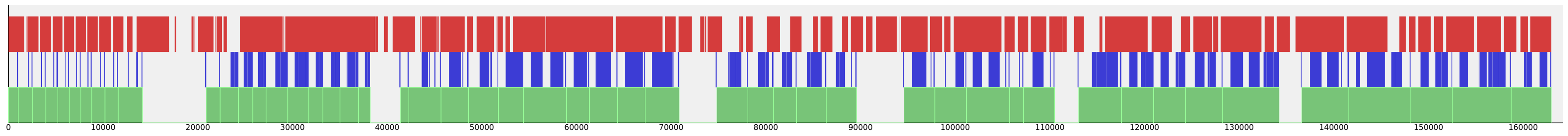


(d) Session 2 with tactile feedback

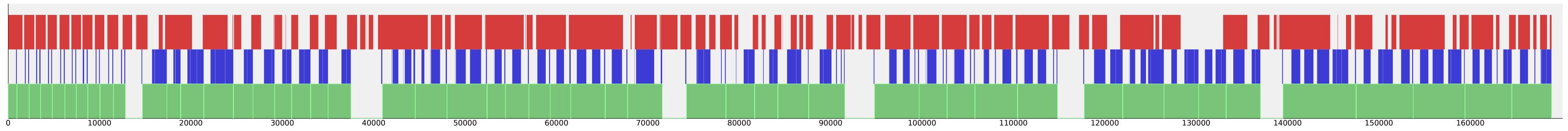
Session Log Participant 9



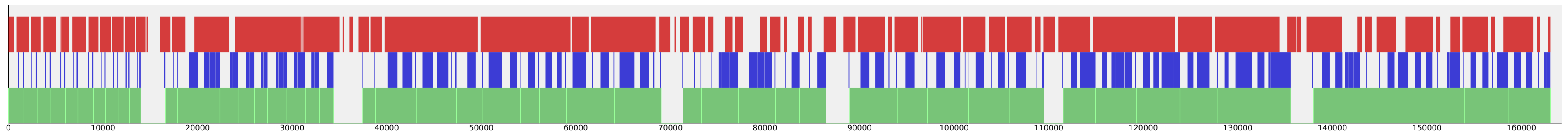
(a) Session 1 without feedback



(b) Session 1 with tactile feedback

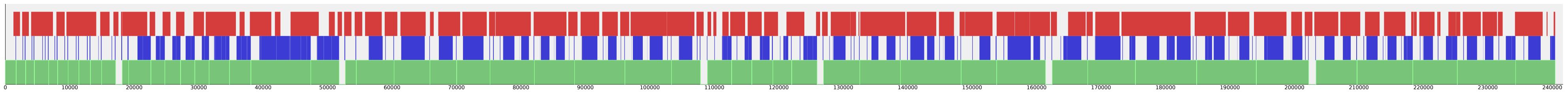


(c) Session 2 without feedback

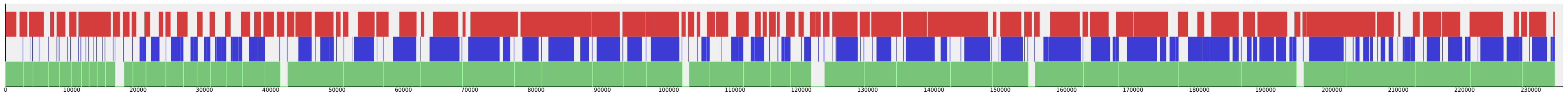


(d) Session 2 with tactile feedback

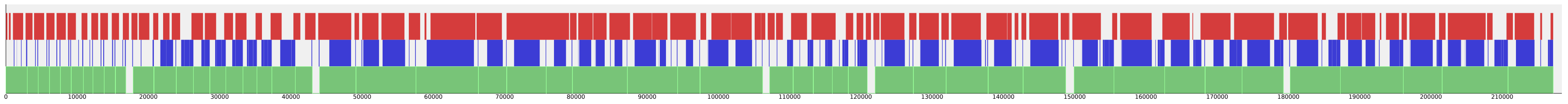
Session Log Participant 10



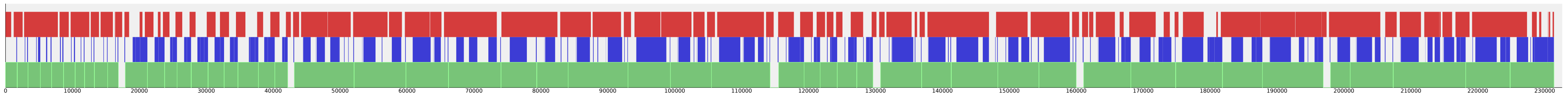
(a) Session 1 without feedback



(b) Session 1 with tactile feedback



(c) Session 2 without feedback



(d) Session 2 with tactile feedback