MULTI-TOUCH INTERFACES AND MAP NAVIGATION

by

Jeremy White

A thesis submitted in partial fulfillment of the

requirements for the degree of

Master of Science

(Cartography and GIS)

at the

UNIVERSITY OF WISCONSIN-MADISON

2009

Table of Contents

Table of Figures ……..……..…… ii

Chapter One: Introduction ……..…….…… 1

Chapter Two: Technology Review ……...…...…… 5

Chapter Three: Literature Review …….…….....…… 11

3.1: Human-Computer Interaction ……..……...……..…… 11

3.2: Map Design and Cognition ……...……...…..…… 14

3.3: Interface Design …….…….…….……15

3.3: Multi-touch Analysis …….…….……....…… 18

Chapter Four: Usability Testing Platform ……..…… 22

4.1: Physical Structure ……..…….…… 23

4.2: Blob detection …….…..…… 29

4:3 Image Input and Map Visualization …….....…… 36

4.4 Custom Gesturing Techniques……..…….…..……39

4.5 DIY Movement …….…….……...…… 42

Chapter Five: Discoveries and Future Direction ……..…....…… 45

5.1: Findings …… 45

5.2: The Market …….…...……...……51

Chapter Six: Conclusion ……...…….……56

Bibliography …….……...……...…… 60

Table of Figures

Figure 2.1: Kasday's Capacitance Interface (1984) ……..……...…9

Figure 2.2: Han's FTIR Interface (2005) ……..……..……..…… 9

Figure 4.1: Compact infrared LED ……...……..…..…… 19

Figure 4.2: Channels added for running wires ……...…...……27

Figure 4.3: Acrylic surface roughened for light diffusion …….……..…… 27

Figure 4.4: LEDs soldered in a series ……....…….…… 27

Figure 4.5: Series of ceramic capacitors …….…….…… 27

Figure 4.6: Modified camera PCBs for IR light detection ……...…….…… 28

Figure 4.7: Camera lens with black cellulose acetate film ……..…..…… 28

Figure 4.8: Housing for modified camera PCBs ……...…….……28

Figure 4.9: Completed usability testing platform …….…...…..…… 28

Figure 4.10: An updated version of the two-pass technique ……....……33

Figure 4.11: Pixel label assignments …….…… 34

Figure 4.12: Original, noisy frame from camera …….…… 35

Figure 4.13: Processed frame after blob detection …….…….…… 35

Figure 4.14: Flowchart showing interaction and visualization …….……39

Figure 5.1: Infrared light escaping through top surface …….…..…… 47

Figure 5.2: No "hover" state is registered from above ……....……47

Figure 5.3: Map features are covered by the user's hand ……..…..…… 48

Figure 5.4: Features remain hidden while panning …….…….……48

Figure 5.5: Menu system covering much of the map ……..…..…… 49

Figure 5.6: Comparison of multi-touch platforms ……..…… 50

Figure 5.7: Touch detection on projected surface ……..…… 53

Figure 5.8: Map navigation through gestures ……...…… 53

Chapter One: Introduction

Development of multi-touch interfaces over the last 25 years has led to their recent adoption into the marketplace. Multi-touch interfaces now allow users to apply several points of contact to a device, an innovation which may provide novel solutions to existing interface limitations. The first publically available multi-touch devices, such as Apple's iPhone and Microsoft's Surface, have introduced the general public to a new form of human-computer (HCI) interaction that may soon become commonplace.

One way multi-touch interfaces will impact the field of geography is through new forms of map navigation. Issues relating to the adoption and best uses of multi-touch will emerge as users are introduced to new map navigation techniques. Historically, designing and testing multi-touch interfaces has been difficult due to the prohibitively high cost and low availability of the necessary software and hardware. This thesis introduces a process for creating a large format, reliable multi-touch interface for the purpose of map navigation. The goal of this project is to

provide insight into the potential impact of multi-touch interfaces on the fields of cartography and GIScience.

Multi-touch interfaces have many advantages over other input devices such as the mouse, keyboard or tablet. Switching between two input devices, such as the mouse and keyboard, increases the time spent on non-productive movement (Westerman & Elias, 2001). The lack of moving parts within a multi-touch interface eliminates any mechanical problems associated with HCI intermediaries. Self-contained multi-touch interfaces present an input option that is always available and requires no additional hardware. Multi-touch interfaces are more scalable than other input devices because of the countless number of combined gestures used in conjunction with other variables such as pressure sensitivity and the ability to simultaneously accept input from multiple users. Using a multi-touch interface, common tasks can be performed with increased speed (Buxton & Smith, 1985), greater accuracy (Arthur et al., 2008), better memory retention (Moscovich, 2007) and decreased user fatigue (Wu et al., 2006).

Digital mapping services, such as Google and Yahoo, have created map navigation methods that are now widely accepted when using a traditional mouse and keyboard. Multi-touch interfaces provide additional functionality (e.g. gesture based controls, simultaneous commands, etc.), which will need to be studied since

not much is known about this nascent form of interaction. For the purpose of this project, I constructed a multi-touch interface, as a usability testing platform, to gather information about the advantages and disadvantages of this new form of HCI.

There were many theoretical and practical steps involved in creating a multi-touch interface as a usability testing platform. First, I reviewed the evolution of multi-touch systems, as well as similar forms of HCI. Several different approaches have been attempted in both academia and the commercial industry in the past 25 years and it was important to consider the benefits and drawbacks of each method. Second, I examined the previous literature on map design and cognition which was important in creating an effective interface for map navigation. Many traditional cartographic principles and interface design considerations can be applied to this new form of HCI, while some modifications need to be made in order to accommodate many new approaches that multi-touch interfaces can provide. Third, I evaluated the technical steps in creating a usability testing platform and separated the process into manageable tasks. The different real-time processes involved with a multi-touch system must all work simultaneously to produce an interface that is adequately responsive. Fourth, I describe how I designed and built a large format multi-touch interface. Fifth, I provide insights and recommendations for multi-

touch interface developments as well as suggestions for improving existing multi-touch gestures.

Chapter Two: Technology Review

Two competing approaches have shaped the development of multi-touch interfaces during the last 25 years. The first approach focuses on the use of optical image analysis to detect surface changes in luminosity. Algorithms have been developed for "blob" detection, which inherit similar methods from motion detection, edge detection and facial recognition. The capacitive approach is a second technique that has been developed that relies on electronics within the surface of the device to register contact points. Generally, multi-touch interfaces that use image analysis, such as Microsoft Surface (http://www.microsoft.com/surface), are better suited for larger formats while the capacitive approach is more applicable to portable devices such as the iPhone due to portability considerations and costs.

One of the earliest working visual multi-touch devices was the Flexible Machine Interface developed by Nimish Mehta in 1982 (Mehta, 1982). His work at the University of Toronto explored the use of projected light onto a frosted glass surface while registering the finger pressure against the glass. Processing the user interaction was performed by an image analysis of the dark areas created by the user's fingers against the white glass, which served as adequate input for

applications that performed simple tasks such as drawing. The results were displayed on a separate monitor instead of being integrated with the glass surface.

Similar research to Mehta's Flexible Machine Interface was conducted in the early 1980's at Bell Labs in Murray Hill, New Jersey (Buxton, 2009). Unlike Mehta's approach, Leonard Kasday and the Bell Labs team (Nakatani & Rohrlich, 1983) developed a multi-touch interface that was integrated with the display itself, which allowed the user to directly control the graphic elements on the screen. The interface was built around the concept of "soft machines" developed by Lloyd Nakatani; loosely defined as graphical interfaces representing hard controls, such as switches (Figure 2.1). Nakatani decided to use a capacitive approach, instead of an optical approach, to detect the presence of the user's fingers against a cathode ray tube (CRT) monitor. The CRT monitor required a capacitive approach because it contained a vacuum tube with an electron gun, instead of a projected light source. The capacitive method proved to be a fast, and therefore transparent, technique for relaying interaction between the hardware and user because there was no need for optical image analysis.

The capacitive approach for detecting contact was further explored at University of Toronto in the form of a multi-touch tablet in 1984. William Buxton, along with the Computer Systems Research Institute, developed a thin tablet that

was capable of detecting an arbitrary number of touch inputs by relaying the location and the force of each touch (Buxton, Hill, & Rowley, 1985). The tablet was considerably smaller than any of the previous devices that had been created because it did not rely upon optical detection hardware. Creating smaller devices using the capacitive approach was an important step forward in building truly portable devices with multi-touch interfaces such as the iPhone. The research conducted at both Bell Labs and the University of Toronto helped shape the development of today's commercially available devices.

Buxton continued to explore alternative methods for sensing touch at Xerox PARC in the early 1990's. Early attempts to create a bi-directional version of a liquid crystal display (LCD) used the company's scanner technology, with added hardware layers, to create alternative displays. Two different devices were created from this process. First, the high resolution color active matrix liquid crystal display (AMLCD) was created by this team, and provided an innovative method for capturing and displaying visual data. However, the use of touch as an input method was dropped in the technology's early stages due to its computational expense and more traditional methods of input were incorporated such as capacitive sensing. The second device born from Buxton's LCD research was the Liveboard Project (Elrod et al., 1992). This device included a multi-state cordless pen to record

the user's actions against a rear projection. The pen allowed the optical detection equipment to register a more precise contact area by solving the inherent issues that are present with varying ambient light levels, such as fluctuation in luminosity and the control of light from the infrared spectrum. Fluctuations in luminosity can cause problems with optical image analysis because each pixel value is compared with previous samples. When the luminosity changes across the entire display surface, the tolerance assigned to the pixels may be reduced, making it more difficult distinguish subtle differences. Controlling light from the infrared spectrum is important because digital cameras can detect light beyond visible wavelengths. Light sources that produce a high ratio of infrared to visible light, such as the sun or high-temperature incandescent bulbs, make it more difficult for an infrared camera to detect changes because more light is present than the camera requires.

More recently, Jefferson Han at NYU's Department of Computer Science, is developing an alternative optical interface using infrared light that has attracted considerable public attention (Figure 2.2) (Han, 2005). Using a method known as frustrated total internal reflection (FTIR), Han was able to track a user's touch by bouncing light from the infrared spectrum off of the finger's contact points and analyze the data with an infrared camera (http://www.ted.com/index.php/talks/jeff_han_demos_his_breakthrough_touchscree

n.html) The FTIR approach is a cost effective alternative to many of the previous techniques for integrating a multi-touch interface and, paramount for widespread adoption, it is also less computationally demanding. Han's approach is more cost-effective because it relies on relatively inexpensive components that do not require custom electronics to be manufactured.

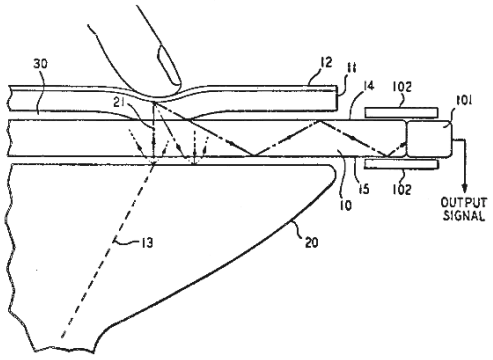

Figure 2.1: Kasday's Capacitance Interface (1984)

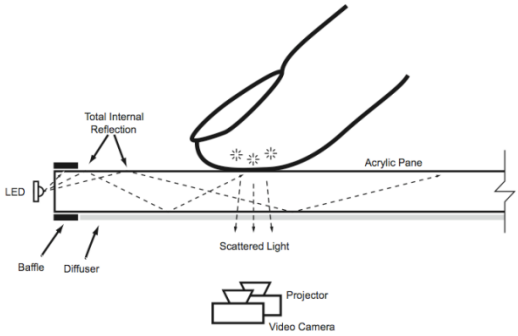

Figure 2.2: Han's FTIR Interface (2005)

The largest distribution of multi-touch interfaces has occurred within the last two years by Apple, with more than 30 million iPhone and iTouch devices sold worldwide from 2007 to 2009 (Zemen, 2009). Both devices can accept multi-point interactions on a limited basis by using techniques developed by Wayne Westerman (Westerman, 1999). The interactive metaphors of "pinching" and "stretching" have been implemented in addition to a single-point drag and scroll. The latest

generation of Apple laptops along with firmware updates have recently been released that allow up to four simultaneous contact points with the trackpad. In January of 2008, Apple introduced the Macbook Air, which includes a multi-touch trackpad with a control panel for configuring the acceptable actions.

Microsoft, with the help of Buxton, has also been conducting research on multi-touch interfaces, and released a product called Surface in April of 2008 (Fried, 2008). Surface is a multi-touch interface initially aimed at the retail market with a 30-inch display in a table sitting at about desk height. Surface includes additional technological advances that extend beyond Buxton's previous work, such as the use of multiple cameras for more precise blob detection and a hardware configuration designed specifically for displaying high resolutions images. In addition to static and active touch sensing, Surface can also identify objects that are placed on top of it. The use of reference objects, in combination with natural and easy to learn hand gestures, expands the total number of recognized inputs. Major limitations to Microsoft's multi-touch interface include the weight of the display (nearly 200 pounds) and the cost per unit, which exceeds $12,000 as of late 2009.

Chapter 3: Literature Review

Human-Computer Interaction

There has been a wide range of research concerning HCI over the last 50 years covering areas such as design principles, methodologies and interface design. While many areas within HCI are related to multi-touch interfaces, literature concerning bimanual interaction provides the most relevant information. Ben Shneiderman has done extensive empirical research on interface design and the use of technology in accomplishing tasks (Shneiderman, 1980 & Shneiderman, 1987). He proposes that effective interactive platforms create feelings of success, competence and mastery while allowing the user to concentrate on their work, rather than on the technology itself. Features which offer informative feedback (e.g. status bars), reduce memory load (e.g. extending establish conventions) and allow actions to be easily reversed (e.g. undo and redo functions) are imperative according to Shneiderman. He suggests that allowing the user to configure elements within their own interface can accelerate their expertise, while conversely offering a preconfigured environment may contribute to a longer learning curve.

The introduction of emerging multi-touch interfaces will likely require the user to learn a new set of manual or bimanual tasks, beyond those from traditional

computer input devices such as the mouse and keyboard. Effective cognitive interpretation and retention of new information is important for the success of an emerging HCI device (Ceaparu et al., 2004). Existing HCI usability literature provides an important viewpoint regarding the strong relationship between the real and perceived effectiveness within interface design.

The concept of "chunking" (Miller, 1956), a learning mechanism for dividing information into retainable pieces, is applicable to interface design when considering the amount of new information presented to the user. It has been observed that people typically have the capability of processing seven, plus or minus two, cognitive tasks, which should be considered when designing a complex user interface. The cognitive limits may even decrease when trying to process information that is entirely new or unrelated (Sakai et al., 2003). Buxton believes this also holds true for motor learning (Buxton, 1995), which has a direct impact on interfaces such as multi-touch that require movement in predefined sets. For example, one of the advantages of a multi-touch interface is the ability to switch between different actions, such as panning or zooming, based on the number of fingers touching the surface. Too many allowable gestures, or motor learning chunks, may initially inhibit the user's ability to retain the available options.

Literature concerning metaphor within HCI suggests that semiotic theories can be applied to interfaces representing real-world equivalents (Condon, 1999). The act of signification provides the user with an established perception regarding how interfaces should respond based on the summation of their prior experiences. Panning a map by dragging a finger across a multi-touch display correlates well with pushing a paper map across the surface of a table. Assuming a general map reading knowledge, navigating digital maps should elicit similar techniques to those used for printed maps, regardless of the number of contact points.

There is currently a gap in multi-touch research concerning gesturing techniques and the navigation of geospatial data. Much of the previous research (Buxton, 2003 & Buxton, 2008 & Han, 2005) has focused on translating existing interface paradigms to a multi-touch environment instead of investigating the merits of specific applications (Moscovich, 2007). This lacuna is addressed by concentrating solely on multi-touch map navigation in the interest of contributing to the field of geography, which helps fill the research gap that currently exists.

Map Design and Cognition

Alan MacEachren (1995) has suggested that effective modern cartography is a blend of art and cognitive science. He suggests that computer graphics tools are necessary for advancement of the field, but the science behind map representation and visualization may be more important than ever. He states:

> At issue will be the processes by which each new level of representation is generated, the implications of these processes for map symbolization and design, and the corresponding implications of symbolization and design decisions for the success at which reasonable cognitive presentations are achieved.

MacEachren's views are relevant to multi-touch interfaces when considering that new techniques for map navigation may require a redesign of existing digital maps. Multi-touch interfaces require that the user's hands block a sizable portion of the viewing area, which is unlike using a mouse and pointer. The proper blending of art and science behind effective cartography, as MacEachren suggests, may be necessary to design an interface that displays the information blocked by the user's hands while maintaining the aesthetic integrity.

Maps have been used to provide both spatial inventories and thematic representations for thousands of years. Spatial cognition enables a map reader to interpret the representation of real-world locations with those on a map and,

therefore, navigate effectively. The long history of using maps as tools for navigation has helped shape cartographic principles, but the variety of display techniques greatly influences the map reader's interpretations (Lewis, 1972).

Interface Design

In order to explore gesturing techniques for map navigation, it was necessary for me to construct a working, tangible multi-touch interface. The testing platform includes a multi-touch interface, in the form of a table, and custom software for tracking interactions. The graphical user interface (GUI) allows user to apply a variety predefined gestures for map navigation and, therefore, experience the most commonly used methods for multi-touch interaction. For example, the user can utilize a predefined gesture for panning, such as two fingers brushing across the screen. The gesture can be performed with either hand at a variable pace.

Shneiderman stresses the importance of demonstrations when introducing new interfaces (Shneiderman, 2003). He found that the most effective type of automated demonstration involves animating graphical elements in a manner that closely resembles how users interact with the actual interface, such as showing windows dragging across the screen instead of jumping from one location to

another. Shneiderman's research indicates that subjects had a much higher success rate in performing interface tasks if the introductory education closely resembled the real-time user experience (Shneiderman, 2003), instead of just showing a series of still images or text based instructions (Carroll & Carrithers, 1984). Often, empirical experiments are encouraged to test the adequacy of interface design. For interactive and dynamic visualizations, empirical studies are suggested as a way to discover significant design improvements (Harrower and Fabrikant, 2008).

Shneiderman also emphasizes the advantages of offering interface components as sets of "layers" that can be used in combination with one another (2000). Allowing users to adjust the visibility of selected layers promotes a more manageable level of utilizable information, especially with regard to geovisualization (Plaisant, 2005). Menus, buttons and other data layers could be designed with visibility controls so the user can limit the display of pertinent information on a multi-touch interface. This would also ensure that map labels will properly ascend the visual hierarchy by not competing with overlapping layers.

Ontological considerations in the creation of interfaces for map navigation may have political ramifications (Dodge et al., 2009). The map itself is already a selective representation, and the interface allowing the visualization and navigation may be subject to cultural, social and economic influences as well. It may be difficult

to untangle the value judgments associated between a map and the interface used for its display. Also, a single interface template may not be appropriate for view all data sets (Edsall, 1999), since the nature of the data may require alternative methods of exploration.

Procedural analysis, as a subset of task analysis, provides a framework for investigating how users favor new forms of HCI. Differences between the idealized and actual results of a user experience can be identified and investigated using procedural analysis (Hackos & Redish, 1998). A user's comfort level is based on many factors and determining cognitive obstacles is helpful for the design process.

Allowing users to define their own rules for interface navigation is one form of participatory design. Studies such as the UTOPIA Project (Bødker et al., 1987) show that experience-based design methods, such as allowing employees to oversee the creation of their own software, yield a more effective workflow through increased cognitive recall. Some key elements of participatory design that shaped the creation of the usability testing platform include problem definition (Grønbæk, Kyng & Mogensen, 1995) and focusing many viable options into a few possible solutions.

Multi-touch Analysis

Previous user testing has been performed for common tasks to compare multi-touch systems with other forms of HCI. Reviewing the existing research is important in order to introduce effective methods of multi-touch interaction, even if the previous studies were not specific to map navigation. Comparing and contrasting multi-touch interfaces with other HCI methods can be accomplished by evaluating several variables including speed, user errors, memory retention, user apprehension and fatigue.

Some tasks can be performed much faster using multi-touch interfaces while other tasks may take considerably more time. Initially, gestures that are perceived as "intuitive" require less time to learn (Westerman & Elias, 2001). Although the learning curve may not necessarily have an impact on task completion times once a set of gestures is committed to memory, the introduction of new gestures requires consideration when comparing similar practices such as learning keyboard shortcuts. Research suggests (Davidson, 2006) that multi-touch gestures are more intuitive when based on mechanical processes in everyday life, such as rotating dials or pushing buttons. For example, introducing a new multi-touch technique for copy and paste (Wu et al., 2006), that was not perceived as intuitive, required more time to learn and twice as long to execute when compared to a keyboard. The time

required for the learning process, when combined with task completion times, may eliminate any advantages associated with a multi-touch interface.

Research concerning the required time to perform tasks using a multi-touch interface, such as scrolling, moving graphic elements, and data entry, reveals both advantages and disadvantages when comparing other forms of HCI. Scrolling times can be significantly reduced (Arthur, 2008) when using a multi-touch interface as opposed to using a mouse or tablet. Arthur found that continuous gestures, such as a spiral motion, could be used to scroll lengthy text blocks quicker than using traditional mouse-based processes. However, non-continuous gestures, such as dragging and lifting a finger repeatedly, contributed to greater task completion times than those afforded by a mouse or tablet. Arthur also found that moving graphic elements using a multi-touch interface, such as windows and images, took a comparable amount of time as using a mouse, but took less time than using a tablet. Research suggests that data entry with a multi-touch interface is considerably slower (Shanis & Hedge, 2003) than using a keyboard and mouse combination. Shanis and Hedge assert that built-in and established data entry tools, such as a dedicated ten key number pad, provide the fastest means for entering data by hand.

Errors from individual and collaborative groups using multi-touch interfaces have been studied. One advantage of multi-touch interfaces is the ability to

simultaneously accept input from multiple users. Errors may be reduced (Morris, 2006) when several people are collectively performing a task that requires a series a steps. Conversely, user errors may increase (Peltonen et al., 2008) when more than one person is using the same multi-touch interface to simultaneously complete non-related tasks. One disadvantage of allowing many users to concurrently use the same interface is that graphic elements may not be properly restricted to each user's allotted screen space. For example, if multiple users are organizing and resizing images in their own digital picture album, a user could scale an image large enough to cover the entire display, thereby overlapping the work area of other users.

The user's ability to retain new multi-touch gestures depends largely on the perceived relationship between the gesture and the performed task (Wu et al., 2006 & Westerman & Elias 2001). When users categorize gestures as obvious or intuitive, with respect to their associated tasks, retention is improved and recall times are reduced. However, some gestures may seem intuitive but should be avoided, such as applying more surface pressure with the fingers (Han, 2005) to simulate pushing away graphic elements. Han suggests that two different gestures from the same user, such as applying greater finger pressure and relaxing their hand position, may be indistinguishable for a multi-touch interface. Therefore, technical limitations may restrict the implementation of some gestures with potentially higher user retention.

First-time multi-touch users claim to be more comfortable navigating the interface if they have prior experience with other HCI methods such as using a keyboard (Detwiler et al., 2000). Multi-touch interfaces may produce less user fatigue (Wu et al., 2006) than other interfaces, especially when compared to tablets (Shanis & Hedge, 2003). Devices such as the keyboard and mouse require the user to adapt the shape of their hand, and finger position, to the physical dimensions of the apparatus. Gesture relaxation (Wu et. al, 2006) is possible with a multi-touch interface, potentially allowing the user to control their comfort level. However, multi-touch users may disapprove of the technology if technical limitations are obvious, such as a lag time between gesturing and recognition (Lee et al., 1985).

Chapter Four: Usability Testing Platform

Several factors made it necessary to construct a usability testing platform, rather than use existing commercially available multi-touch hardware and software. Scalability was an important consideration for exploring different map navigation methods, both in terms of the physical size and the implementation of additional interface elements. Microsoft Surface is approximately the same physical dimension as the testing platform that I built, but the $12,500 retail price was prohibitive for the purpose of this project. Smaller handheld devices, such as the iPhone and iPod Touch, were more financially viable but the physical size of the devices limits the scope of recognized gestures. Constructing a large format multi-touch device provides a cost effective means of exploring map navigation while allowing complex gestures to be recognized.

The limitations of multi-touch application programming interfaces (APIs) influenced the decision to build the testing platform. Multi-touch APIs, such as the Surface SDK, can decrease development times by providing solutions to common multi-touch tasks. However, API frameworks can limit development due to restrictions on system resources. Some multi-touch APIs are also built around specific hardware, which requires all development to be contained within a

predetermined environment such as the 3M Multi-touch Developer Kit (Fried, 2008). The multi-touch software that I created for the testing platform does not rely on external libraries or existing APIs which allows for future, independent development.

There were four major steps in the testing platform creation process. First, I built the physical structure of the interface which included the housing for all of the electronic components. The physical structure is both an input device for user interaction and an output device for displaying map tiles. Second, I created a custom blob detection algorithm for processing and tracking contact points on the top surface of the testing platform. Third, I designed the layout and process for visualizing the map tiles and GUI components. Existing software needed to be modified in order to create an interface that would accommodate multi-touch interaction. Fourth, I developed and incorporated new gesturing techniques to overcome existing multi-touch interface limitations.

Physical Structure

The usability testing platform that I constructed is rather large, with the top surface measuring roughly three feet horizontally and two feet vertically. The frame

is constructed of oak, providing enough weight and stability to prevent any movement that may occur as a user is touching the top surface. A projector is mounted perpendicular to the back edge of the top frame. A mirror on the floor reflects light from the projector back to the top surface of the table. A camera next to the mirror captures the FTIR activity from the above surface and sends the data to a connected computer for processing (Figure 4.9).

I accomplished the FTIR method by bouncing light from infrared light emitting diodes (LEDs) through two sheets of acrylic surrounding a sheet of drafting paper. The scattering of infrared light is not a result of the reflection from the drafting paper, but rather the result of light escaping from the top layer of the acrylic when an object, such as a human finger, touches the surface. The drafting paper allows the projected image to be seen as it passes through the bottom sheet of acrylic. Since the two light sources do not overlap on the light spectrum, it is easier to the distinguish changes in the infrared levels in order to determine individual points of contact.

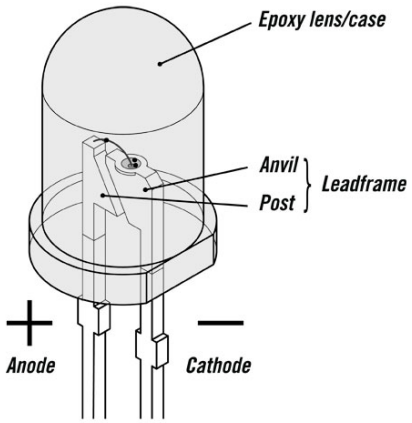

Figure 4.1: Compact infrared LED

A total of 88 infrared LEDs are used to provide enough coverage to encompass the entire surface of the top acrylic sheet. The LEDs are soldered together in four separate series so that the diagnostics associated with finding and replacing a defective diode becomes less of a burden (Figure 4.4). The LEDs are aligned flush with the outside edge of the acrylic so that the emitted light can bounce between the top and bottom surface of the ¼ inch sheet. I roughened the edges of the acrylic during construction, using a brass polishing agent, in order to scatter the infrared light non-uniformly. The bouncing light, as a result, is not restricted to the grid layout in which it was originally constructed because each roughened edge can spread the incoming light 180 degrees. The infrared LEDs create light with an electromagnetic radiation whose wavelength is around 900 nanometers, which lies outside the visible light spectrum for humans, typically between 400 and 700 nm. Therefore, no visual interference is created by the infrared LEDs.

I chose LEDs as an infrared light source for several reasons. First, energy consumption is relatively low for LEDs when compared to other forms of incandescent lighting. Each LED in the testing platform has a maximum rating of 1.2 volts of direct current (VDC), and each of the four series can be illuminated with 26.4VDC. Second, LEDs have a longer lifetime than other light sources. The LEDs used in the testing platform are rated for 30,000 hours of use, which far exceeds the

1,500 hour rating given to most incandescent bulbs. Last, the compact size of LEDs is a more viable solution for use in a compact design (Figure 4.1). At only 5mm in length, the LEDs require only a small channel surrounding the acrylic sheet.

The registration of diffused infrared light is accomplished optically with the usability testing platform. I made several camera modifications in order to acquire the desired type of light detection, data throughput and resulting frame rate. Since digital camera sensors have filters to block infrared light from contributing to unwanted visual noise, these filters are replaced with a layer of black cellulose acetate as a way to allow the passage of infrared light while simultaneously blocking all visible light (Figure 4.7). The components from two different inexpensive computer cameras are combined into one housing, sharing a single lens, providing the desired combination optical clarity and data granularity (Figure 4.8). Only the internal components utilizing the fastest available Universal Serial Bus (USB) connection are used (Figure 4.6). This allows the camera's optics to be easily adjusted while still taking advantage of the increased data rate that the USB 2.0 specification provides over the previous USB 1.1 specification, an increase from 12 megabits per second to 480 megabits per second. As a result, an additional 20 images can be received and processed from the camera every second.

Figure 4.2: Channels added for running wires

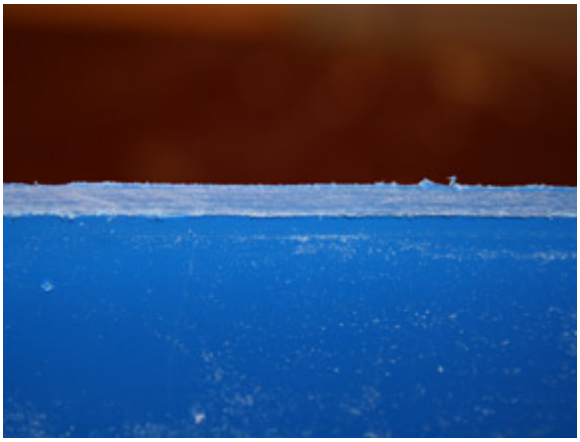

Figure 4.3: Acrylic surface roughened for light diffusion

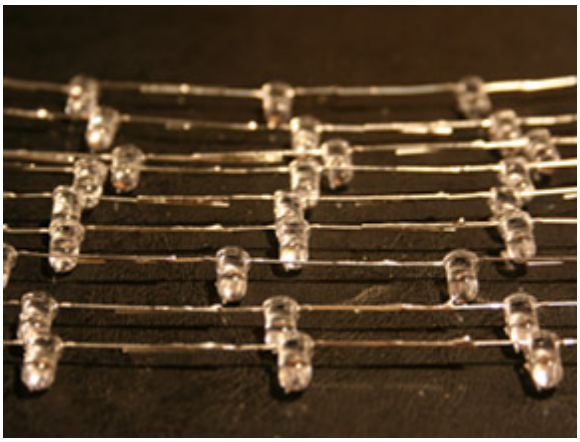

Figure 4.4: LEDs soldered in a series

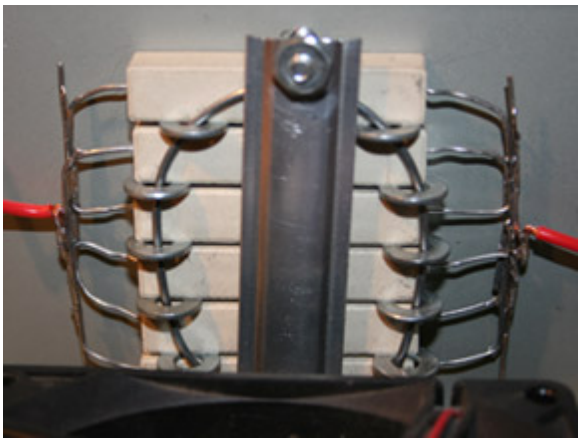

Figure 4.5: Series of ceramic capacitors

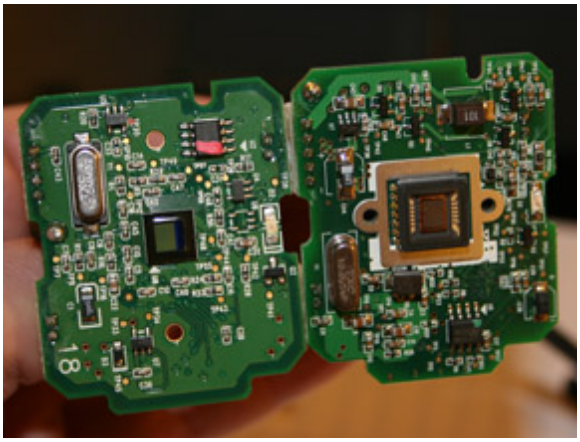

Figure 4.6: Modified camera PCBs for IR light detection

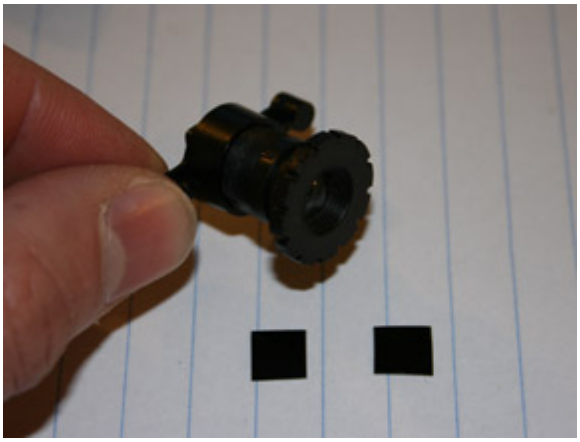

Figure 4.7: Camera lens with black cellulose acetate film

Figure 4.8: Housing for modified camera PCBs

Figure 4.9: Completed usability testing platform

I constructed the platform in the same manner as a typical piece of furniture, using reinforcements on the inside edges for stability. Additional consideration has been given for the electronic parts and wiring required for interactivity. Extra "channels" exist in frame for the wires extending from the power source, through

the resistors and eventually supplying the LEDs. Normally, a ventilation system is required for electronic projects containing a large number of electronic components, however, since the resistors exists outside of the wooden structure, very little heat is generated within the wooden channels.

All of the electronic parts that exist in the multi-touch table are powered by different levels of direct current (DC), rather than the constant voltage and amperage that alternating current (AC) from a typical wall outlet. A modified desktop computer power supply is mounted to the back of the table which provides enough current for the 88 LEDs. Six ceramic resistors, mounted within the transformer housing, restrict the current to an acceptable operability level for the LEDs (Figure 4.5). At 250 watts, the power supply is capable of destroying all of the electronic components of the usability testing platform if proper measures are not taken to restrict its output.

Blob Detection

A blob detection algorithm needs to meet a few minimum requirements in order for proper multi-touch registration to occur. First, the detection of multiple blobs at the same time is mandatory. Without this basic functionality, any multi-

touch environment would be reduced to a single touch system. Second, blobs need to be temporally linked in order to track changes in their relative position. Gestures such as pinching and de-pinching rely on variation in the position of contact points over time. Blobs also need to maintain their unique identifications so that relationships can be established. Third, a blob detection algorithm has to be efficient enough to process each frame captured by the camera at 30 frames per second. Ideally, a frame rate higher than 30 would be used, but there is a ceiling to the number of frames consumer webcams can produce per second. A noticeable display lag can occur if the processing rate drops below 20 frames per second.

I combined techniques from several methods to meet the minimum requirements while reducing the computational overhead. The following blob detection algorithms or libraries were considered or tested, but they either did not meet the minimum requirements or contained additional computational overhead without modification.

The Laplacian of Gaussian (LoG) method is one of the oldest and most widely used. It searches for blobs of a known size, which is adequate if the application has predefined limits for the areal coverage of a single blob (Lindeberg, 1998). However, the variability of a blob size matters when determining its center point. Determining the center point is important in order to know the user's intended target when

selecting a feature within a graphical interface. Graphic elements such as virtual keyboards, map markers and timeline sliders all require input based on determining a single point of contact. The LoG method also looks for both light and dark blobs which creates additional computational expense.

The maximally stable extremal regions (MSER) method finds differences between two regions and analyses subsets of the results. The MSER method is best suited to handle input from two different sources, such as a pair of cameras (Matas et al., 2002). This method can be modified to treat the input from one camera, at two different time points, as a set of images for comparison. However, if no difference is registered, perhaps because the user is holding a finger to the surface without moving, then the method would return a false negative. It is likely that Surface uses a technique similar to MSER since each unit uses two cameras for blob detection, although it is difficult to discern since the blob detection portion of Microsoft's Surface SDK is proprietary.

The grey-level blobs detection method evaluates contrast across several samples. Essentially, captured images are compared to a master image which is updated with new pixel information on a regular basis (Lindeberg, 1993). This method is ideal for detecting blobs in an environment where the ambient light levels may substantially change or when luminosity values for individual blobs cannot be

predetermined. Problems arise for a multi-touch environment, in a similar manner as MSER, when a point of contact remains static. The contrasting area defining a static blob will eventually be added to the master image which will then register a difference in contrast once the blob is removed. Over time, the master image will be recalibrated but a blob will be falsely detected in the interim. A modified portion of this method is used for the testing platform to help reduce camera noise and to control variations in light levels. A master image is used for comparison in the testing platform but the area that a registered blob covers is not used for updates.

The connected component analysis method is a two pass process whereby unique identifiers are determined on the first pass while adjacent neighbors are determined on the second pass (Shapiro and Stockman, 2002). Labels are then given to all connected features. Performing the blob detection in two passes reduces the frame processing rate by half, which is not ideal for a real-time system such as the testing platform. However, there are some aspects of the method that are appealing. First, the method uses either 8-way or 4-way connectivity to search for adjacent pixels with existing label assignments (Figure 4.10). Using a subset of the 8-way connectivity allows the two pass process to be reduced to a single pass via the pixels to the left, top-left and top (Figure 4.11). Individual blob labels can be maintained over time, making blob extraction less prone to error (Horn 1986). Second, the

method can be theoretically distributed across multiple threads using parallel processing. The scalability of connected component analysis makes it appealing for adding additional features such as object recognition and input from multiple users.

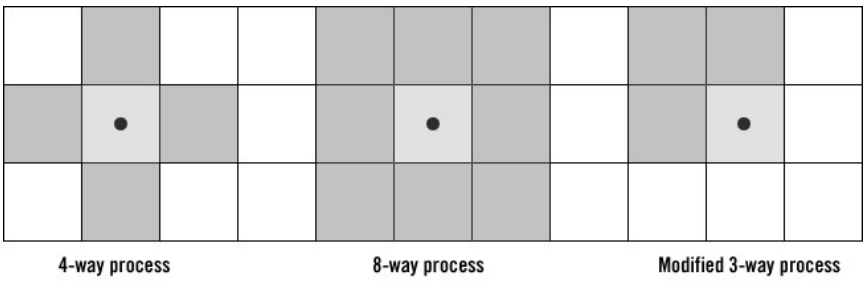

Figure 4.10 An updated version of the two-pass technique is reduced to a single pass by using a modified 3-way process

There are image processing libraries that contain classes for blob detection. OpenCV, an image processing library for C, C++ and Python, can find connected components within a single image. OpenCV finds blobs by performing an edge detection based on contrast and then assigns a unique identifier to the detected area. Edge detection is accomplished by using the 8-way process similar to connected component analysis. Edge and contour methods for blob detection generally require more processing than other techniques and are therefore not commonly used (Dillencourt et al., 1992). Another popular library for image analysis is vvvv which was originally built for real-time video synthesis (http://www.vvvv.org). Processing

images at a high frame rate is inherent to vvvv since its main data source is typically video, yet it remains a closed environment in terms of communication with external binaries. Since the final layer of map visualization is handled by Flash, a closed environment for blob detection would not allow a user to control the map.

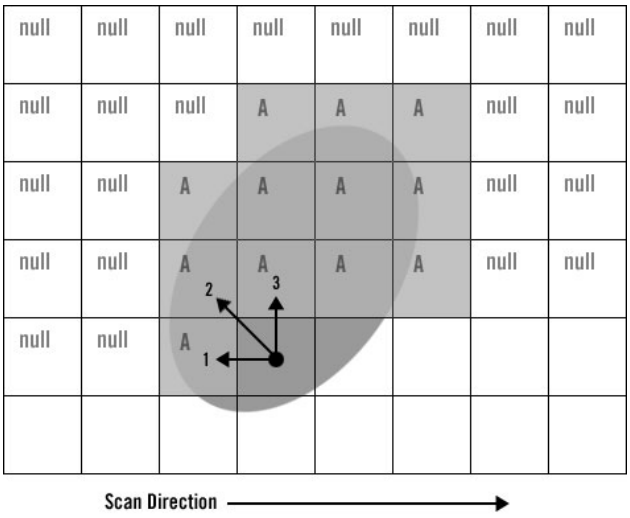

Figure 4.11: Pixels are assigned a label, such as A, when the designated area is occupied by any visible element. Adjacent pixels are compared for existing label assignments by using the modified 3-way process to the left (1), above and left (2), and above (3) of the current quadrant. The current pixel will inherit the label of any non-null, adjacent pixel.

The custom blob detection method I created for the testing platform borrows techniques from the grey-level blobs detection method and the connected component analysis method. A composite master image is used for comparison on each frame that is processed by the custom blob detection method, but updates to the master image only occur when no blobs are detected. The master image is composed of five frames and is updated by adding pixel information from one

frame every second. The software recalibrates every five seconds if no blobs are registered unless the areal coverage of the combined blobs is more than half the display size, signaling a major change in light levels. Sequential frames are used to create a new composite master frame if a change in light levels is presumed. Eight-way connectivity analysis, which requires two frames for comparison, is reduced to a 3-way search in order accommodate processing a single frame. The original image (Figure 4.12) and the processed image (Figure 4.13) differ in terms of bit depth, with the processed image only containing two colors representing the areas where blobs are either present or absent. The 3-way analysis could be modified in the future to evaluate every second pixel, for example, if the native resolution of peripheral cameras increases.

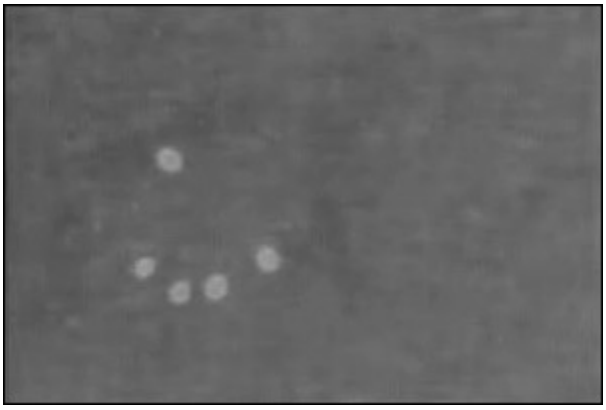

Figure 4.12: Original, noisy frame from camera

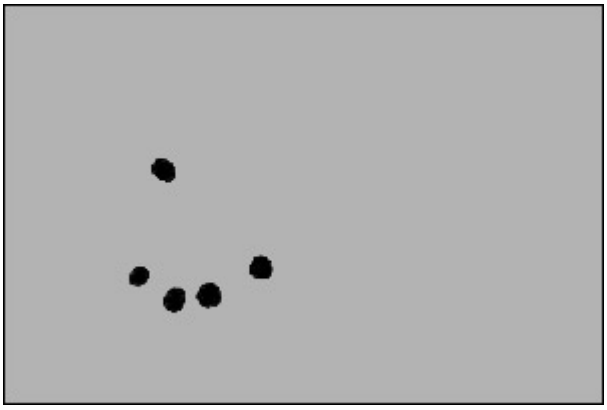

Figure 4.13: Processed frame after blob detection

Image Input and Map Visualization

I wrote custom software to process the data stream from the digital camera. The application, written in C#, evaluates the luminance values for each pixel in order to apply a blob detection algorithm that groups areas with similar, adjacent values. The data stream is accessed by the camera's drivers through an object dependency within the application. The exposed data are then converted to usable pixel values by Microsoft's DirectX application programming interface (API). Using the DirectX API allows the image preprocessing to utilize the computer's built-in graphics hardware, which allows the computation required for the blob detection to consume unshared processing power from the central processing unit (CPU).

The software is written in C# in order to take advantage of some key native language features. As part of the .NET platform, C# is both object-oriented and component-oriented. This allows external components, such as Flash, to be incorporated into the programming environment as a referenced object capable of two-way communication. Features such as automatic garbage collection allow C# applications to be created faster than using other languages such as C++. Several

integrated development environments (IDEs) exist for C# including the Visual Studio .NET IDE that was used to create the software for the testing platform.

The visual display is handled by embedding an Adobe Flash component that consumes the entire visible area of the application (Figure 4.14). The Flash API allows communication between Windows applications and a limited number of predefined Flash properties and variables. Programming Flash to recognize multiple points of input is accomplished using ActionScript, its native scripting language. The process of altering the native user input methods for Flash was a difficult process, but I managed to create a solution that produces very few errors. The base maps from popular mapping services such as Yahoo, Microsoft, OpenStreetMap and NASA Blue Marble, are displayed within the embedded Flash component using Modest Maps (http://www.modestmaps.com).

Modest Maps is a library for Flash, Python and JavaScript that is used for displaying map tiles. The Flash version of the library is used for the testing platform in order to display continuous base tiles from the Microsoft and Yahoo mapping services. A limited number of interactive features are built into the Modest Maps library, such as panning, zooming and resetting the map view. Modest Maps is not, however, an inherently multi-touch environment so several modifications have been made in order to register events from several contact points. The modified version

of the library is compiled within the SWF, which is then embedded into the stand-alone Window's application. The expected mouse input for Modest Maps is replaced by a data stream from C# that includes the placement and relationship of blobs, or contact points, generated by the user.

The decision against using either the Flash or static versions of the Google Maps API was mainly based on two factors. First, relying solely on a single mapping service could prove disastrous if the company proving the service decides to limit its public availability or drastically modify the API. Updating the testing platform to work with another mapping service could be very time consuming especially if the new API is not as robust as the original. Second, the static version of the Google Maps API requires a web browser, or browser component, for visualization. Incorporating a web browser into the visualization pipeline would add unnecessary overhead to the testing platform.

I created a custom data format in order for communication to occur between C# and the embedded Flash file containing the Modest Maps API. Instead of using the built-in callback methods in Flash, which can create bottlenecks, a string variable assignment is formatted to resemble XML. The string is then parsed using Flash's included XML handling methods so that the amount of data can vary per frame, such as the total number of blobs and relationship information. Frames can be

dropped if there is a delay in processing, unlike using the built-in callback methods in Flash which get stacked and processed sequentially.

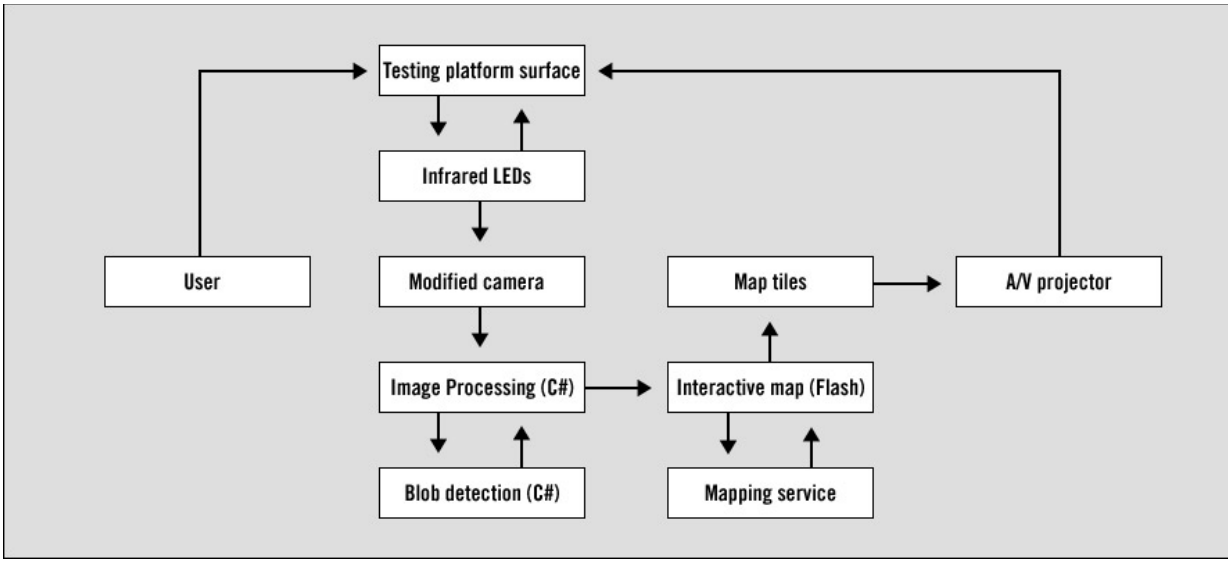

Figure 4.14: Flowchart showing interaction and visualization for the usability testing platform

Custom Gesturing Techniques

Many of the gesturing techniques that I programmed and implemented into the testing platform resemble gestures used in other multi-touch interfaces. However, I created custom gesturing techniques that either extended or replaced more common gestures in order to enhance map navigation. I developed

enhancements for some of the most common map navigation tasks including zooming, panning and accessing menus.

Gestures for zooming include pinch and de-pinch using two fingers on the same hand or one finger from each hand. Bringing the fingers closer together will expand the current map extent, versus spreading the fingers apart which will zoom in. In an effort to increase precision, I based the spread distance required to perform a zoom on the initial location of the fingers, so that the varying width between two contact points is divided by the initial width, creating a zoom ratio that can be generated using one or two hands. The user can perform a more precise zoom by using one finger on each hand near the outside, opposite edges of the top surface. When zooming with only one hand, the user's other hand can pan the map by dragging a single finger. Bimanual operations such as simultaneous pan and zoom may potentially decrease map navigation times.

One method of panning a map on the testing platform is accomplished in a similar manner as using an iPhone or Surface, by dragging a single finger across the interface. New map content is revealed by essentially pulling base tiles into view. This logical gesture has been shown to be an effective dragging-and-dropping technique (Wu et al., 2006), but improvements were made through the testing platform in order to create a gesture that is better suited for map navigation. When

using the iPhone, Surface or similar devices, there is a one-to-one relationship between the pan distance and the movement of the user's finger. This relationship is not entirely necessary when comparing more common forms of HCI, such as the mouse, in which smaller movements from the user are translated into larger movements on the display. I increased the magnitude of the panning distance on the testing platform to 1.5 times the distance traveled by a user's single finger. Dragging two fingers produces twice the panning distance while using three fingers increases the distance by a factor of three. Implementing these enhancements can potentially reduce the scale of a user's movements and reduce panning times.

Accessing and selecting menus is accomplished through graphic buttons or drop down menus in many applications. Allocating permanent screen real estate may be unnecessary using a multi-touch interface due to the high number of possible individual and combined gestures. There are a number of custom menu options that I programmed for the usability testing platform, including a quick selection method and gestures for navigation features that would normally require the user to select a button. All of the menus in the testing platform are hidden until the user performs particular gestures. For example, the menu for selecting base tiles is shown by pressing four fingers on the left hand against the surface of the testing platform while pressing up to five fingers on the right hand. The number of fingers

pressed on the right hand corresponds with the selected menu item. In the previous example, selecting the second set of available base tiles is accomplished by pressing four fingers on the left hand along with two fingers on the right hand.

Some map navigation tasks may not need to be grouped in the same menu with other features, such as resetting the map extent. For a quick map reset, the user can press and hold five fingers from either hand against the top of the testing platform. Future versions of this technique could include the ability for the user to twist their pressed fingers, as if turning a dial, to select additional menu items.

DIY Movement

The inspiration for many of the technical portions of this project came from the do-it-yourself (DIY) movement. The social aspects of Web 2.0 provide a platform for DIY enthusiasts to share their experiences. The recent rise DIY movement has been aided by several factors. The propagation of broadband installations, availability of DIY publications, increase in web publishing tools, decreased cost of electronic components and new methods of media exposure have all contributed to an increase in the number of individuals working on projects themselves, rather than purchasing finished products.

Reliable high speed internet has changed the way in which its subscribers are able to view, create and share information. Popular sites such as instructables.com and youtube.com allow their users to upload text, images and video that is maintained in an easily accessible and searchable manner. Blogs and other open-source content management systems (CMS) are making it easier for anyone with a reliable internet connection to post and update information without web development experience.

DIY publications such as Make magazine and ReadyMade provide detailed instructions for projects ranging from robots to electric bicycles. Many of the projects in these publications deal with the modification of existing consumer electronics to perform alternative tasks, such as modifying webcams to view a limited portion of the light spectrum. In addition to detailed instructions, Make magazine also includes a parts list for each project along with information on suppliers.

Prices for electronic parts continue to fall as consumers are increasingly buying parts directly from the manufacturers. The infrared LEDs used in the testing platform were purchased directly from a manufacturer in China for half the cost of buying them from an electronics reseller. Historically, only wholesale orders were

allowed from large manufacturers, but changes to ordering limitations are occurring to meet the growing demand from individual consumers.

The total cost of the testing platform was approximately $250.00, including the building materials, electronic components, power transformer and USB cameras. Created a larger testing platform would require higher costs for the building materials, but the other equipment costs would remain largely unchanged. The software for the testing platform was creating using a free version of the Microsoft Visual C# IDE, with all programming reference materials available online. The projector was the most expensive component required for accomplishing the FTIR method, and its cost was not included in the total because one was already available. However, the price of standard definition projectors continues to drop with entry level models starting below $300.00. A safe assumption for building a multi-touch platform similar to the one that I've created, with a surface area of roughly six square feet, would be $600.00 for all of the components including the projector.

Chapter Five: Discoveries and Future Direction

Findings

The creation and operation of the usability testing platform led to several findings. Technical difficulties and design limitations were discovered throughout the process. First, the FTIR method is a low cost solution for multi-touch interaction in terms of computational expense and financial consideration, but controlling the release of infrared light is challenging. Second, the lack of a "hover" state eliminates one form of interaction that has become commonplace among mapping applications. Third, common navigation techniques such as panning require additional consideration in terms of cartographic design and physical limitations when exposing the interface to a general audience.

Infrared light can escape through the top surface and reveal objects, such as the palm of the user's hand, that are close but not actually touching (Figure 5.1). The luminosity threshold that is present in the blob detection application can be adjusted, but non-touching objects may still be detected. Using two cameras, separated by at least a foot, could be used to reduce the number of false positives by comparing the offset of blobs for each frame. The two camera approach could also, theoretically, register the objects above the surface for gesture recognition without

actually touching the testing platform. The end result would be an interface that could accept single and multi-touch inputs as well as gesture recognition used independently or as a combination of events.

A hover state, similar to a mouse-over, does not exists using the FTIR method (Figure 5.2). Traditional interface features such as pop-up windows and buttons with mouse-over states do not exist in an environment where the interaction is limited to on or off states. A viable replacement is a time-based holding technique where interface features are triggered when the user holds one or more fingers at the same location for predetermined amount of time. This would require that all selection events are based on the point at which a blob no longer exists at the same location, instead of when the blob first appears. The equivalent interaction using a mouse would be the difference between mouse-down events and mouse-up events. Multi-touch events based on blob removal are intuitive as long as the interface events remain consistent.

Figure 5.1: Infrared light escaping through top surface

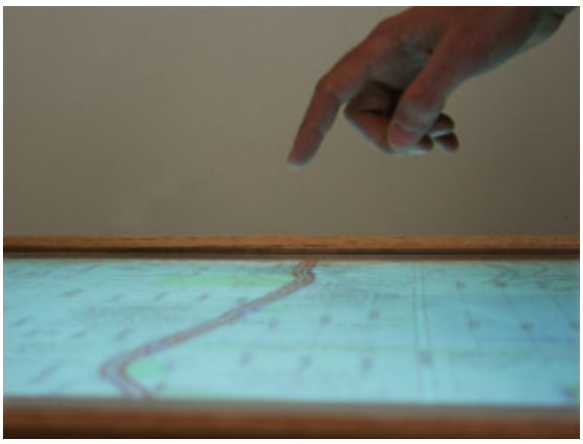

Figure 5.2: No "hover" state is registered from above

Panning a map by dragging one or more fingers across the surface of the platform is intuitive but there are some inherent problems. First, the area under the user's hand is covered for the duration of the pan (Figures 5.3 & 5.4). Unlike clicking and dragging with the mouse, the user's hand can potentially hide important map features during navigation. The spatial extent of the coverage can be quite large depending on the scale of the map. Second, humidity or other factors contributing to a damp surface can create resistance when dragging fingers across a large display such as the testing platform. Moisture can also interfere with blob detection by allowing the infrared light to escape through the top of the acrylic, whether from dampness in the air or oils on the skin. Lastly, long fingernails make it difficult maintain an angle for the fingers to apply enough pressure to the top

surface. The acrylic surface can also become scratched if fingernails or jewelry are dragged with excessive force. Glass was a consideration for the top surface, which would be more scratch resistant, but the glass I tested released an unacceptable amount of infrared light through the top surface. Also, applying a thin sheet of glass over the existing acrylic layers would not allow the infrared light to disperse properly when touched because the contact points would blend together and become less defined.

Figure 5.3: Map features are covered by the user's hand

Figure 5.4: Features remain hidden while panning

Light is released through the top surface of the testing platform using the FTIR method. Adequate pressure against the acrylic from the user's fingers is necessary because the detection algorithm requires a predefined minimum size for each blob. The amount of pressure to apply is not necessary intuitive, especially if the user has no previous experience with a multi-touch surface. Inadequate pressure creates a very small footprint and may not be detected as a blob since visible noise from the camera is ignored by the blob detection algorithm. Familiarity with capacitive multi-touch devices, such as the iPhone or iPod Touch, may add to the confusion because very little pressure is required in order to register blobs.

Menus are an integral part of many interactive mapping platforms, but they also require valuable screen real estate (Figure 5.5). Individual buttons are generally spread apart more than non-touch interfaces to accommodate the areal coverage of fingers, rather than the single pinpoint provided by a mouse pointer. By allowing the user to use two hands, menu systems for the testing platform are hidden until five

Figure 5.5: Menu system covering much of the map

fingers from one hand are pressed and held against the top surface for two seconds. Interaction with the other hand can still occur by allowing the user to simultaneously navigate the map and display a desired menu. The press and hold technique could be expanded by assigning different menus and items to each hand. For example, using three fingers on the left hand could select the third menu while using four fingers on the right hand could select the fourth item from that menu. A standard touch and release menu system should be used in conjunction with a more complex process since the user would be required to know the order and layout of each menu for a two-handed quick navigation method.

Features	Apple iPhone	Microsoft Surface	Usability Testing Platform
Zooming with pinch and de-pinch	X	X	X
Panning with touch and drag	X	X	X
Portable	X		
Ability to personalize multiple gestures			X
Suitable for multi-user collaborative environments		X	X
Select menu options with a single tap	X	X	X
Accelerated panning with one, two or three fingers			X
Quick menu selections using two-handed finger combinations			X
Under $250 (without licensing or service contract)			X

Figure 5.6: Comparison of multi-touch platforms

The Market

Projected sales of iPhone and iTouch are estimated at roughly seven million units in the fourth quarter of 2009, while Surface has been initially released in test markets such as San Francisco, New York City, Seattle and San Antonio. Map navigation is possible using these portable and large format multi-touch interfaces but little is known about the way in which users would prefer to use these interfaces for specific navigational tasks. As multi-touch technology continues to propagate to a larger consumer audience, the gap between what is known about users' preferences and the techniques that are currently available continues to widen, with the exception of internal user testing from companies such as Apple and Microsoft. Both Apple and Microsoft have designed multi-touch interfaces designed to work with a variety of applications, but effective map navigation may prove to be one of the most critical.

Beyond the iPhone and iTouch, the market for touch screen devices is expected to see major growth within the next few years, with an estimated 21 million units sold in 2012 (Wong, 2008). There will also be an increase in the number of GPS enabled devices as chipsets become smaller and are designed to consume less

power. Google continues to expand its support for GPS enabled multi-touch devices by increasing the number of platforms supported by Google Maps, including the iPhone, Android, Windows Mobile, Java, Palm's webOS and several others. Map navigation using handheld devices will likely gain popularity as hardware and software products continue to evolve.

Microsoft Surface is commercially available to select AT&T retailers for in-store promotions, but a consumer version will not be available until 2010. One potential barrier to Surface's early widespread commercial success could be its current price of $12,500.00 per unit (Microsoft, 2009). Prices are expected to fall once more units have shipped in 2010. Microsoft claims to currently have 120 partners developing Surface applications for commercial release (Kirk, 2009).

Two major operating systems and one development platform currently support multi-touch for the desktop and portable markets. Mac OS X Snow Leopard (version 10.6) supports multi-touch as an interface method for software such as Safari and iLife. Microsoft Windows 7 has multi-touch application support for software such as Internet Explorer 8 while allowing further development to occur though the Surface SDK. The MT4j development platform for Java uses OpenGL to provide accelerated multi-touch application support for the Windows and Linux operating systems.

One potential area for growth in multi-touch development is the removal of any required surfaces for touch registration. Systems that analyze gestures within the same plane may be used to eliminate the need for a predefined multi-touch surface. The "Sixth Sense" project from the Fluid Interfaces Group at MIT's Media Lab projects the viewing area onto any surface visible to the user (Figure 5.6). The interaction is then handled by the same device used to project the interface and map navigation is accomplished in a similar manner as existing multi-touch devices (Figure 5.7), using intuitive hand movements (Mistry, 2009). The projected surface becomes scalable in not only size, but capability because external objects, such as fiducial markers, can be recognized by the system in addition to hands and individual fingers.

Figure 5.7: Touch detection on projected surface

Figure 5.8: Map navigation through gestures

Many of the multi-touch applications and demonstrations that are currently available concern graphics-based approaches to common interface problems, such as map navigation and viewing image galleries. However, a growing area of multi-touch development is the e-Book reader, or e-Reader, market. Hardware manufacturers such as Sony, Amazon and Bookeen are all incorporating navigation features based on multi-touch interaction. One advantage of multi-touch interfaces is that several points and gestures can be combined to extend the utility of the original interaction. For example, a recent eBook concept from Nedzad Mujcinovic at Monash University (Standards Australia, 2008) included a multi-touch interface that allows the user to advance pages by a swiping or flicking a single finger. Using multiple fingers increased the functionality, with double and triple finger gestures increasing the pages by 10 and 50 respectively. Applying the same logic, panning a map using a single finger on the testing platform will give you a one to one ratio in the distance traveled on the screen to the respective pan distance. Using two fingers could increase the pan distance by a factor of two, using three fingers by a factor of three, and so on.

Dell released the first commercially available multi-touch tablet, the Latitude XT2, on April 9th 2009. Early adopters of the integrated multi-touch display reported

problems with the input drivers not working properly (Murph, 2008). Early technical problems have limited the release of new products in the past, such as the Xbox 360, and can take years for a full recovery. Fortunately, other hardware manufacturers are releasing multi-touch enabled laptops, tablets and netbooks despite Dell's early technical issues and negative publicity.

Adobe's Flash Player 10.1 is scheduled for general release by the end of 2009 and it will contain native support for multi-touch interaction in a similar manner as mouse events. A new TouchEvent library will allow developers to track events such as pinching, de-pinching and rotation without the need to develop custom methods for allowing input from sources beyond the mouse and keyboard (Ricker, 2009). Providing a framework to handle basic multi-touch interaction will allow Flash developers to create additional event handlers for more complex gestures. Native support for multi-touch will also enable developers to easily add another form of interaction to computers and devices that have the proper hardware, while still allowing the mouse and keyboard to be used in a traditional manner.

Chapter Six: Conclusion

A scalable and affordable multi-touch testing platform for testing map navigation was successfully created. The FTIR method was incorporated into a low cost structure which enabled an interface for map navigation to be constructed for research and exploration. Proprietary multi-touch APIs were avoided by creating custom software in order to avoid the inherent limitations that exist with commercial applications. The testing platform can be scaled, both in terms of physical dimensions and additional functionality, to discover other areas of multi-touch research in the future. The process of creating a testing platform, along with an interface for map navigation, provides valuable insight into many aspects of multi-touch development. Technical and aesthetic challenges are inherent to any multi-touch system and the testing platform allows these obstacles to be explored in an affordable and accessible manner.

Effective forms of HCI provide informative feedback while reducing memory load (Shneiderman, 1987). The testing platform includes responsive visual cues along with intuitive multi-touch gestures in an effort to provide an interface with minimal cognitive barriers. Limiting the number of gestures may increase retention

for the user while allowing the multi-touch system to reduce the computation necessary to evaluate a large number of acceptable gestures.

Reviewing the different techniques for multi-touch registration, such as capacitive approaches and FTIR, reveal the strengths and weaknesses of each process. Capacitive approaches are better suited for smaller devices such as laptop computers and mobile phones because of the lower power consumption and smaller footprint. The number of capacitive multi-touch devices is expected to increase rapidly over the next few years (Wong, 2008). FTIR methods work well for large format displays, such as Microsoft Surface, due to its inexpensive and flexible characteristics. The FTIR approach can also be accomplished without the need for custom electronic parts.

One of the most important features of the testing platform is the blob detection algorithm. The process of extracting blobs at a rate of 30 frames per second requires an efficient labeling routine. Properly assigning labels to individual blobs is accomplished by comparing adjacent pixels using a 3-way process. The testing platform uses aspects from several different blob detection techniques for calibration and establishing temporal relationships.

Discoveries concerning the construction and operation of the testing platform include limiting the release of infrared light, the restrictive nature of binary touch events, determining the minimum level of pressure from the user's fingertips and design considerations for the obstructed views created by the user's fingers and hands. A two camera system would be better suited to handle the infrared light that reflects off objects that are close to, but not touching, the top surface of the testing platform. Comparing the images from two cameras can determine the distance to objects based on the offset between images. The lack of a hover state could also be addressed using a two camera system, which would enable the implementation of common interface events such as those based on a mouse-over state. The contact pressure required from a user's fingertips can vary between multi-touch systems and the FTIR method requires a larger footprint than capacity systems. Users of FTIR systems may require more time to familiarize themselves with the acceptable contact pressure limits. When designing a multi-touch interface, consideration should be given to the obstructed views created by the user's fingers and hands. Important information and navigation elements should not reside in areas where views may be potentially hindered.

The development of large and small scale multi-touch devices will see substantial growth in the next few years (Wong, 2008). Innovations in GPS receivers

and online mapping services will also drive new forms of map navigation. An increase in integrated multi-touch systems creates a demand for intuitive user interfaces. Therefore, continued testing of map navigation using multi-touch systems will be necessary for creating innovative solutions to existing and future interface challenges.

Bibliography

Arthur, Kevin, Matic, Nada & Ausbeck, Paul (2008). *Evaluating Touch Gestures for Scrolling on Notebook Computers*. CHI 2008, April 5–10, 2008.

Bertin, Jacques (1967). *Sémiologie Graphique. Translated by the University of Wisconsin Press, 1983.*

Board, Christopher (1978). *Map Reading Tasks Appropriate in Experimental Studies in Cartographic Communication*. The Canadian Cartographer, 78, 1-12.

Bødker, S. Ehn, P., Kammersgaard, J., Kyng, M., & Y. Sundblad (1987). *A Utopian Experience*. In Bjerknes, G., Ehn, P. & Kyng, M., (eds.). Computers and Democracy - a Scandinavian Challenge, 251-278.

Buxton, W., Hill, R. & Rowley, P. (1985). *Issues and Techniques in Touch-Sensitive Tablet Input*. Computer Systems Research Institute, University of Toronto.

Buxton, William (1995). *Chunking and Phrasing and the Design of Human –Computer Dialogues*. Human-Computer Interaction: Toward the Year 2000

Buxton, Bill. (2008). *Multi-Touch Systems that I Have Known and Loved*. Retrieved from http://www.billbuxton.com/multitouchOverview.html

Carroll, J. M. and Carrithers, C., (1984) *Training Wheels in a User Interface*, Comm. ACM 27, p. 800-806.

Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., and Shneiderman, B. (2004). *Determining causes and severity of end-user frustration.* International Journal of Human-Computer Interaction, 17(3), p. 333- 356.

Condon, Christopher (1999). *Metaphor in the Human-Computer Interface*. Brunel University PhD Dissertation

Davidson, Philip, Han, Jefferson (2006). *Synthesis and Control on Large Scale Multi-Touch Sensing Displays*. Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06), Paris, France.

Dillencourt, Michael B., Samet, Hannan and Tamminen , Markku (1992). *A general approach to connected-component labeling for arbitrary image representations.* ACM

Dodge, Martin, Kitchin, Rob, Perkins, Chris (2009). *Rethinking Maps.* Routledge

Elrod, S., Bruce, R, Gold, R., Goldberg, D. Halasz, F., Jannsen, WI, Lee D., McCall, K., Pedersen, E., Pier, K, Tang, J. & Welch, B. (1992). *Liveboard: A Large Interactive Display Supporting Group Meetings, Presentations, and Remote Collaboration*. Conference on Human Factors in Computing Systems, 1992, p. 599 – 607

Fried, Ina (2008). *Microsoft's Surface pricier than anticipated.* Retrieved from: http://news.cnet.com/8301-13860_3-10073929-56.html

Grønbæk, K., Kyng, M. & P. Mogensen (1993). *Cooperative Design in Engineering Projects*. Communications of the ACM, 36, 6, 67-77

Hackos, J.T. & Redish, J.C. (1998) *User and Task Analysis for Interface Design*. John Wiley & Sons, Inc.

Han, Jefferson (2005). *Multi-touch Sensing Through Frustrated Total Internal Reflection*. ACM Special Interest Group on Computer Graphics and Interactive Techniques.

Harrower, Mark & Fabrikant, Sara (2008). *Geographic Visualization: Concepts, Tools and Applications*. John Wiley & Sons, Ltd., p. 49-66.

Horn, Berthold Klaus Paul (1986). *Robot Vision*. MIT Press, 69-71.

Jenks, George A. (1973). *Visual Integrity in Thematic Mapping: Fact or Fiction*? International Yearbook of Cartography, 13, 27-35.

Kirk, Jeremy (2009). *Microsoft Expands Surface to Europe, Middle East.* Retrieved from http://www.pcworld.com/article/160473/article.html?tk=nl_dnxnws

Lee, SK., Buxton, W., Smith, K.C. (1985). *A Multi-touch Three Dimensional Touch-sensetive Tablet*. CHI 1985 Proceedings.

Lewis, D. (1972). *We the Navigators* . Australian National University Press.

Lindeberg, Tony (1993). *Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A Method for Focus-of-Attention.* International Journal of Computer Vision 11 (3), 283-318.

Lindeberg, Tony (1998). *Feature detection with automatic scale selection.* International Journal of Computer Vision 30 (2): 77-116.

MacEachren, A. M., 1995, *How Maps Work: Representation, Visualization, and Design*. Guilford Press.

Matas, J., Chum, O., Urba, M., and Pajdla, T. (2002). *Robust wide baseline stereo from maximally stable extremal regions.* Proc. of British Machine Vision Conference, 384-396.

Mehta, Nimish (1982), *A Flexible Machine Interface,* M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto supervised by Professor K.C. Smith

Miller, George A. (1956). *The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information*. First published in Psychological Review, 63, 81-97.

Microsoft (2009). *Purchasing Microsoft Surface.* Retrieved from Surface product site at http://www.microsoft.com/surface/Pages/HowToBuy/HowToBuy.aspx

Mistry, Pranav (2009). *s i x t h s e n s e - a wearable gestural interface (MIT Media Lab).* Retrieved from http://www.pranavmistry.com/projects/sixthsense

Morris, Meredith Ringel, Huang, Anqi, Paepcke, Andreas & Winograd, Terry (2006). *Cooperative Gestures: Multi-User Gestural Interactions for Co-located Groupware*. CHI 2006, April 22–28, 2006, Montréal, Québec, Canada

Moscovich, Tomer (2007). *Principles and Applications of Multi-touch Interaction*. Brown University PhD Dissertation.

Murph, Darren (2008). *Multi-touch display giving Dell Latitude XT users fits?* Retrieved from http://www.engadget.com/2008/10/20/multi-touch-giving-dell-latitude-xt-users-fits/

Nakatani, L. H. & Rohrlich, John A. (1983). *Soft Machines: A Philosophy of User-Computer Interface Design*. Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'83), 12-15.

Peltonen, Peter, Kurvinen, Esko, Salovaara, Antti, Jacucci, Giulio, Ilmonen, Tommi,

Evans, John, Oulasvirta, Antti, & Saarikko, Petri (2008). *"It's Mine, Don't Touch!": Interactions at a Large Multi-Touch Display in a City Centre*. CHI 2008, April 5–10, 2008, Florence, Italy.

Ricker, Thomas (2009). *Adobe Flash Player 10.1 and AIR 2 betas are out, multi-touch and video acceleration are in.* Retrieved from http://www.engadget.com/2009/11/17/adobe-flash-player-10-1-beta-is-out-multi-touch-and-video-accel/

Sakai, K., Kitaguchi, K., and Hikosaka, O. (2003). *Chunking during human visuomotor sequence learning.* Experimental Brain Research, 152:229-242.

Shanis, Jenna & Hedge, Alan (2003). *An Exploration Of Multitouch Technology: A Comparison Of A Split Angle Keyboard With Multitouch And Current Input Technologies*. Proceedings of the Human Factors and Ergonomics Society 47th Annual Meeting, Oct. 13-17, Denver, CO, p. 746-750.

Shapiro, L., and Stockman, G. (2002). *Computer Vision*. Prentice Hall. pp. 69–73.

Shneiderman, Ben (1987). *Designing the User Interface. Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing.

Shneiderman, Ben (2003). *Promoting universal usability with multi-layer interface design.* ACM Conference on Universal Usability

Standards Australia (2008). *LIVRE - Australian Design Award - James Dyson Award.* Retrieved from http://student.designawards.com.au/application_detail.jsp?status=2&applicationID=3503

Westerman, Wayne (1999). *Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-Touch Surface*. University of Delaware PhD Dissertation.

Westerman, Wayne & Elias, John (2001). *Multi-Touch: A New Tactile 2-D Gesture Interface for Human-Computer Interaction*. Human Factors and Ergonomics Society 45th Annual Meeting October 8-12, 2001.

Wong, May (2008). *Touch-screen phones poised for growth.* USA Today

Wu, Mike, Shen, Chia, Ryall, Kathy, Forlines, Clifton, Balakrishnan, Ravin (2006). *Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces*. IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2006.

Zemen, Eric (2009). *Who's The Champ? Apple, Microsoft, Nokia Report Smartphone Sales.* Retrieved from http://www.informationweek.com/blog/main/archives/2009/01/whos_the_champ.html;jsessionid=ROPTAOD3VOPGTQE1GHOSKHWATMY32JVN