Multi-touch For General-purpose Computing An Examination Of Text Entry

Paul David Varcholik
University of Central Florida

MULTI-TOUCH FOR GENERAL-PURPOSE COMPUTING: AN EXAMINATION OF TEXT ENTRY

by

PAUL DAVID VARCHOLIK
B.S. Valdosta State University, 1998
M.S. University of Central Florida, 2008

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Modeling & Simulation
in the College of Engineering & Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2011

Major Professors: Charles E. Hughes
Joseph J. LaViola

ABSTRACT

In recent years, multi-touch has been heralded as a revolution in human-computer interaction. Multi-touch provides features such as gestural interaction, tangible interfaces, pen-based computing, and interface customization – features embraced by an increasingly tech-savvy public. However, multi-touch platforms have not been adopted as "everyday" computer interaction devices; that is, multi-touch has not been applied to general-purpose computing.

The questions this thesis seeks to address are: Will the general public adopt these systems as their chief interaction paradigm? Can multi-touch provide such a compelling platform that it displaces the desktop mouse and keyboard? Is multi-touch truly the next revolution in human-computer interaction?

As a first step toward answering these questions, we observe that general-purpose computing relies on text input, and ask: "Can multi-touch, without a text entry peripheral, provide a platform for efficient text entry? And, by extension, is such a platform viable for general-purpose computing?"

We investigate these questions through four user studies that collected objective and subjective data for text entry and word processing tasks. The first of these studies establishes a benchmark for text entry performance on a multi-touch platform, across a variety of input modes. The second study attempts to improve this performance by examining an alternate input technique. The third and fourth studies include mouse-style interaction for formatting rich-text on a multi-touch platform, in the context of a word processing task.

These studies establish a foundation for future efforts in general-purpose computing on a multi-touch platform. Furthermore, this work details deficiencies in tactile feedback with modern multi-touch platforms, and describes an exploration of audible feedback. Finally, the thesis conveys a vision for a general-purpose multi-touch platform, its design and rationale.

ACKNOWLEDGMENTS

I wish to acknowledge the assistance of a variety of groups and individuals, without whose support, this work would not have been possible. I am sincerely grateful to my family, friends, and co-workers at the University of Central Florida (UCF), the Institute for Simulation & Training (IST), the Florida Interactive Entertainment Academy (FIEA), the Media Convergence Lab (MCL), the Interactive Systems and User Experiences Lab (ISUE) and the Applied Cognitive Training & Immersive Virtual Environments Lab (ACTIVE). I also wish to thank the esteemed members of my dissertation committee: Charles Hughes, Joseph LaViola, Stacey Scott, Brian Goldiez, and Michael Moshell.

These efforts have been funded, in part, by: IARPA, under US Air Force Research Lab award FA87500820202; SAIC, under Science Applications International Corporation award 4400157271; and by the National Science Foundation under award number DRL0638977

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYNMS/ABBREVIATIONS

3DUI	3-Dimensional User Interaction
ANN	Artificial Neural Network
API	Application Programming Interface
ATM	Automated Teller Machine
BPP	Bits Per Pixel
CG&A	IEEE Computer Graphics and Applications
CLI	Common Language Infrastructure
COTS	Commercial-Off-The-Shelf
CPS	Characters Per Second
CPW	Characters Per Word
CHI	Conference on Human Factors in Computing Systems
DI	Diffused Illumination
DSI	Diffused Surface Illumination
FSR	Force Sensitive Resistor
FTIR	Frustrated Total Internal Reflection
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HCI	Human-Computer Interaction

IR	Infrared
IEEE	Institute for Electrical and Electronics Engineers
IFSR	Interpolating Force-Sensitive Resistance
KSPC	Key Strokes Per Character
LED	Light-Emitting Diode
LLP	Laser Light Plane
MDF	Medium-Density Fiberboard
MSD	Minimum String Distance
NEAT	Neuro-Evolution of Augmenting Topologies
OO	Optical Occlusion
OSC	Open Sound Control
OLTP	Online Transaction Processing
PDA	Personal Digital Assistant
PVC	Polyvinyl Chloride (a common plastic material)
QWERTY	Denotes the common layout of English-language computer and typewriter keyboards
RTF	Rich Text Format
SD	Shadow Detection
SDK	Software Development Kit
SQL	Structured Query Language

TABLETOP	The IEEE International Workshop on Horizontal Interactive Human Computer Interactive Systems
TIR	Total Internal Reflection
TOCHI	ACM Transactions on Computer-Human Interaction
TWEANN	Topology and Weight Evolving Artificial Neural Network strategy
USB	Universal Serial Bus
WIMP	Window, Icon, Menu, Pointing Device
WinForms	Windows Forms
WPM	Words Per Minute
WPF	Windows Presentation Foundation
XNA	XNA’s not Acronymed – Microsoft Game Development Platform
XML	Extensible Markup Language

CHAPTER ONE: INTRODUCTION

1.1 Motivation

In 2006, Jeff Han unveiled a multi-touch platform at TED (Technology, Entertainment, Design) [75] demonstrating a human-computer interaction system that the general public had only seen in science-fiction movies. Han demonstrated a touch-based system that employed gestural interaction as the primary mechanism for computer input. His system allowed users to rotate pictures, navigate digital maps, and visualize large amounts of scientific data with just your hands. Han’s TED talk triggered a wave of excitement, marked a turning point in the availability of commercial multi-touch platforms, and dramatically increased the number of active researchers in the field.

Since that time, multi-touch has proven itself to be a commercially successful computing platform through mobile devices such as the Apple iPad [5]; desktop computers such as the HP TouchSmart [31], and tabletop platforms such as the Microsoft Surface [52]. However, multi-touch platforms have not been adopted as “everyday” computer interaction devices – multi-touch has not been applied to general-purpose computing.

The questions we seek to answer are: Will the general public adopt these systems as their chief interaction paradigm? Can multi-touch – with features such as gestural interaction, tangible interfaces, pen-based computing, and interface customization – provide such a compelling platform, that it displaces the desktop mouse and keyboard? Is multi-touch the next revolution in human-computer interaction?

1.2 Multi-Touch

Multi-touch describes a classification of computer input device that detects multiple simultaneous points of contact on a two-dimensional surface. Other input devices collect spatial input, and often from multiple simultaneous sources, but the term "multi-touch" emphasizes the tactile nature of the technology and its origins in traditional touchscreen interaction. Touchscreens incorporate input and output into a single hardware component. The user interacts with a screen, through a stylus or his/her finger, and in conjunction with the visual display. The success of touchscreens is evidenced through everyday activities, such as using an ATM, purchasing groceries, or tapping out commands on a smart phone. However, traditional touchscreen technology has been limited to detecting a single point of contact, and from a single user. Multi-touch allows many concurrent users to interact with a display and through multiple points of surface contact.

1.3 General-Purpose Computing

We define general-purpose computing as the “every-day” work that is accomplished through the assistance of a computer. Applications such as email, web browsing, word processing, and spreadsheets, for example, are encompassed by the term general-purpose computing. Typically, activities with such applications are performed on a laptop or desktop computer and the dominant interaction mechanisms are the keyboard, mouse and monitor. In this definition, we classify computing platforms as “desktop” or “mobile”, where desktop and laptop computers fall under the “desktop” category, while mobile devices such as cell phones, Personal Digital Assistants (PDAs), smart phones, and newly emerging “tablet” computers such and the Apple iPad [5], Samsung Galaxy Tab [67], and Motorola Xoom [55] are classified as “mobile”. Certainly, desktops can be moved, and laptops are designed to be mobile – we make the distinction between desktop and mobile computing platforms based on their capability as general-purpose computing devices rather than their ease of transportation. Some might argue that mobile computing platforms are more prevalent than desktop computers, and can host many of the same general-purpose applications. However, while mobile computing platforms are becoming ever more powerful and feature-rich, we exclude them from the definition of general-purpose computing because of their poor ergonomics for long-term-single-session use.

To expand upon this notion, we observe that mobile computing devices are designed to be held in one or two hands. Thus, their form factor is such that the display screen is quite small and text input is typically performed by the user's thumbs. We suggest that these factors limit the amount of time a typical user will operate on such a platform in a single sitting. Short email messages, notes, or text messages are much more common on a mobile computing platform than is the composition of a lengthy document or spreadsheet. Moreover, the relatively small computing resources (processing power, memory, and non-volatile storage) of a mobile device limit interaction with applications that are common to a desktop computer. Furthermore, we suggest that work environment plays a large part in the type of task performed. If the user is on-the-move, he/she may be less disposed to work on a thought-intensive task than in the relative seclusion of an office, home, or even coffee shop.

Finally, we support our definition of general-purpose computing through the ergonomics of the user's desk. We suggest that long-term-single-session computer use is dependent, in large part, on the physical desk that the user sits at. The basic design of the desk (desktop + chair) has not been fundamentally changed in generations. Our desktop computing platforms have been designed for integration into this traditional workstation environment. With our office environments based upon desktop workstations, it is natural that general-purpose computing would be conducted in this setting.

In summary, we define general-purpose computing as the “every-day” work that is accomplished, through the assistance of a desktop computer, in a typical office environment.

1.4 Research Question

While defining general-purpose computing, we suggested a number of enabling factors: 1) a physical desktop workstation; 2) a large display and keyboard; and 3) sufficient computing resources for rich-content applications. We propose that these factors stem from the need for long-term-single-session use. Furthermore, if we analyze some common applications of general-purpose computing, we find a recurring theme: *general-purpose interaction depends upon text input*. Email, word processing, spreadsheets – all require considerable text input from the user. Perhaps the Achilles' heel of mobile computing platforms, as general-purpose computing devices, is their poor support for text input as compared to the full-size keyboards on desktop computers [12, 44]. The keyboard on a mobile device can be either software-based or a physical keyboard, but is always miniaturized to accommodate the form-factor of the device. Thus, the user generally types with his/her thumbs and is typically limited to pressing one key at a time. With only “single-touch” capability, chorded keys used for capitalization, punctuation, or menu short-cuts require multiple keystrokes and, anecdotally, reduce the user's efficiency as compared against a full-sized, "multi-touch" keyboard. Thus we propose that, in order to support general-purpose computing, the platform must:

1. Facilitate long-term-single-session use
2. Accommodate efficient text input

With the definitions of multi-touch and general-purpose computing in place, we make a final observation that leads to our research question: *multi-touch platforms have, thus far, been special-purpose devices* [20, 51, 95]. Multi-touch platforms have been presented as museum kiosks [34], collaboration devices [26], and as novel user-interfaces [20], but have not been applied to general-purpose computing. This is not to suggest that general-purpose applications cannot function atop a multi-touch platform. Indeed, a desktop computer, hosting a general-purpose operating system, is commonly a chief component of a multi-touch device, and can thus launch any compatible software applications. Our observation is that multi-touch platforms, to date, have not *targeted* general-purpose computing as the application to support the platform's existence. Moreover, many of today's multi-touch platforms do not support text input or do so only through a traditional keyboard peripheral. With the assertion that efficient text input is a requirement of a general-purpose computing platform, such multi-touch devices cannot be considered general-purpose.

Understanding that general-purpose computing relies upon efficient text entry, we ask: *Can a multi-touch device, without a text entry peripheral, provide a platform for efficient text entry?* And, by extension, *is such a platform viable for general-purpose computing?* Supporting questions include:

1. Are there existing text entry methods that are viable for a multi-touch platform?
2. If viable existing text entry methods are identified, how do they perform on a multi-touch platform?
3. For a given text entry method, how does performance change as characteristics of the method are modified?
4. How does performance differ among multi-touch technologies?
5. If the performance of existing methods compares unfavorably to traditional, physical-keyboard-text-entry, could we develop a suitable method specific to a multi-touch platform?

1.5 Research

Our work to answer these questions can be broken down into the following research thrusts:

1. Examination of existing text entry methods
2. Empirical evaluation of text entry and word processing methods

This section will introduce these topics and the association of each thrust to their respective research questions.

1.5.1 Examination of Existing Text Entry Methods

Our first research thrust examined existing text entry methods and their applicability to multi-touch. These efforts leveraged research by Hinrichs, et al. [27] who, in 2007, published an examination of text entry methods and their applicability to tabletop displays. In this work, they established a set of evaluative criteria for investigating text entry methods and a taxonomy for such systems. We have tailored their work for application on a single-user, general-purpose multi-touch platform, and detail these evaluative criteria in Chapter 2.4.1.1. Subsequently, we applied these criteria to identify existing text entry methods that could be viable for our multi-touch platform, and selected an on-screen, soft keyboard employing the QWERTY character layout, for empirical evaluation. We present our examination and rationale for this selection in Chapter 2.4.1.2

1.5.2 Empirical Evaluations of Text Entry and Word Processing Methods

Our second research thrust was the empirical evaluation of text entry and word processing methods upon multi-touch and physical computing platforms supporting the QWERTY character layout. We performed four user studies, and collected objective and subjective data on various text entry and word processing modes and techniques.

1.5.2.1 Text Entry Study

The first study examined the performance of our implementation of an on-screen, multi-touch QWERTY keyboard, across a variety of text entry modes, and compared the results against a physical keyboard. In particular, participants entered short phrases of text on a multi-touch device and a physical keyboard platform. These phrases were presented in all lowercase letters and in mixed-case, and we studied the multi-touch platform using two different input modes: “multi-touch” mode, which allowed any number of simultaneous surface contacts; and “single-touch” mode, which limited the platform to only a single simultaneous input. Additionally, we varied the text entry task between *Memorization* and *Copy*, where the phrase was either hidden (once the participant began typing) or where the phrase was continuously presented.

This study gathered a variety of performance metrics, including measurements of speed, accuracy, and conscientiousness (a participant’s tendency to fix errors). We also observed *how* participants input text through the multi-touch platform; in particular, how they positioned their hands and employed their fingers. This *Input Style* provides insights into the ergonomics of the platform. Furthermore, we asked participants a series of post-study questions to gauge their reaction to the multi-touch platform and the text entry task.

In this fashion, we established a baseline measurement of the performance of a multi-touch text entry system, across a variety of input modes. The results show that a physical keyboard outperforms the multi-touch platform; however the post-study questionnaire data offers support for the potential adoption of the multi-touch device for everyday text entry tasks.

1.5.2.2 Text Entry Improvement Study

We followed the initial study with an attempt to improve multi-touch text entry performance by modifying *how* text entry is accomplished through our multi-touch platform.

In the first study, we observed a tendency for participants to “drag” their fingers across the multi-touch keyboard to get to the next key. We suspected that this was a cause of mistakes made in entering text on the multi-touch platform, and this theory led us to develop an alternate technique for triggering a virtual key press on the multi-touch surface. In the original implementation, key presses are detected at the moment the user makes physical contact with the multi-touch surface and within the bounds of a virtual key (dubbed the *Key Down* technique). In the technique studied in the second experiment, key presses are detected once contact is made with a key and then released (*Key Up*).

We collected the same objective performance measurements employed in the first experiment, along with data on user preference between the two interaction techniques. No significant difference was found between the *Key Up* and *Key Down* techniques for any of the chief quantitative measurements. Nor was any significant difference found within the users’ preference between the two techniques.

1.5.2.3 Word Processing Technique Study

In the first study we found tentative support for the adoption of multi-touch for everyday text entry tasks. Still, multi-touch has more features available than text entry, and other aspects of the platform might make for a more compelling user experience than text entry alone. Moreover, while we assert that text entry is a critical element of general-purpose computing, it is certainly not the only component; nor is the task of entering text done without context. Indeed, a common application of general-purpose computing is word processing, where *plain text* is modified by formatting options, and the task of entering text combines keyboard and mouse-style interaction. Thus we asked: “How does the performance of a multi-touch platform compare against a physical platform, within a word processing task?” Furthermore, we wished to understand how the inclusion of mouse-style interaction would impact the adoption potential of the multi-touch platform. However, before we could answer these questions, we had to first investigate *how* one might accomplish word processing on a multi-touch system.

As mentioned, word processing typically involves formatting plain text; for example, text might be modified to appear bold or italicized. Such formatting is commonly applied through the use of a mouse-style pointing device. Thus, we had to determine how a user might engage formatting options on a multi-touch platform. Borrowing work from the state of the art in multi-touch (described in Chapter 2) and by leveraging our own research in mouse-style interaction through multi-touch (detailed in Appendix A), we entertained two approaches toward answering the question: *Direct Interaction* and *Indirect Interaction*.

Direct Interaction indicates that the user touches directly onto the portion of the screen he/she wishes to interact with. Conversely, *Indirect Interaction* specifies a system more akin to a traditional mouse or laptop track pad – where the user touches an area of the screen marked for mouse-style interaction, and doing so causes interaction with some other part of the screen. We also refer to the *Indirect Interaction* technique as “Virtual Mousepad” as it was designed to emulate the behavior of a traditional mouse – more specifically, our implementation mimics the operation of a typical laptop trackpad.

The third user study, then, compared the performance of a word processing task between the *Direct* and *Indirect Interaction* techniques on a multi-touch platform. Participants were presented a series of *rich-text* paragraphs and asked to transcribe this text along with punctuation, organization, and formatting. After they transcribed these paragraphs, participants were asked to choose a preferred interaction technique (with “no preference” as an available option).

Our results found statistical significance in only one of the 11 objective measurements – revealing a slight performance advantage of *Direct Interaction*. Additionally, the results found unanimous agreement that *Direct Interaction* was the preferred technique. This information provided clear direction for our fourth study: examining word processing between multi-touch and traditional computing platforms.

1.5.2.4 Word Processing Platform Study

Our fourth study leveraged the findings of the word processing technique study, to compare formatted text entry performance and user preference between traditional computing platforms and a multi-touch system. In particular, we collected data on a desktop computer with keyboard and mouse, a laptop with its integrated keyboard and trackpad, and a multi-touch platform. The intent of this experiment was similar to that of the original text entry study: to establish baseline performance characteristics of a task between multi-touch and traditional computing systems. Moreover, we wished to determine the impact of mouse-style interaction on the potential for adoption of the multi-touch platform for everyday computing.

As with the word processing technique study, participants transcribed a series of rich-text paragraphs – two paragraphs per platform. In addition to the quantitative performance data, we also asked participants to rank the three platforms in order of preference.

As expected, the desktop and laptop platforms outperformed the multi-touch system. However, there was some support that the multi-touch platform would be adopted for everyday word processing activities. Interestingly, this support was lower than that found in the original text entry study; suggesting that the inclusion of mouse-style interaction actually reduced the potential of adoption of the multi-touch platform.

1.5.3 Additional Investigations

The four studies we have just introduced are presented in detail in Chapters 3 through 6. Additionally, we have pursued a number of other topics in the area of general-purpose multi-touch computing, to help answer our research question. This section introduces some of these efforts.

1.5.3.1 General-Purpose Multi-Touch Platform (GPMT)

Our four empirical evaluations were performed on a commercially available multi-touch platform which, arguably, was not designed to support general-purpose computing. Indeed, throughout this writing we argue that multi-touch platforms, to date, have not targeted general-purpose computing. Thus, we have speculated on what a general-purpose multi-touch platform (GPMT) might look like. Section 7.1 discusses our vision of a GMPT, its design and our rationale.

1.5.3.2 Tactile Feedback

Throughout our investigations of multi-touch text entry, a glaring deficiency has been the lack of tactile feedback. We have often speculated that the performance of text entry would significantly improve if the multi-touch platform “felt” more like a traditional keyboard. Section 7.2 discusses our considerations for tactile feedback on multi-touch platforms through two principal elements: pressure sensitivity and the physical boundaries presented by traditional keys.

1.5.3.3 Audible Feedback

The software we developed for our studies provided audible feedback in response to a key press – a “tack-tack” sound recording of the depression of a physical key. However, we briefly studied alternate audible feedback mechanics, including the notions of *Voice Feedback* and *Bin-based Audio*.

Voice Feedback refers to playing a recording of a spoken letter when a virtual key is pressed. Specifically, we implemented voice feedback through two sets of sounds files and informally studied the performance impact of these implementations. *Bin-based Audio* describes a technique by which the surface of our multi-touch device is divided into an invisible grid of locations and each grid cell is mapped to a unique sound.

Ultimately, we rejected further pursuit of both audible feedback systems. Details of these explorations are provided in Chapter 7.3.

1.6 Contributions

Our research has focused on the task of quantifying the performance of text entry on a multi-touch platform. To our knowledge such research has not been previously conducted. Thus, our chief contribution is the establishment of empirical performance measurements for text entry tasks on a particular type of multi-touch platform – a flat screen, monitor-sized system that detects surface contacts through capacitance. These measurements form a baseline from which we can compare future efforts toward improving this performance. This baseline includes a substantial set of metrics across a variety of text entry modes and techniques. Moreover, we include observations of *how* users accomplish text entry on a flat screen multi-touch system.

Furthermore, we have implemented and analyzed one attempt at improving text entry performance through software – the *Key Up* technique introduced in Section 1.5.2.2. We also explored tactile and audible feedback mechanics – efforts aimed at improving performance.

Additionally, in the process of examining word processing on a multi-touch platform, we developed new metrics for measuring formatting errors made during word processing tasks.

Finally, we offered a vision for a general-purpose multi-touch platform and conclude, at least, that the pursuit of such a platform is a worthwhile endeavor.

1.7 Summary

In this chapter, we introduced our work in general-purpose multi-touch computing. We defined multi-touch as: *a classification of computer input device that detects multiple simultaneous points of contact on a two dimensional surface.* We defined general-purpose computing as: *the "every-day" work that is accomplished, through the assistance of a desktop computer, in a typical office environment.* And we proposed that, in order to support general-purpose computing, a platform must: 1) facilitate long-term-single-session use; and 2) accommodate efficient text input.

We observed that: *multi-touch platforms have, thus far, been special-purpose devices;* and this observation led us to ask: “Can multi-touch, without a text entry peripheral, provide a platform for efficient text entry? And, by extension, is such a platform viable for general-purpose computing?” With these questions in hand, we introduced our work through two research thrusts:

1. Examination of existing text entry methods
2. Empirical evaluation of text entry and word processing methods

For our initial thrust, we presented a tailored set of evaluative criteria for examining existing text entry methods, and applied these to identify a category of potentially viable input systems – on-screen soft keyboards.

For our second research thrust, we introduced the four user studies we conducted that examined text entry and word processing tasks. These studies employed our implementation of the dominant western character layout – QWERTY – through an on-screen keyboard for a multi-touch platform.

Finally, we introduced a set of additional investigations in general-purpose multi-touch computing. In particular, we touched on our vision of a general-purpose multi-touch platform (a GPMT), our consideration of tactile feedback, and a brief exploration of audible feedback in response to a key press.

The remaining chapters will expand upon this introduction, beginning with a background of multi-touch computing, presented in Chapter 2. Chapters 3 through 6 will present our four text entry and word processing user studies. Chapter 7 will detail our additional investigations and Chapter 8 will summarize our work, present our conclusions and offer suggestions for future efforts in this area.

CHAPTER TWO: BACKGROUND ON MULTI-TOUCH COMPUTING

2.1 Introduction

2.1.1 Origins

Multi-touch has a long history, evolving from of a number of input devices. Indeed, some might consider the first multi-touch device to be the keyboard. Han [24] suggests that Frustrated Total Internal Reflection (described in detail in Section 2.2.2.1) was used for touch interaction as early as 1970 [36]. However, we can trace the first multi-touch platform, using modern definitions, at least as far back as 1984 with Boie's Multi-Touch Screen [10]. Buxton published a paper on two-handed input in 1986 [11] and has been an active contributor to multi-touch technology for more than two decades.

Commercial products featuring touch screen interfaces began appearing in the late 1980s and early 1990s with products such as the Apple Newton [90], the IBM Simon [10, 89], and the Wacom digitizing tablet [10, 84]. These technologies later evolved into smart phones, and touchscreens such as the Apple iPhone [6] and Wacom Cintiq [83], both released in 2007.

The 1990s also saw a number of academic papers published on the topic of touch computing, its technology and interaction techniques including [4, 21, 23, 35, 39, 62, 63]. A key contributor to multi-touch technology was work by Westerman with his PhD dissertation on *Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-Touch Surface* in 1999 [86]. Westerman created a company – Fingerworks [18] – from this work, which was acquired by Apple in 2005.

It was the 2000s, though, that saw a major surge in multi-touch actively, and is when multi-touch moved beyond the lab and into the hands of consumers. In 2001, Deitz and Leigh presented the Mitsubishi DiamondTouch [14] a front-projection, multi-touch system that uses an array of antennas to transmit identifying signals that are capacitively coupled through the user. In 2005, Han presented work on constructing low-cost multi-touch surfaces using Frustrated Total Internal Reflection [24]. But it was Han's multi-touch demonstration at TED (Technology, Entertainment, Design) [75] that triggered a wave of excitement, marked a turning point in the availability of commercial multi-touch platforms, and dramatically increased the number of active researchers in the field. In 2006, Han created Perceptive Pixel [61], a commercial enterprise to market his efforts in multi-touch.

In 2004, Wilson, of Microsoft Research, introduced TouchLight [92] and in 2007 Microsoft announced the Microsoft Surface [52]– an exciting development for bringing multi-touch technology closer to the consumer market. In 2009, Microsoft released Windows 7 which incorporates multi-touch capability directly into the operating system.

2.1.2 Hollywood, the Media, and Multi-Touch

Hollywood has a history of capitalizing on technological advancements – whether actually realized science or science fiction – and their adoption of multi-touch is no exception. We believe this is beneficial to both researchers and the general public for 1) providing exposure to a new technology that might otherwise be unknown outside of academia, and 2) offering inspiration to researchers on how new user interfaces could be employed. Hollywood began exploring multi-touch as early as 2002 with the film *Minority Report* and continued in 2005 with the film *The Island*. However, it was 2008 that witnessed a major uptick in multi-touch references, both in film and television, with *Quantum of Solace*, *Babylon A.D.*, and the 2008 U.S. Presidential Election. The Election coverage made extensive use of multi-touch, notably employing CNN’s Magic Wall [13, 25] provided by Jeff Han’s Perceptive Pixel [61].

2.1.3 Multi-touch Components

We break down the field of multi-touch into two broad categories: hardware and software. Our discussion of hardware focuses on the physical components of a multi-touch platform. This includes a treatment on the various techniques for detecting surface contact and in-depth discussions of the most popular approaches. The software category includes multi-touch development frameworks, sensing techniques to refine and compensate for hardware deficiencies, gesturing, and multi-touch presentation layers.

These systems do not exist in isolation, and certainly work in conjunction, but we divide the topic in this fashion to facilitate discussion and to recognize that hardware and software from different vendors can be made to interoperate.

2.2 Hardware

When considering multi-touch hardware, we can further categorize the components into those detecting input and those producing visual output – input and display. Again, this is not to suggest that input and display technologies are independent or can be combined in any configuration. Indeed, we present a matrix, further in this section, listing common combinations of input and display technologies. We separate these components to frame the discussion and call out specific techniques.

2.2.1 Display

There are two varieties of multi-touch display technology that are presently being employed: projected and non-projected displays; with unique considerations and tradeoffs for each.

Projected displays are either front- or rear-projected, indicating that the projector is placed in front of or behind the diffuser, respectively. The diffuser is the material that reflects the projected image such that it will be visible to the user. When considering front- or rear-projected displays an initial consideration is the effect of image occlusion. Recall that, in multi-touch systems, the display surface is also the interaction surface. Generally speaking, rear-projected displays will not be occluded by the user, whereas the user's hands, arms, or entire body might occlude a front-projected image as he/she interacts with the display.

A secondary consideration, for either front- or rear-projected displays, is the throw ratio of the projector. A projector's throw ratio is defined as the distance, between the lens and the diffuser, divided by the width of the image it will project. Thus a projector with a throw ratio of 2:1 requires 2 units of throw for every 1 unit of image width. This value is important, as it dictates the installation requirements for the display. Arguably, this value is more critical for rear-projected displays, as the throw ratio defines the depth requirement of the hardware – whereas, front-projected displays may have more flexibility in the throw distance. Traditional projectors have a throw ratio typically greater than 0.85, while short-throw projectors range from 0.5 to 0.7. Thus, short-throw projectors might fit form-factors with small depth requirements, though typically at an increase in price compared with the more common, longer-throw projectors.

Additional factors for projected displays include: aspect ratio, resolution, luminosity, contrast, vertical offset, and keystone correction. As with non-projected displays, modern projectors support a native aspect ratio (image width divided by height) of 4:3 or 16:9, and will often support the non-native aspect ratio as well (though with a corresponding reduction in light output). However, projector resolution (the number of pixels projected), and contrast (the ratio between the brightest white and darkest black) will almost always compare unfavorably to a non-projected display system. A projector’s luminosity, the quantity of light being projected (i.e. brightness), is an important consideration for installations in environments with high ambient light. Lastly, a projector’s vertical offset indicates the amount of the image that is presented above the projector’s lens. A 100% vertical offset specifies that the beginning of the image is exactly in line with the lens. Whereas a 50% vertical offset indicates an on-axis projector, with half of the image projected above the lens, and half below. Short-throw projectors will often have vertical offsets greater than 100%. This value factors primarily into encased multi-touch systems, where the projector is limited in its offset with respect to the diffuser. Tilting the projector will compensate for a small offset, but requires either lens shifting or digital keystone correction.

When choosing between projected and non-projected display technologies, two factors are chief among concerns: cost and image size. Projected displays are less expensive per unit of image size than non-projected displays. And for very large, non-tiled displays, non-projected options may simply not exist. Projected displays are most appropriate for semi-permanent or fixed displays such as tabletops, kiosks, or wall-sized multi-touch systems. Though pocket-sized projectors, such as the 3M MP180 [2], are starting to emerge; desktop, laptop, or mobile multi-touch platforms are presently the domain of non-projected displays: namely flat-screen LCD technology.

2.2.2 Input

There are generally two approaches being employed to detect multi-touch input: vision and electricity. Vision-based systems use light – either emitted, reflected, or occluded – to determine the presence of a surface contact. Electrical-based systems use capacitance, electrical resistance, or a related principle.

The following lists the most commonly implemented vision-based multi-touch input technologies:

- Frustrated Total Internal Reflection (FTIR)
- Diffused illumination (DI)
- Diffused Surface Illumination (DSI)
- Optical Occlusion (OO)
- Laser Light Plane (LLP)
- Shadow Detection (SD)

2.2.2.1 Frustrated Total Internal Reflection (FTIR)

Total Internal Reflection (TIR), introduced in relation to multi-touch by Han [24] in 2005, is the same principle that governs fiber-optic cable. When light enters a waveguide, and encounters an interface with a lower index of refraction, that light is reflected away from the interface, and back into the waveguide. A typical multi-touch surface, utilizing Frustrated TIR, employs a sheet of transparent acrylic whose interface, on either side, is typically air (a medium with a lower index of refraction). By surrounding the acrylic with infrared light-emitting diodes (LEDs) and emitting light “edge-on” into the acrylic, the light will undergo TIR past a critical angle (generally exceeded within the first few millimeters of the light-acrylic interface). A user can “frustrate” the TIR by changing the index of refraction at the acrylic-air interface – namely by touching the surface with a finger. At the point of surface contact, a portion of the light will escape the waveguide and enter the user’s finger. A portion of this light will be absorbed and converted to heat, and some will be reflected orthogonally through the surface. If we place an infrared camera beneath the waveguide with its lens parallel to surface, we can detect this reflected light. FTIR’s popularity likely stems from its low-cost, simple implementation and its scalability to large displays. However, there are a number of issues that must be addressed to achieve an adequate FTIR multi-touch solution, most notably the need to couple the user’s fingers to the waveguide and reflect the light with near-zero pressure. Additionally, as with most of the vision-based techniques, dynamic lighting environments, or excessive ambient infrared light can negatively impact the systems performance. These issues are discussed in detail in [82].

2.2.2.2 Diffused Illumination (DI)

Diffused Illumination [58] is similar in concept to FTIR, in that light is reflected by the user’s fingertips to an infrared camera beneath the surface. However, with DI, the light is not conducted through a waveguide. Instead, light is emitted from beneath and toward the interaction/display surface. This technique obviates the need for a coupling material between the user’s fingers and the interaction surface. It also allows for more flexibility in the choice of surface material – as the material no longer needs good optical transmission properties. It need only be transparent to infrared light (and visible light if a rear-projected display is employed). However, diffused illumination requires a fairly uniform distribution of infrared light across the surface, and with enough intensity that the small percentage of light, reflected by a fingertip, is detectable by the camera. Additionally, DI is indiscriminant about what reflects light and hence the accompanying software may need to filter out a “glowing” palm or forearm. Furthermore, DI is subject to “hovering” where reflection begins slightly above the surface and without explicit contact. This may be positive or negative depending on the application and can be mitigated through software configuration.

2.2.2.3 Diffused Surface Illumination (DSI)

Diffused Surface Illumination [65] could be considered a hybrid between FTIR and DI. As with FTIR, light is injected into a waveguide, but does not undergo total internal reflection. Instead, the waveguide contains micro-facets that evenly distribute the light throughout the guide and allow it to escape across the entire surface. When a user touches the surface, light is reflected, as it is with DI, to be captured by a camera. This technique eliminates the need for a coupling material (a compliant surface between the user and the waveguide) and ensures an even distribution of emitted light. However, the common source of DSI acrylic, a product called EndLighten [16], is more expensive than standard acrylic. Furthermore, less light is reflected, and is therefore more difficult to detect and differentiate from noise than with FTIR and DI systems.

2.2.2.4 Optical Occlusion (OO)

Optical Occlusion describes a technique where opposite light transmitters and receivers and arrayed across a surface or at its corners. When a finger is placed on the surface, light is prevented from reaching its corresponding receiver and this occluded “light line” or “light radial” is reported as a surface contact. Detecting the occluded light at different angles allows the determination of the contact’s two-dimensional location. However, this approach is limited by the number of simultaneous points whose positions can be unambiguously determined. This technique has been employed by the HP TouchSmart [31].

2.2.2.5 Laser Light Plane (LLP)

As with FTIR, DI, and DSI, Laser Light Plane [59] technology uses a camera to detect light reflected off of a surface contact (i.e. a finger). With LLP, the emitted light is highly focused (laser) and transmitted as a layer just above the interaction surface. When the user’s finger disrupts the beam, some of the light is reflected toward the camera and detected. This technique is similar to Optical Occlusion, in that the number of simultaneous points that can be unambiguously detected, is limited by the number (and angle) of light sources that are employed.

2.2.2.6 Shadow Detection (SD)

Thus far, we have described vision-based systems that detect emitted light. However, the inverse is also possible; that is, detecting the absence of light caused by surface contact. In this case, the emitted light could lie in the visible or infrared parts of the spectrum, and is either intentionally emitted in the direction of the camera or is ambient in the environment. As the user interacts with the surface, he/she interrupts the light that would otherwise reach the camera, creating a shadow that the camera can detect. These systems are limited to front-projection and are affected by a dynamic lighting environment, but they do represent an interesting, and potentially low-cost, approach to multi-touch input detection. Work by Echtler [15] demonstrates a multi-touch system using shadow detection.

2.2.2.7 Capacitance

Capacitance based multi-touch systems can generally be classified as either surface capacitive, projected capacitive or capacitive coupling. “Surface capacitive technology consists of a uniform conductive coating on a glass panel. Electrodes around the panel’s edge evenly distribute a low voltage across the conductive layer, creating a uniform electric field. A touch draws current from each corner. The [hardware] controller measures the ratio of current flow from the corners and calculates the touch location.” [78]

Projected capacitance systems contain a grid of micro-fine wires sandwiched between layers of protective glass. When a user touches an outer glass layer, capacitance forms between the grid and the user’s finger. The hardware controller measures the change in the grid’s capacitance to determine the position of the surface contact. [77]

Capacitive coupling is the technology employed by the Mitsubishi DiamondTouch [14] and describes a system where the user completes a circuit between a transmitter and receiver. The DiamondTouch and its capacitive coupling technology is unique among the described hardware systems, in that it is capable of associating a surface contact with the specific user (up to four) touching the screen. This is possible because each user is physically connected to a receiver, either by standing on a conductive plate, or sitting on a chair connected to the plate. An array of transmitters is embedded at regular intervals within the surface. As the user touches the surface, he/she completes the circuit and the specific frequency assigned to that user is registered at that location.

2.2.2.8 Resistance

Resistance-based multi-touch systems are generally made of two layers of conductive material separated by an insulator. The interaction surface is made of a flexible material that, when pressed, connects the two conductive layers and creates an electrical circuit. A hardware controller measures this connection to determine the position of the surface contact. Such systems are commonly found in mobile gaming devices [56], and PDAs and do not differentiate the type of interaction device (i.e. finger or stylus).

2.2.2.9 Interpolating Force-Sensitive Resistance (IFSR)

IFSR is a subset of resistance-based multi-touch, but is of enough significance to warrant special attention. Unlike traditional resistance technologies, which simply detect a surface contact, Force Sensitive Resistors (FSRs) become more conductive as force is applied. “FSR ink is bumpy at a microscopic level so that when two ink-coated surfaces are pushed together, the surface area in contact increases. When this happens, electricity flows from one layer to another. FSR sensors operate by sandwiching layers of FSR ink between sheets of plastic printed with conductive wires.

When pressure is applied to a point on an FSR sensor, current flows from powered wires in one layer through the FSR ink to wires in the other sheet.” [1]. By measuring the amount of current through the system, a controller can determine how much force has been applied. IFSR platforms can be printed on paper-thin substrates for integration into flat-screen display systems.

We consider IFSR a significant innovation for multi-touch, particularly with respect to general-purpose computing, because of its support for pressure detection. We content that pressure sensitivity is of vital importance to improving multi-touch text entry performance, as pressure detection is likely to reduce unintended key presses (errors). We discuss pressure sensitivity further in Section 7.2.1.

2.2.2.10 Input Summary

Other multi-touch input hardware technologies exist, including Hodges and Buxton’s ThinSight [29], but the ones described in this section are currently the most prominent. As mentioned previously, the input and display technologies are not entirely interchangeable. Table 1 provides a matrix of common combinations of these systems. Please note, that the absence of a combination does not preclude the systems from interoperating. The list merely indicates the most common pairings of these technologies.

Table 1 Common combinations of input & display technologies

Input Technology	Rear Projection	Front Projection	LCD
FTIR	*		
DI	*		
DSI	*		
OO			*
LLP	*		
SD		*	
Surface Capacitance			*
Projected Capacitance			*
Capacitive Coupling		*	
Resistance			*
IFSR			*

2.3 Software

Multi-touch software can be categorized into lower-level device interaction and higher-level application development. Device interaction refers to “driver” software, or software which communicates with the input hardware to detect surface contacts. If the hardware is capable of fully processing surface contacts, this software layer might be fairly thin – having little functionality – and may simply provide the hardware-collected data to the operating environment. Otherwise, the device interaction software might work in conjunction with the hardware to complete the processing of surface contacts, making them useful for higher-level applications. In particular, vision-based input systems will often relay noisy data, or may simply forward on raw camera frames with little-to-no hardware-based processing. In these scenarios, the device interaction software is responsible for detecting surface contacts within the data stream. Appendix A.1.3 provides details on an image processing pipeline for vision-based multi-touch systems.

In addition to point detection, common features for lower-level device interaction software include:

- Point Tracking
- Hit Testing
- Shape Detection
- Gesture Recognition
- Presentation-Layer Independence

Point tracking refers to the ability to track surface contacts over time. For many input systems, data is provided as a “snapshot” in time. That is to say, that the hardware itself does not label or track a surface contact from one time step to the next (frame-to-frame). The software device interaction layer will uniquely identify a surface contact and determine if that contact is present in subsequent frames. This can be a challenging task for input devices that provide noisy data or capture data at a slow rate. For example, it is not uncommon for a user to inadvertently, and momentarily, lift a finger from the surface while moving along a path. When the surface contact “reappears” in the hardware data stream, the software needs to determine if the point is completely new, or was a type of “skip” or “stutter”. In another example, the user may move his/her fingers across the surface faster than the input device can detect. This is particularly troublesome for vision-based systems using low-cost cameras, whose frame rate may be quite low (e.g. 15-30fps) even at low resolutions (e.g. 320x240 or 640x480 pixels). In such circumstances the software needs to predict the trajectory of the surface contact to compensate.

Hit testing refers to a common requirement of higher-level multi-touch applications – the ability to query a position of the surface for interaction.

Shape detection will process the shape of a surface contact to determine what type of object is causing the interaction. Most commonly, these objects will be: a finger, an entire hand, or an undefined blob. Shape detection might also label fingers as belonging to a left or right hand and identify which of the five fingers is making contact. Another common feature, which we classify under shape detection, is interaction with tangible objects on a multi-touch surface. Jorda, et.al [37] demonstrated tangible object interaction with their reacTable project. Microsoft demonstrated this concept as well, with their Surface product, by placing smart phones and MP3 players on the Surface for interaction with multi-touch enabled software [52]. Such interactions are generally performed using fiducials – bar codes or visual patterns easily identifiable by vision-based systems.

Gesture recognition refers to the identification of movements on a multi-touch surface and mapping them to actions. In its most generic form, gesture recognition allows for the training of gestures, through some form of machine learning algorithm, for subsequent classification. Gesture recognition can also be done heuristically, and the device interaction layer may pre-define a set of common gestures. Appendix A.2 describes a degenerate case of multi-touch gesture recognition, where single contact points are combined into ink-style stroke information for training and classifying 2D symbols.

Presentation-layer independence is the final feature we include in the common set of device interaction software functionality, and is actually more the absence of a limitation. Specifically, presentation-layer independence refers to the accessibility of multi-touch input across operating and graphical presentation environments. A multitude of operating systems, programming languages, and graphical presentations environments exist, including: Microsoft Windows, Linux, MacOS; C++, C#, Java; OpenGL, DirectX, XNA, WinForms, XWindows, and Windows Presentation Foundation

(WPF); to name but a few. The multi-touch device interaction layer should be largely independent of the choice of environment. This can be accomplished through ports of the software, but a popular alternative is the use of network protocols to transmit multi-touch data. In particular, the open-source community has embraced the Open Sound Control (OSC) specification [60] an open, lightweight, message-based protocol for communicating multi-touch data to non-native environments. OSC is the underlying specification for the reacTable project's TUIO protocol [38], which has been widely adopted for transmitting multi-touch interactions over the network. Appendix A.1.3 provides additional details concerning OSC and its employment in relation to multi-touch.

Moscovich [54] describes additional multi-touch components in his Ph.D. dissertation

2.4 Multi-Touch for General-Purpose Computing

The second category of multi-touch software encompasses higher-level applications. This is the true "purpose" of multi-touch technology: the development of software systems that provide natural user interfaces for compelling, productive experiences. As discussed in Chapter 1, we contend that multi-touch applications, to date, have been special-purpose. Our work explores the application of multi-touch to general-purpose computing – the every-day work that likely occupies the majority of our time spent at a computer.

Other researchers have begun to look at multi-touch for general-purpose computing and the examination of text entry on multi-touch platforms. Hinrichs [28] has developed an interesting text entry system – BubbleType – for a tabletop multi-touch system. In 2007, Wigdor published an “analysis of long term office use of a multi-touch table” using the Mitshubishi DiamondTouch [87]. And, of course, the field of text entry on myriad computing platforms has a long and successful research history. Most notably, work by researchers including MacKenzie, Soukoreff , Zhang, Wobbrock, and Myers [41-48, 50, 68-71, 93] has directly influenced our efforts. We discuss much of this this work in detail in the remaining chapters.

Furthermore, we have observed a recent trend in the support of multi-touch directly within popular commercial operating systems. For example, the operating systems for the Apple iPad [5], and Motorola Xoom [55] have built-in multi-touch support and expose multi-touch capability to third-party software developers. And, perhaps most significantly, Microsoft, with its release of Windows 7 in 2009, has directly integrated multi-touch support within its general-purpose operating system. We believe that such support indicates the potential for multi-touch as a general-purpose computing platform.

2.4.1 Examination of Existing Text Entry Methods for Multi-Touch Application

As introduced in Chapter 1, our initial research in general-purpose multi-touch computing was the examination of existing text entry methods. Our efforts leveraged research by Hinrichs, et al. [27] who, in 2007, published an examination of text entry methods and their applicability to tabletop displays. In this work, they established a set of evaluative criteria for investigating text entry methods and a taxonomy for such systems. We have tailored these criteria for application on a single-user, general-purpose multi-touch platform.

2.4.1.1 Evaluative Criteria

Although our multi-touch platform is intended for individual use, and therefore deviates from the form factor of tabletop displays, we have adopted the framework that Hinrichs [27] has established as a basis from which to explore text entry methods for general-purpose multi-touch computing. Specifically, Hinrichs et al. include the following criteria for method evaluation: *Visual Appearance*, *Performance*, *Environmental Factors* and *Simultaneous Interaction*.

Visual Appearance relates to text entry methods with a visual representation of characters, and Hinrichs et al. define this through the *character arrangement* and *character layout*. The arrangement of the characters (keys) refers to their shape (e.g. rectangular or circular) while the layout describes where keys are positioned within the keyboard (e.g. Q next to W, W next to E). Hinrichs et al. suggest that these factors can impact the performance of a text entry method.

They defined *Performance* by two factors: *efficiency* and *ease of learning*. Efficiency is generally described as the speed of text entry that a person can achieve through a particular entry method, reported as words per minute (WPM), and the number of errors they commit while typing (accuracy). MacKenzie et al. [12, 50] provide evidence that the efficiency of a text entry method with a visual appearance is a function of the *visual search time* to find a particular character, and the *movement time* from one character to the next. Thus, efficiency can be improved by choosing a character layout that minimizes the distance between consecutive characters. However, Ward et al. observed that ease of learning and familiarity are often at odds with efficiency, with respect to method adoption [85]. Therefore, we support Hinrichs' notion that one must rate the importance of efficiency and learnability of a text entry method, depending on the targeted application.

Hinrichs et al. also include *Environmental Factors* as an evaluative criterion. These factors include: *space requirements*, *rotatablity*, *direct-touch interaction*, and *mobility* – elements that are perhaps particular to tabletop displays versus desktop computers or mobile devices. The space requirements of a tabletop platform are typically larger than those of a traditional desktop computer, as the platform will often support multiple users and a collaborative workspace [26]. The size of the workspace often impacts the choice of a text entry method, and two general approaches are possible: an external text entry method or an on-screen method. An *external* text entry method describes a peripheral device used in conjunction with the tabletop platform while *on-screen* indicates a method where the text-entry system is superimposed on the tabletop display. Hinrichs also introduced the notion of *collapsibility* – the ability of an on-screen text-entry method to be “collapsed” or otherwise temporarily hidden. In our work, space requirements are still important, though the specifics differ from a tabletop form factor with affordance for multiple concurrent users. However, our research question is specific to the viability of on-screen text entry methods on a multi-touch platform. Thus, we safely eliminate external text entry methods when evaluating existing systems.

Rotatability denotes the orientation dependency of tabletop displays, since such displays can often be approached from different directions. In our research, we disallow rotation and design affordance for only one useful orientation.

Direct-touch describes an interaction technique where the input mechanism is superimposed on the display. This is the hallmark of a traditional single- or multi-touch display [14, 52] and such interaction is generally performed using styli or hands. Examples of *indirect-touch* systems are the Wacom Bamboo [84] or a common laptop touchpad, where the interaction mechanism is still operated by hands or styli, but is separate from the display surface. In our work, we are focused on direct-touch systems capable of detecting multiple points of simultaneous contact.

The final environmental factor, *mobility*, indicates the text entry method's affordance for relocation. For tabletop displays, where the user can move to another position around the table, this factor is quite important. A wired keyboard, for example, could hamper the person's ability to move. Likewise, an ideal on-screen text entry method would allow the system to be presented from any location around the table. For our research, mobility is less of a factor since the intended use is for a single-user with no affordance for approaching the platform from different directions. However, we do consider this an important factor from the perspective of integration with the traditional office workstation.

Hinrichs' last criterion for text entry method evaluation is simultaneous interaction. This denotes collaboration of multiple individuals on a single input device. For tabletop displays, simultaneous interaction is an important factor. However, for our research focus – that of a single individual – we can forego this criterion. Please note that simultaneous interaction should not be confused with concurrent surface contacts made by a single person.

In summary, we have discussed a set of evaluative criteria used in an examination of text entry methods. Adopting Hinrichs' evaluation framework [27] as a basis for this aspect of our work, we have tailored the criteria for application on a single-user, general-purpose multi-touch platform. The modified criteria are presented in Table 2.

Table 2 Text entry method evaluative criteria

Evaluative Criterion	**Defining Characteristics**
Visual Appearance	Character Arrangement Character Layout
Performance	Efficiency Ease of Learning
Environmental Factors	On-Screen Space Requirements Direct-Touch Interaction Mobility (as pertaining to workstation integration)

2.4.1.2 Identifying Viable Existing Text Entry Methods

With our evaluative criteria in hand, and again leveraging related work, our initial research thrust sought to identify existing text entry methods for viability on our multi-touch platform. Hinrichs et al. introduced a taxonomy of text entry methods, categorizing them as either: *external methods*, those requiring an external physical device; or, *on-screen methods*, whose interaction mechanism is superimposed on the display [27]. They went on to label *physical keyboards*, *mobile physical keyboards*, and *speech recognition* as external text-entry methods. *Physical keyboards* refer to the traditional desktop computer keyboard, without respect to their character arrangement or layout. *Mobile physical keyboards* are text-entry systems commonly found on PDAs or smartphones; and *speech recognition* systems are an alternative to manual text entry, using a microphone and one's voice for automated dictation. Hinrichs et al. continued by categorizing *handwriting*, *gestural alphabets*, *stylus keyboards*, *soft keyboards*, and *gesture-based keyboards* as on-screen methods.

Handwriting refers to text entry methods that support natural writing – typically collected through a stylus or a user's fingers. Alternate mechanisms of handwriting text entries include digital encoding pens such as the IOGear Digital Scribe [33] or optical character recognition, as a post-processing step, in conjunction with an optical scanner. Since our focus is on real-time text entry, we constrain our consideration of handwriting methods to real-time input only. A significant benefit of handwriting as a digital input mechanism is the ease of learning. Indeed, users need only familiarize themselves with the stylus and associated software application, as they can reuse familiar writing skills. However, the performance of handwriting text entry is typically quite low, relative to other input methods. In 1999 MacKenzie and Soukoreff offered evidence that the speed of handwriting is approximately 15 wpm [46] – compared with expert typing speeds of 56 wpm on a physical QWERTY keyboard [57]. Additionally, handwriting recognition systems are not 100% accurate in identifying a person's input [41], further reducing the performance of such systems.

Gestural alphabets are a subset of handwriting entry methods, developed to improve performance. These systems replace the relatively free-form input of individual handwriting with a specific representation of each character in the alphabet. Examples of gestural alphabets include *Unistrokes* [22] and *Graffiti* [8, 49], introduced in 1993 and 1995, respectively. Unistrokes' performance was found to be 34 wpm [22, 46] – more than twice the speed of traditional handwriting. However, such efficiency improvements come at the cost of learnability and potentially limited adoption [85]. Nonetheless, gestural alphabets have found commercial success in mobile devices and can be readily implemented on a multi-touch platform. For both real-time handwriting and gestural alphabets, a visual representation is optional. However, providing an on-screen or physical boundary, within which recognition is performed, can aid the user in providing “clean” input. Likewise, such systems can echo the user’s input strokes to provide a visual grounding from which to construct words and sentences.

Stylus keyboards are on-screen text entry methods with visual representations. We reuse this classification label from Hinrichs’ work [27], but note that the term “stylus” does not preclude the use of one’s fingers to perform input. These entry methods present a virtual keyboard on the display screen which can often be tailored for an application or individual. Moreover, these keyboards might provide tactile or auditory feedback in response to a key press. Hinrichs et al. define two sub-categories of stylus keyboards: *soft keyboards* and *gesture-based keyboards*.

Soft keyboards have the direct visual mapping of physical keyboards and accept input in the same fashion as physical keyboards – touch – albeit through a graphical representation of the character set. To date, most soft keyboards have accepted only one point of simultaneous contact, thereby excluding the notion of concurrent modifier keys (e.g. shift, alt, or ctrl). Soft keyboards commonly have a rectangular shape, though recent work has included circular representations as well [28] . As with physical keyboards, the character layout impacts the performance of the input system. While the most common western layout is QWERTY, many alternatives have been introduced, intended to improve performance over the de facto QWERTY standard. These include: Dvorak, Fitaly, OPTI, and Metropolis [47, 50]; as well as alphabetic character layouts or arrangements based on frequently used letters or digraphs. Predictive models exist for some of these character layouts implemented through soft keyboards [71]. However, these models do not always agree with empirical studies. Moreover, these models have, to date, been developed for input systems with single-touch capability and must be updated to take advantage of multi-touch platforms.

Gesture-based keyboards are stylus keyboards that allow for continuous gestures that connect visually presented characters to form a word [27]. Research has demonstrated that such keyboards can improve text entry performance [27]. However, such performance improvements can come at the expense of learnability. Additionally, the input mechanic of gesture-based keyboards is comparable to continuous handwriting with consideration of the single-touch nature of the input. This prompts the question of the efficacy of this text entry method versus two-handed soft keyboard input.

When considering our application of single-user-general-purpose computing on a multi-touch platform we can narrow the aforementioned set of existing text entry methods. In particular, our research questions the viability of a multi-touch platform as a general-purpose computing device without the use of external text entry peripherals. Therefore, we can eliminate the set of external text entry methods, including: physical keyboards, mobile physical keyboards, and speech recognition; from our list of potential text entry methods. This is not to suggest that a multi-touch computing platform could not benefit from these input systems. We chose this route to study the viability of the multi-touch platform for general-purpose computing as a stand-alone platform. Thus, for the purposes of our research we have considered only on-screen text entry methods for empirical evaluation. Of the four categories of on-screen methods: *Handwriting*, *Gestural Alphabets, Soft Keyboards,* and *Gesture-based keyboards*; we suspect that handwriting and gestural alphabets are unlikely to benefit from a multi-touch platform. These input methods, whether employing a stylus or finger, are inherently single-touch systems. This is also true for gesture-based keyboards where, although a visual representation of a keyboard is presented, the user traces a gesture through the keys with finger or stylus and does not provide input via touch-tapping. Therefore, through this examination of existing text entry methods, we have identified soft keyboards as the focus of empirical study as applied to our multi-touch platform.

Furthermore, we recognize the QWERTY character layout as the de facto standard for western keyboards. Moreover, our ultimate goal is to determine the viability of multi-touch as a general-purpose platform; and this requires comparison of performance against the predominant text-entry method: the physical keyboard. Thus, we have focused on evaluating the QWERTY layout, implemented on a single-user multi-touch platform with support for two-handed input and multiple points of simultaneous contact.

2.5 Summary

Multi-touch interfaces have the potential to radically change Human-Computer interaction. The technology has been at least 25 years in the making, and recent years have seen a tremendous surge in excitement over the modality. This chapter has provided a brief background into the origins of multi-touch, and a look at the most common multi-touch hardware and software components in use today. We have also presented our examination of existing text entry methods and our selection of a soft keyboard QWERTY layout for empirical evaluation.

CHAPTER THREE: TEXT ENTRY STUDY

3.1 Introduction

As a first step in investigating the viability of multi-touch for general-purpose computing, we pose the question: “c*an multi-touch devices, without a text entry peripheral, provide a platform for efficient text entry?”* It is unclear if there is a simple yes-or-no answer to this question – more likely it is a matter of degree and tradeoff between text entry performance and additional benefit provided by the multi-touch platform. However, before we can attempt to improve upon multi-touch text entry performance, or offer the user recompense for any expected decrease, we must first establish a baseline of text entry performance on a multi-touch platform.

3.2 Experiment

This experiment is an exploratory evaluation of text entry performance on a multi-touch platform. As described in Section 3.2.2, text entry performance is primarily quantified as speed, commonly measured in words per minute (WPM), and accuracy, a rate of errors made during text input (both corrected and uncorrected). Our design follows the structure of work presented by Soukoreff, MacKenzie, and Zhang, et al. [42, 46-48, 50] over the past two decades. Specifically, participants are presented a series of phrases, one at a time, and asked to transcribe those phrases through a text entry device. Software records the corresponding input stream for subsequent analysis.

In this study, we seek an understanding of the text entry performance characteristics of a single-user multi-touch platform and compare them against the performance of a physical keyboard. Our results establish a baseline from which to evaluate future efforts in multi-touch text entry.

3.2.1 Materials and Methods

3.2.1.1 Participants

Forty-eight unpaid volunteers participated in this study (6 females, 42 males). All were right-handed and had no visual or physical impairments that prevented normal computer use (as reported by the participants). Their ages ranged from 22 to 48 with an average age of 28. The participants rated their familiarity with touch screen computer systems (including single-touch or multi-touch) from 1-5: 5 highest; their responses were 2%, 8%, 4%, 10%, and 75%, respectively.

3.2.1.2 Apparatus

Two separate platforms were used by each participant during the experiment: a computer with a physical keyboard and another with a multi-touch monitor. Hereafter, we will refer to these platforms as the *Physical Keyboard/Platform* and the *Multi-Touch Keyboard/Platform*.

Figure 1 Dell SK-8135 keyboard

The physical keyboard platform received input from a Dell keyboard, model SK-8135 pictured in Figure 1. This is a traditional QWERTY keyboard with 104 keys including a full number pad. The multi-touch platform employed a 3M M2256PW [3] pictured in Figure 2. This is a 22” capacitive multi-touch monitor capable of detecting 20 points of simultaneous contact. Both monitors displayed a resolution of 1680x1050 pixels. Both platforms were situated on identical desks with adjustable height chairs. However, the mounting stand for the 3M M2256PW elevated the interaction surface approximately 9” above the desk. Therefore, we provided a traditional drafting-table chair, for the multi-touch platform, that allowed participants to sit at an appropriate height, relative to that of the physical keyboard platform. The 3M multi-touch monitor allows its orientation to be adjusted from 0-degrees (fully horizontal) to 90-degrees (fully vertical). Participants were instructed to adjust the orientation of the multi-touch monitor and chair height to best suit themselves.

Figure 2 3M M2256PW 22” multi-touch monitor

3.2.1.3 Software

A custom C# software application, pictured in Figure 3, was developed to present a virtual keyboard on the display and collect data for our study. This application accepts input from the physical keyboard, the mouse, or the 3M M2256PW multi-touch monitor; without special configuration. Thus, this same application was presented for both the physical keyboard and multi-touch platforms. During the experiment, the virtual keyboard was still presented on-screen for the physical keyboard platform, even though participants interacted with the physical keyboard. The virtual keyboard provides visual and auditory feedback as keys are pressed. Toggle-style keys such as the caps-lock or num-lock, display the *key-down highlight* graphic when toggled to the *on* state. Additionally, uppercase letters are displayed for character keys, when a shift key or the caps-lock key is depressed.

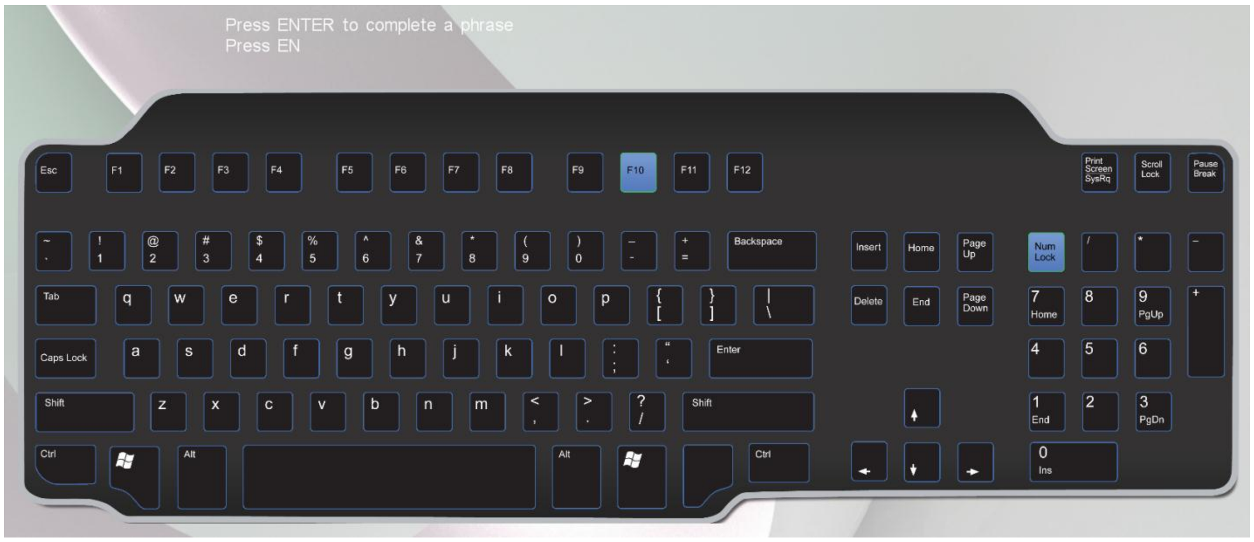


Figure 3 Text entry study virtual keyboard software

The graphical presentation for the virtual keyboard mimics the Dell SK-8135 keyboard and closely approximates its key spacing. However, the 3-dimensional aspect of the physical keys is not reproduced and virtual keys are presented as flat 2-dimensional graphics. Nonetheless, the rectangular bounding volumes of the 2D virtual keys mimic the outer boundaries of their corresponding physical key shapes (frustums). Thus, the *press-able* area of a key is larger than the displayed key – roughly doubling the true area of the key. One might consider the 2D key graphic to represent the near-plane of the corresponding 3D key frustum, and the invisible bounding volume the far-plane.

The native resolution of the combined virtual keyboard graphic is 1650x586 pixels (17.19"x6.11" at 96 pixels/inch) and is centered on the screen. When displayed on the 22" 3M M2256PW, at a resolution of 1680x1050 pixels, the virtual keyboard is approximately 18.25"x6.375" – marginally larger than the 17.375"x6.25" size of the Dell SK-8135.

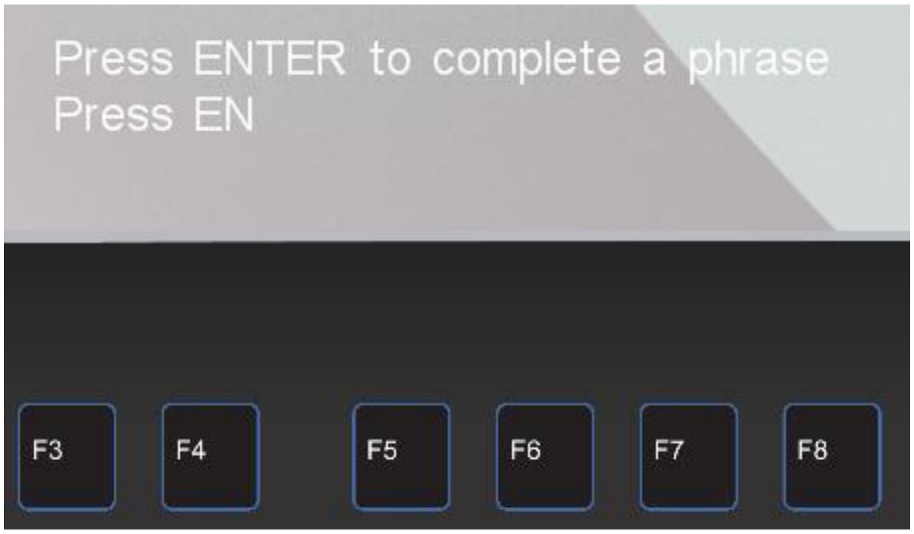
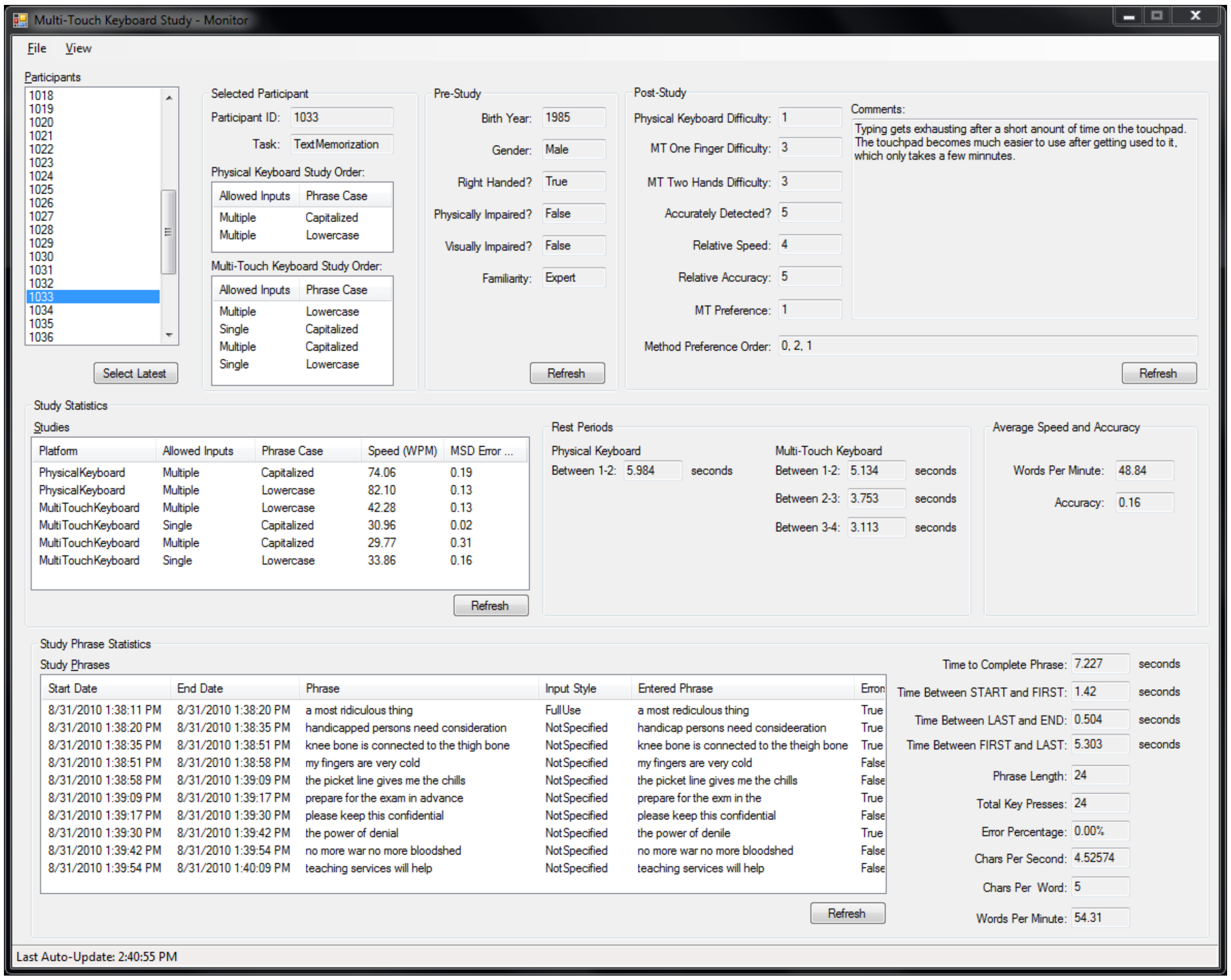

Press ENTER to complete a phrase
Press EN

Figure 4 Close-up of the virtual keyboard software depicting the presented and transcribed text

As previously mentioned, participants were asked to transcribe presented phrases. Depicted in Figure 4, these phrases were displayed above the virtual keyboard, with the participant's transcribed phrase just below the presented phrase. A traditional flashing cursor denoted the participant's "input location" as their typed characters were presented on the screen. Each key stroke in the input stream was time-stamped and recorded to a database. Participant’s completed each phrase by pressing the *Enter* key.

Pictured in Figure 5, a custom monitoring application allowed the experimenter to observe a participant’s progress in real-time. Additionally, this application was used to record the participant’s input style for a particular device or phrase. The notion of input style is described in more detail in Section 3.2.3.



Participants

1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036

Selected Participant

Participant ID:	1033
Task:	TextMemorization

Physical Keyboard Study Order:

Allowed Inputs	Phrase Case
Multiple	Capitalized
Multiple	Lowercase

Multi-Touch Keyboard Study Order:

Allowed Inputs	Phrase Case
Multiple	Lowercase
Single	Capitalized
Multiple	Capitalized
Single	Lowercase

Pre-Study

Birth Year:	1985
Gender:	Male
Right Handed?	True
Physically Impaired?	False
Visually Impaired?	False
Familiarity:	Expert

Post-Study

Physical Keyboard Difficulty:	1
MT One Finger Difficulty:	3
MT Two Hands Difficulty:	3
Accurately Detected?	5
Relative Speed:	4
Relative Accuracy:	5
MT Preference:	1
Method Preference Order:	0, 2, 1

Comments:
Typing gets exhausting after a short amount of time on the touchpad. The touchpad becomes much easier to use after getting used to it, which only takes a few minnutes.

Study Statistics

Studies

Platform	Allowed Inputs	Phrase Case	Speed (WPM)	MSD Error ...
PhysicalKeyboard	Multiple	Capitalized	74.06	0.19
PhysicalKeyboard	Multiple	Lowercase	82.10	0.13
MultiTouchKeyboard	Multiple	Lowercase	42.28	0.13
MultiTouchKeyboard	Single	Capitalized	30.96	0.02
MultiTouchKeyboard	Multiple	Capitalized	29.77	0.31
MultiTouchKeyboard	Single	Lowercase	33.86	0.16

Rest Periods

Physical Keyboard
Between 1-2: 5.984 seconds

Multi-Touch Keyboard
Between 1-2: 5.134 seconds
Between 2-3: 3.753 seconds
Between 3-4: 3.113 seconds

Average Speed and Accuracy

Words Per Minute: 48.84
Accuracy: 0.16

Study Phrase Statistics

Study Phrases

Start Date	End Date	Phrase	Input Style	Entered Phrase	Error
8/31/2010 1:38:11 PM	8/31/2010 1:38:20 PM	a most ridiculous thing	FullUse	a most rediculous thing	True
8/31/2010 1:38:20 PM	8/31/2010 1:38:35 PM	handicapped persons need consideration	NotSpecified	handicap persons need consideeration	True
8/31/2010 1:38:35 PM	8/31/2010 1:38:51 PM	knee bone is connected to the thigh bone	NotSpecified	knee bone is connected to the theigh bone	True
8/31/2010 1:38:51 PM	8/31/2010 1:38:58 PM	my fingers are very cold	NotSpecified	my fingers are very cold	False
8/31/2010 1:38:58 PM	8/31/2010 1:39:09 PM	the picket line gives me the chills	NotSpecified	the picket line gives me the chills	False
8/31/2010 1:39:09 PM	8/31/2010 1:39:17 PM	prepare for the exam in advance	NotSpecified	prepare for the exm in the	True
8/31/2010 1:39:17 PM	8/31/2010 1:39:30 PM	please keep this confidential	NotSpecified	please keep this confidential	False
8/31/2010 1:39:30 PM	8/31/2010 1:39:42 PM	the power of denial	NotSpecified	the power of denile	True
8/31/2010 1:39:42 PM	8/31/2010 1:39:54 PM	no more war no more bloodshed	NotSpecified	no more war no more bloodshed	False
8/31/2010 1:39:54 PM	8/31/2010 1:40:09 PM	teaching services will help	NotSpecified	teaching services will help	False

Time to Complete Phrase:	7.227	seconds
Time Between START and FIRST:	1.42	seconds
Time Between LAST and END:	0.504	seconds
Time Between FIRST and LAST:	5.303	seconds
Phrase Length:	24	
Total Key Presses:	24	
Error Percentage:	0.00%	
Chars Per Second:	4.52574	
Chars Per Word:	5	
Words Per Minute:	54.31	

Last Auto-Update: 2:40:55 PM

Figure 5 Real-time experiment monitoring application

3.2.1.4 Procedure

Participants initially performed two sets of ten phrases on the physical keyboard platform. They then moved to the multi-touch platform and completed four sets of ten phrases. On each platform, they performed three warm-up phrases to familiarize themselves with the platform and task. Participants were instructed to type “as quickly as possible as accurately as possible”. The arrow keys, *Delete*, *Home,* and *End* keys were disabled, leaving *Backspace* as the only available correction key.

Each participant’s 60 total phrases were randomly selected from a set of 465 phrases. This set was derived from MacKenzie and Soukoreff’s database of 500 phrases [45]. The 35 removed phrases contained the word “I” and were removed so as not to confound the examination of capitalized text.

After finishing the experiment, participants completed a post-study questionnaire.

3.2.1.5 Environment

The environment of the study is pictured in Figure 6: a small room housing the physical and multi-touch platforms. Note the difference in chairs between the two platforms as well as the adjustable orientation of the multi-touch platform. Seating was available behind these workstations from which the experimenter could observe the participant.

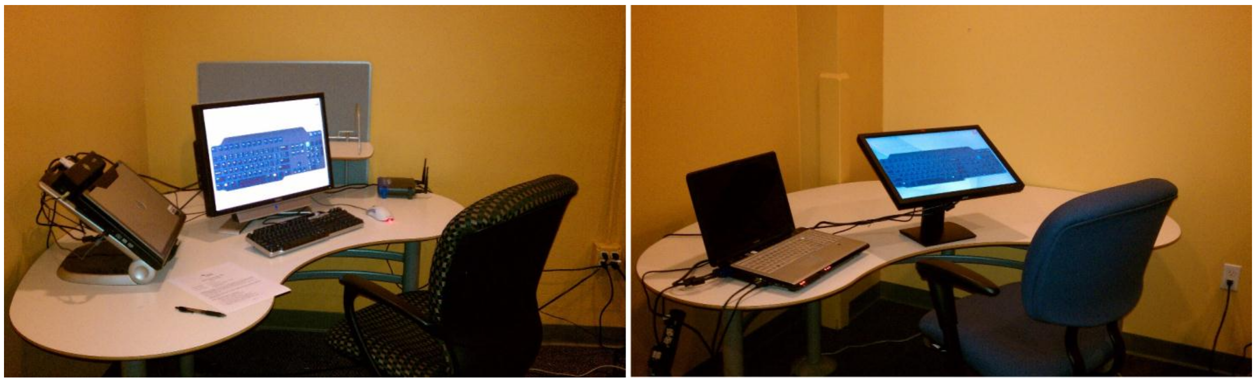


Figure 6 Text entry study environment: physical platform (left) and multi-touch platform (right)

3.2.1.6 Design

The first aspect of this experiment employs a between-subjects single-factor design, where the factor is *Task*, with levels: *Text-Copy* and *Text-Memorization*. This replicates a similar study [69], where the presented phrase is either always displayed (copy) or disappears once the participant begins typing (memorization). Participants were randomly assigned a task with an equal number of participants (twenty-four) for each task.

For both tasks, the experiment uses a within-subjects multi-factor design. The three factors are: *Platform*, *Allowed Inputs*, and *Phrase Case*; and each factor has two levels. The two platforms are *Physical Keyboard* and *Multi-Touch Keyboard*.

The *Allowed Inputs* factor denotes the number of simultaneous contact points allowed by the platform. For the physical keyboard platform, this is always *Multiple,* while the multi-touch platform has two operating modes: *Multiple* or *Single*. A single allowed input on the multi-touch platform forces the device to accept only one point of simultaneous contact. Thus, chorded-key input is disabled on the multi-touch platform in single-allowed-input mode. To produce a capitalized letter in this mode, one must press the caps-lock key, then press the character key, then press the caps-lock key again to resume lowercase typing. We omitted single-allowed-input mode from the physical keyboard platform as typical physical keyboards have no such notion. The *Phrase Case* factor includes *Lowercase* and *Mixed-Case* levels. The first level indicates that all characters within the phrase are lowercase. *Mixed-Case* signifies that some of the letters may be capitalized. The mixed-case phrases were generated from the all-lowercase database of 465 phrases. We took a different approach for capitalizing letters or words between the *Text-Copy* and *Text-Memorization* tasks. For the copy task, we randomly capitalized either one or two words from the phrase. For the selected word or words, we randomly capitalized the first letter of the word or the entire word. For the memorization task, only the first letter of the phrase was capitalized. We chose this route, as we felt the additional cognitive load of the memorization task prohibited unnatural capitalization. The resulting capitalized phrases were stored in the phrase database. Thus, two participants who were presented the same phrase would have the same capitalization. The final phrase database has an average of 4.22 capitalized characters per phrase for the copy task, and exactly 1 capitalized character per phrase for the memorization task. No punctuation characters were included in any of the phrases.

The *Allowed Inputs* and *Phrase Case* conditions were administered in a balanced manner. However, participants always operated on the physical keyboard platform first. This allowed the (presumably) experienced physical keyboard typist to become familiar with the task before moving to the multi-touch platform. For both platforms (physical and multi-touch), we constructed pairings for the Allowed Inputs and Phrase Case factors. Specifically, the physical keyboard had pairings { Multiple Inputs, Lowercase } and { Multiple Inputs, Mixed Case } identified as pairings 0 and 1, respectively. Recall, that the physical keyboard did not support the notion of single allowed inputs, thus we have two pairings rather than four. The multi-touch platform had pairings { Single Input, Lowercase } , { Multiple Inputs, Lowercase }, { Single Input, Mixed Case } and { Multiple Inputs, Mixed Case } identified as pairings 0, 1, 2, and 3, respectively. To balance these pairings, we built two Latin Squares from the pairing identifiers – one set of orderings for each platform. All 48 participants performed their text entry tasks on both platforms. Thus 24 participants used pairing 0 first, for the physical keyboard, and 24 used pairing 1 first. For the multi-touch platform, with four pairings, 12 participants performed each of the pair orderings. In this fashion, we counterbalanced the conditions to mitigate learning effects. Furthermore, we analyzed the collected data, by condition order, and found no detectable learning effects.

Each participant completed the experiment in approximately 30 minutes.

3.2.2 Metrics

Between the presented phrase and the transcribed input stream, we are able to derive a surprising amount of information. As previously mentioned, speed and accuracy are the chief performance measurements, but other metrics such as *conscientiousness* and *key strokes per character* can offer important insights into the text entry system under examination.

Speed is measured initially in characters per second (CPS). This is the number of characters in the presented phrase divided by the elapsed transcription time; where the elapsed time is measured between the first and last key strokes. In this experiment, we allow the participant to rest between phrases. Thus, speed timing does not begin until the participant performs the first key stroke after the phrase is presented (also allowing us to measure the associated rest period). Correspondingly, we disregard the time between the final key stroke in the phrase and the *Enter* press which commits the phrase, thus measuring the elapsed time as the true first and last key strokes in the phrase. Speed is commonly reported as *words per minute* (WPM) through a simple transformation:

$$WPM = \frac{CPS}{CPW} * 60 \tag{3.1}$$

where CPW (characters per word) is the traditional five characters per word [94].

Accuracy is a more complicated measurement, though at first glance it might not seem to be. Borrowing an example from Soukoreff and MacKenzie [68], consider the following:

```
quick brown fox         (presented text)
quixck brwn fox         (transcribed text)
```

How many transcription errors were made in this example? A character-by-character analysis suggests six errors. However, a human might look at the example and determine than only two errors were made – an inserted ‘x’ and an omitted ’o’. In 2001, Soukoreff and MacKenzie [68] proposed a technique, based on the Levenshtein minimum string distance (MSD), to measure text entry error rates through the optimal set of primitive operations (insertion, deletion, and substitution) required to transform one string into another. They refined this work a year later [43], introducing the notion of optimal alignment strings to help determine the actual errors committed. The resulting equation for error rate is:

$$New\ MSD\ Error\ Rate = \frac{MSD(P,T)}{\overline{S_A}}\ x\ 100\% \tag{3.2}$$

where $\overline{S_A}$ is the mean length of the set of optimal alignment strings.

In 2003 and 2004, Soukoreff and MacKenzie [69, 70] developed a taxonomy for constituents of the text input stream, using the optimal alignment strings and the input stream to identify key strokes as either Correct (C), Incorrect Not Fixed (INF), Incorrect Fixed (IF), or Fix Keys (F). They used these classifications to develop a new metric for error rate:

$$Total\ Error\ Rate = \frac{INF + IF}{C +\ INF + IF}\ x\ 100\% \tag{3.3}$$

The total error rate can be further split into *Corrected Error Rate* and *Not Corrected Error Rate*:

$$Corrected\ Error\ Rate = \frac{IF}{C +\ INF + IF}\ x\ 100\% \tag{3.4}$$

$$Not\ Corrected\ Error\ Rate = \frac{INF}{C + \ INF + IF}\ x\ 100\% \tag{3.5}$$

In their 2003 work, they also introduced a measure of participant conscientiousness, which represents the ratio of corrected errors to total errors.

$$Conscientiousness = \frac{IF}{INF + IF} \tag{3.6}$$

Throughout these efforts, Soukoreff and MacKenzie discuss the notion of *Key Strokes per Character (KSPC)*. Consider a scenario where the error rate is zero (as calculated through the MSD). This does not mean that the participant did not commit any errors during transcription – they may have, but corrected them before completing the phrase. Thus, the KSPC statistic can provide insight into the number of key strokes required to produce the resultant phrase. The original equation for KSPC is:

$$KSPC = \frac{|InputStream|}{|TranscribedText|} \tag{3.7}$$

Thus, a KSPC of 1.0 (on a keyboard that required single key strokes for all text entry) would indicate that no errors were made during transcription. Soukeroff and MacKenzie [69] refined this equation using the input stream constituents to:

$$New\ KSPC \approx \frac{C + INF + IF + F}{C + INF} \tag{3.8}$$

We employed all of these metrics within this study, though we have omitted reporting the older MSD Error Rate and KSPC metrics in favor of their newer and more accurate counterparts. Furthermore, we have used these metrics for each of the studies discussed in chapters 4 through 6.

3.2.2.1 Post-Study Questionnaire

In addition to the quantitative measures just described, we asked each of the participants to respond to the following statements in a post-study questionnaire:

- Rate the difficulty of entering text on the physical keyboard (scale: 1-5)
- Rate the difficulty of entering text on the multi-touch platform in single-touch mode (scale: 1-5)
- Rate the difficulty of entering text on the multi-touch platform in multi-touch mode (scale: 1-5)
- The multi-touch platform accurately detected my intended key presses (scale: 1-5)
- Rate the relative speed of the two input techniques on the multi-touch platform (scale: 1-7)
- Rate the relative accuracy of the two input techniques on the multi-touch platform (scale: 1-7)

- Rate your preference of the two input techniques on the multi-touch platform (scale: 1-7)
- Rank your preferred text entry method as (1, 2, 3; 1 highest)
- Would you use the multi-touch device to enter text for everyday computer activities?

3.2.3 Results

In total, participants entered 2,880 phrases: 1,440 for each task; 60 for each participant; 10 for each within-subjects condition excluding the *Single Allowed Input* mode from the physical keyboard platform. The means and standard deviations for each condition and metric are summarized in Table 3. To analyze the quantitative data, we ran a 3 (Platform/Allowed Input) by 2 (Phrase) by 2 (Task) repeated measures mixed ANOVA on each metric. In addition, we ran a post-hoc analysis on significant factors using t-tests. To analyze the subjective data from the post-questionnaires, we ran Friedman tests, followed by post-hoc analysis using Wilcoxon Signed Rank tests. We used Holm's sequential Bonferroni adjustment [30] to control for Type I errors for the post-hoc tests.

Table 3 Text entry study summary of results

								Total		Not Corrected		Corrected			
		Allowed	Phrase	WPM		New KSPC		Error Rate (%)		Error Rate (%)		Error Rate (%)		Conscientiousness	
Task	Platform	Inputs	Case	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Copy	Physical	Multiple	Lowercase	72.26	17.86	1.09	0.10	3.73	3.53	0.25	0.29	3.48	3.44	0.87	0.18
Copy	Physical	Multiple	MixedCase	59.12	16.04	1.14	0.11	5.94	3.88	0.42	0.83	5.52	3.87	0.89	0.14
Copy	MultiTouch	Single	Lowercase	39.57	10.83	1.24	0.21	9.82	6.99	0.69	1.15	9.13	7.17	0.84	0.21
Copy	MultiTouch	Multiple	Lowercase	39.58	8.20	1.30	0.16	11.93	5.21	0.61	0.74	11.31	5.37	0.89	0.13
Copy	MultiTouch	Single	MixedCase	28.79	6.42	1.24	0.13	9.89	4.87	0.62	0.92	9.27	4.37	0.91	0.11
Copy	MultiTouch	Multiple	MixedCase	30.20	7.11	1.36	0.17	14.26	5.30	0.64	0.74	13.62	5.33	0.93	0.10
Mem	Physical	Multiple	Lowercase	80.53	14.32	1.07	0.06	4.11	2.70	0.99	1.40	3.13	2.51	0.76	0.29
Mem	Physical	Multiple	MixedCase	73.48	12.23	1.09	0.07	5.53	4.03	1.79	2.14	3.74	2.85	0.68	0.22
Mem	MultiTouch	Single	Lowercase	42.05	9.71	1.18	0.14	9.42	5.26	2.49	2.42	6.94	4.79	0.72	0.21
Mem	MultiTouch	Multiple	Lowercase	46.46	11.04	1.24	0.17	11.42	6.22	2.35	2.15	9.07	5.84	0.72	0.21
Mem	MultiTouch	Single	MixedCase	34.48	7.97	1.20	0.14	10.49	5.51	2.80	3.04	7.69	4.68	0.75	0.19
Mem	MultiTouch	Multiple	MixedCase	39.18	9.29	1.29	0.21	13.61	7.37	2.67	2.66	10.94	6.83	0.75	0.17

3.2.3.1 Between Subjects Data

We analyzed each metric on the between-subjects condition *Task* and found significant differences for WPM ($F_{1,46} = 9.94$, $p < 0.05$), Not Corrected Error Rate ($F_{1,46} = 28.59$, $p < 0.05$), and Conscientiousness ($F_{1,43} = 24.15$, $p < 0.05$). This indicates that participants typed faster when recalling the phrase from memory than when the phrase was constantly displayed. And while participants committed a similar number of errors for both tasks, they were more conscientious about fixing them for the copy task. These results support the original study by Soukoreff and MacKenzie [69] and suggest that these effects generalize across text entry technologies.

3.2.3.2 Within Subjects Data

We analyzed the data, within each task, across the *Platform, Allowed Inputs,* and *Phrase Case* factors with a repeated measures mixed ANOVA. For WPM, we found significant differences for *Platform* ($F_{2,92} = 238.59$, $p < 0.05$) and *Phrase Case* ($F_{1,46} = 246.18$, $p < 0.05$). For New KSPC, we found significant differences for *Platform* ($F_{2,92} = 46.66$, $p < 0.05$) and *Phrase Case* ($F_{1,46} = 8.11$, $p < 0.05$). For Total Error Rate, we found significant differences for *Platform* ($F_{2,92} = 52.54$, $p < 0.05$) and *Phrase Case* ($F_{1,46} = 14.61$, $p < 0.05$). For Not corrected Error Rate, we found significance for just *Platform* ($F_{2,92} = 5.46$, $p < 0.05$). Finally, for Corrected Error Rate, we found significant differences for *Platform* ($F_{2,92} = 52.45$, $p < 0.05$) and *Phrase Case* ($F_{1,46} = 10.55$, $p < 0.05$).

The results of our post-hoc analysis, with seven comparison pairs for each metric, are presented in Table 4. Metrics that were found to be statistically significant are marked in bold. Significant results were found for WPM, New KSPC, Total Error Rate, and Corrected Error Rate; generally implying that typing is faster on the physical platform, with fewer errors, and when the phrase is presented in lowercase.

Table 4 Text entry study within-subjects t-test results

Task	Platform	Allowed Inputs	Phrase Case	WPM		New KSPC		Total Error Rate (%)		Not Corrected Error Rate (%)		Corrected Error Rate (%)		Conscientious.	
				t_{23}	p-value	t_{23}	p-value	t_{23}	p-value	t_{23}	p-value	t_{23}	p-value	t_{23}	p-value
Copy	P-To-P	M-To-M	L-To-M	9.38	**$p < 0.007$**	-2.97	**$p < 0.013$**	-3.25	**$p < 0.013$**	-0.92	$p = 0.366$	-3.15	**$p < 0.013$**	-0.44	$p = 0.668$
Copy	M-To-M	S-To-S	L-To-M	6.19	**$p < 0.008$**	-0.03	$p = 0.979$	-0.05	$p = 0.957$	0.28	$p = 0.785$	-0.11	$p = 0.911$	-2.00	$p = 0.057$
Copy	M-To-M	M-To-M	L-To-M	7.57	**$p < 0.010$**	-1.51	$p = 0.145$	-1.99	$p = 0.058$	-0.16	$p = 0.878$	-1.99	$p = 0.059$	-1.32	$p = 0.200$
Copy	P-To-M	M-To-M	L-To-L	8.83	**$p < 0.013$**	-7.12	**$p < 0.007$**	-7.53	**$p < 0.007$**	-2.23	$p = 0.036$	-7.06	**$p < 0.008$**	-0.40	$p = 0.694$
Copy	P-To-M	M-To-M	M-To-M	8.74	**$p < 0.017$**	-6.14	**$p < 0.008$**	-7.05	**$p < 0.008$**	-1.04	$p = 0.308$	-7.14	**$p < 0.007$**	-1.30	$p = 0.208$
Copy	M-To-M	S-To-M	L-To-L	-0.01	$p = 0.076$	-1.66	$p = 0.110$	-1.52	$p = 0.143$	0.34	$p = 0.734$	-1.60	$p = 0.123$	-1.32	$p = 0.201$
Copy	M-To-M	S-To-M	M-To-M	-1.86	$p = 0.996$	-4.47	**$p < 0.010$**	-5.17	**$p < 0.010$**	-0.12	$p = 0.907$	-4.97	**$p < 0.010$**	-0.62	$p = 0.539$
Mem	P-To-P	M-To-M	L-To-M	3.95	**$p < 0.025$**	-1.17	$p = 0.255$	-2.25	$p = 0.034$	-1.85	$p = 0.077$	-1.13	$p = 0.271$	1.16	$p = 0.258$
Mem	M-To-M	S-To-S	L-To-M	5.92	**$p < 0.013$**	-0.98	$p = 0.335$	-1.30	$p = 0.206$	-0.57	$p = 0.575$	-1.10	$p = 0.282$	-0.80	$p = 0.435$
Mem	M-To-M	M-To-M	L-To-M	6.87	**$p < 0.010$**	-2.28	$p = 0.032$	-2.31	$p = 0.030$	-0.52	$p = 0.610$	-2.17	$p = 0.041$	-0.65	$p = 0.521$
Mem	P-To-M	M-To-M	L-To-L	10.70	**$p < 0.008$**	-4.27	**$p < 0.008$**	-5.14	**$p < 0.007$**	-2.89	$p = 0.008$	-4.53	**$p < 0.008$**	0.35	$p = 0.729$
Mem	P-To-M	M-To-M	M-To-M	12.30	**$p < 0.007$**	-4.61	**$p < 0.007$**	-4.99	**$p < 0.008$**	-1.39	$p = 0.177$	-5.14	**$p < 0.007$**	-1.90	$p = 0.070$
Mem	M-To-M	S-To-M	L-To-L	-3.05	**$p < 0.050$**	-2.04	$p = 0.053$	-2.06	$p = 0.051$	0.25	$p = 0.803$	-2.33	$p = 0.029$	-0.04	$p = 0.965$
Mem	M-To-M	S-To-M	M-To-M	-4.07	**$p < 0.017$**	-3.11	**$p < 0.010$**	-2.34	$p = 0.028$	0.17	$p = 0.868$	-3.53	**$p < 0.010$**	-0.27	$p = 0.786$

The permutations for *Platform* comparison are: Physical- To-Physical, MultiTouch-To-MultiTouch, and Physical-To-MultiTouch (abbreviated P-To-P, M-To-M, and P-To-M, respectively). *Allowed Inputs* are compared as: Multiple-To-Multiple, Single-To-Single, and Single-To-Multiple (abbreviated M-To-M, S-To-S, and S-To-M, respectively). *Phrase Case* comparisons are: Lowercase-To-MixedCase, Lowercase-To-Lowercase, and MixedCase-To-MixedCase (abbreviated L-To-M, L-To-L, and M-To-M, respectively). Note, that there are no comparisons that consider a single-allowed-input on the physical platform, as this mode was omitted from the experiment.

3.2.3.3 Input Style

We observed three distinct input styles during the experiment: *Hunt-and-Peck*, *Full-Use*, and *Hybrid.* Hunt- and-Peck describes those participants who type with exactly two fingers – typically the index fingers. Full-Use designates that the participant engaged both hands and all (or most) of their fingers in the typing task. Hybrid is somewhere between Hunt-and-Peck and Full-Use – this input style was rare and encompassed participants who were mostly hunt-and-peck but might have periodically engaged their middle fingers or thumbs. We observed the participants' input styles for each of their 60 phrases, allowing that their style might change between platforms and/or phrase conditions. All participants, across all conditions employed Full-Use with the physical keyboard. Hunt-and-Peck or Hybrid where only employed on the multi-touch platform and then only for 250 phrases (13%) and 80 phrases (4%), respectively. Thus participants predominately typed in the same style (Full-Use) on the physical and multi-touch platforms. There were no significant differences between these ratios across *Task*, *Allowed Inputs* and *Phrase Case* conditions.

3.2.3.4 Post-Study Questionnaire Data

In the post-study questionnaire, we asked each participant seven Likert-scale questions pertaining to: the difficulty of entering text; whether the multi-touch accurately detected intended key presses; and the speed and accuracy of text entry on the multi-touch platform. Figure 7 presents the results of the five-point Likert scale questions, where the value 1 indicates *Very Easy* and 5 for *Very Difficult* in relation to the "difficulty" questions, or *Strongly Disagree* and *Strongly Agree* for the "accurately detected" question.

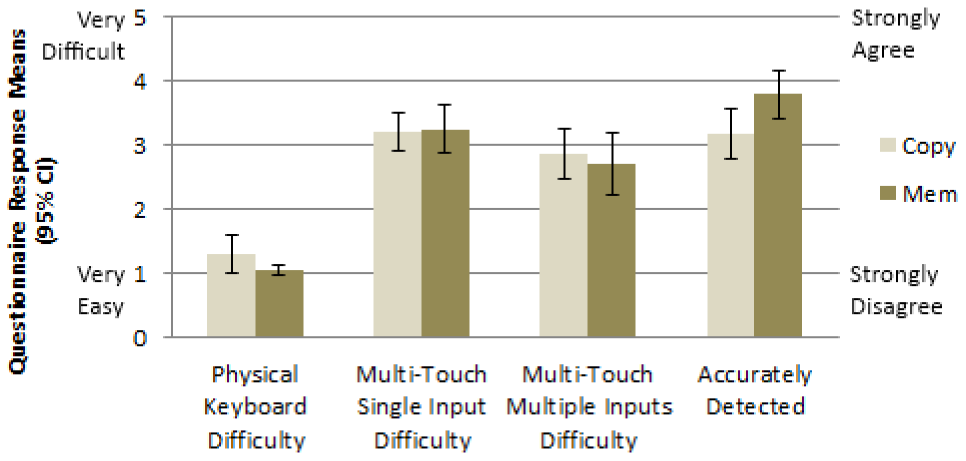


Figure 7 Text entry study questionnaire response means (five-point Likert)

To analyze the responses, we conducted Friedman tests on each of the questions. We found statistical significance on the "difficulty" questions at $\chi^2_2 =$

$29.792, p < 0.05$ for the *Copy* task and $\chi_2^2 = 36.167, p < 0.05$ for the *Memorization* task. Post-hoc analysis using Wilcoxon Signed Rank tests revealed multi-touch-single-input to be significantly more difficult than the physical keyboard (Copy: Z=-4.08, p < 0.017, Mem: Z=-4.27, p < 0.017) and likewise for multi-touch-multiple-inputs (Copy: Z=-3.81, p < 0.025, Mem: Z=-3.97, p < 0.025). Results were split between the Copy and Memorization tasks for multi-touch multiple-input-to-single-input difficulty (Copy: Z=-1.06, p = 0.109, Mem: Z=-2.50, p < 0.050). Figure 8 presents the responses to questions concerning the participant's perception of the relative speed and accuracy, and their preference of the multi-touch platform considering the two *Allowed Inputs* levels. These questions employed a seven-point Likert-scale, where smaller values indicated better speed and accuracy, or higher preference, for the multi-touch platform using multiple simultaneous inputs.

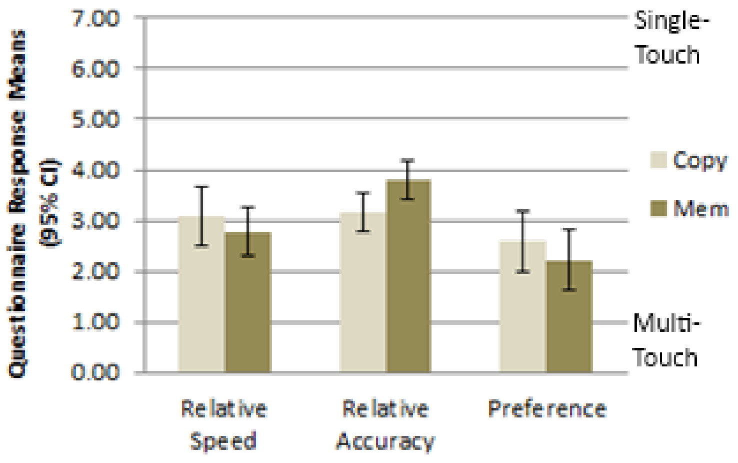


Figure 8 Text entry study questionnaire response means (seven-point Likert)

We also asked participants to rank their preferred text entry method between: the physical keyboard; multi-touch with only a single allowed simultaneous input; and multi-touch with multiple simultaneous inputs. These results are presented in Figure 9. The results clearly show a ranking of Physical Keyboard, Multi-Touch (Multiple Inputs) and Multi-Touch (Single Inputs) in order of most to least preferred.

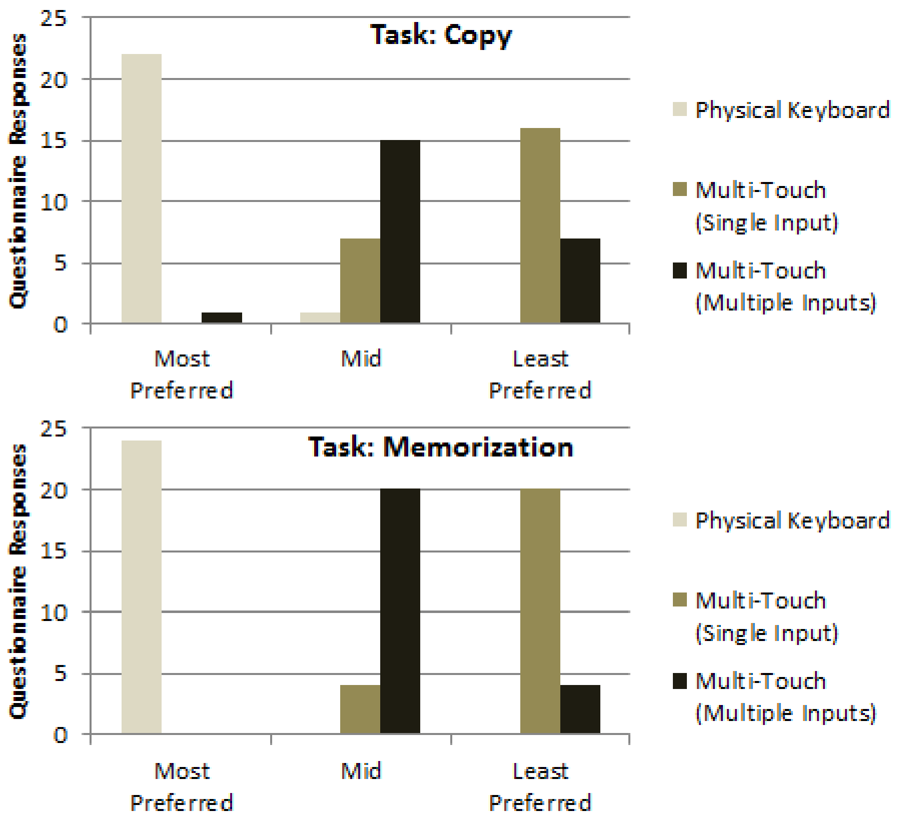


Figure 9 Text entry study questionnaire responses on method preference

The final subjective question asked if the participant "would use the multi-touch device to enter text for everyday computer activities." The responses to this question are presented in Figure 10.

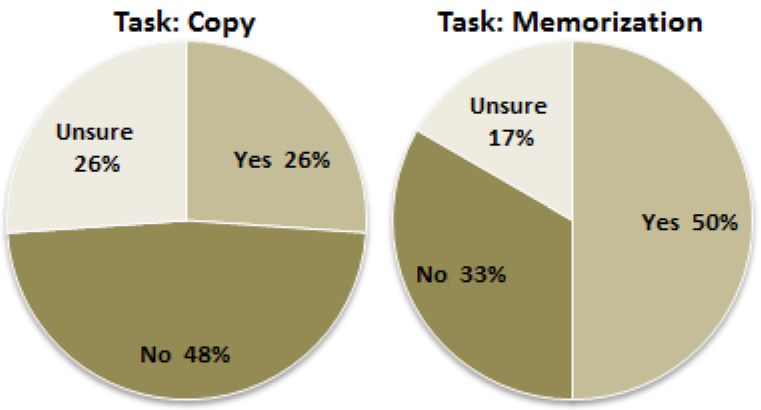


Figure 10 Text entry study questionnaire responses on "would use" multi-touch platform to enter text for everyday computer activities

3.2.4 Discussion

The results of this experiment are not surprising. Text entry on the physical keyboard is faster than the multi-touch platform and has a lower total error rate. Typing in lowercase is faster than mixed-case and generally has a lower total error rate (though there was no statistical significance due to the Holm's sequential Bonferroni adjustment). Typing on the multi-touch platform allowing multiple simultaneous inputs is generally faster than with only a single input – though statistical significance for this comparison was split between the copy and memorization tasks. More key strokes per character were required when typing on the multi-touch keyboard; which corresponds with the higher total error and corrected error rates. Conscientiousness remained consistent between the two platforms. This is interesting because, while participants made more errors on the multi-touch platform, they were not disinclined to fix these errors – even though they considered the multi-touch platform to be more difficult to use than the physical keyboard (as per the post-study questionnaire results).

So while we intuitively expected the physical keyboard to outperform the multi-touch platform; our question was always “by how much?” And, by extension, “how good is good enough for adoption as a general-purpose computing platform?” Certainly, if the performance of the multi-touch platform was greater-than-or-equal to the physical platform, one could consider the device “adoptable” from the perspective of performance. While we are not certain what that adoption-performance-lower-bound is (or that we haven’t already passed it), we have a clear upper-bound and a goal to strive for. The results of this experiment establish a baseline from which to evaluate future efforts in text entry on a multi-touch platform.

Furthermore, our participants were all very competent typists and performed above expert typing speeds of 56 WPM [57] on the physical keyboard, while this was their first experience with this multi-touch platform. A longitudinal study would better establish expert-level performance on the multi-touch platform and is a topic of future work.

Though the performance evidence leans heavily in favor of the physical keyboard, the post-study questionnaire responses offer support for the usefulness of the multi-touch platform as a text entry device. 18 of 47 participants (38%) stated that they would use the multi-touch platform to enter text for everyday computer activities. This is only 2% lower (19 participants) than those that explicitly said they would not use the platform. While some of this support may be ascribed to the novelty of the device, it is not entirely without merit.

We call attention to the comparison between single- and multiple-allowed-inputs on the multi-touch platform. Again, we expected participants to perform better with multiple allowed inputs and for preference to follow suit. The results generally matched this expectation but only by a thin margin and statistical significance was scattered amongst the metrics. Intuitively, one would expect typing to be faster and easier when allowed to use both hands in concert. Moreover a reasonable expectation would be that multi-key input is better performed when multiple simultaneous key strokes are allowed. Indeed, chorded keys such as *Shift-A*, *Ctrl-C,* or *Ctrl-Alt-Delete* are challenging to perform, if not impossible, without concurrent key strokes. So why would the statistical data not show *far* superior data for this text entry mode? First, we note that the “single-touch” mode used within this experiment is not similar to that of a stylus – where the user must move the one “finger” across the virtual keyboard. Our multi-touch platform allows participants to orient their hands and fingers in a fashion similar to the physical keyboard – where both hands and all ten fingers might be engaged. The constraint is simply that the user cannot be *touching* the multi-touch device in more than one location concurrently. As mentioned previously, the input style was predominately *Full-Use* on the multi-touch platform and we believe this to be the primary reason behind the similar performance metrics between single- and multiple-allowed-inputs. A contributor to the *lower* performance of the multi-touch platform in single-touch mode is likely that participants were unable to rest their palms on the bottom portion of the screen without causing an unintended surface contact – forcing participants to “hover” over the multi-touch monitor. Although the virtual keyboard is the only portion of the screen that gives feedback when touched, the entire multi-touch display is interactive. In single-touch mode, resting your palms on the screen is still considered an interaction – the one-and-only allowed interaction – even though there’s no obvious feedback that you’re providing input. When multiple simultaneous inputs are allowed, such incidental surface interaction has no consequence. As a counterpoint to this observation, we note that a common cause of mistakes made in multiple-allowed-inputs mode is that users tend to “drag” their fingers across the screen to get to the next key. Against a physical keyboard such low-pressure key contact does not trigger a key press – but not so with the zero-pressure 3M capacitive multi-touch monitor. In single-touch mode, participants were more apt to pick up their fingers when proceeding to the next key.

Finally, we wish to point out the myriad comments made by participants and recorded during the post-study questionnaire. Specifically, we collected two sets of free-form comments: those in response to the “Would Use” question and its follow-up question: “Why or Why Not”; and “Other” feedback that the participants provided. These comments are itemized in Appendix D.1. A recurring theme within these comments is the lack of tactile feedback on the multi-touch platform. These responses, along with those from studies 2 through 4 helped direct our research and encouraged our exploration of audible and tactile feedback (discussed in Chapter 7).

3.3 Summary

In this work we have posed the question: “c*an multi-touch, without a text entry peripheral, provide a platform for efficient text entry?”* And, by extension, *is such a platform viable for general-purpose computing?* As a foundation to this effort, we have implemented and evaluated a virtual QWERTY keyboard on a multi-touch platform. We have presented the results of this evaluation which establish a baseline for text entry performance characteristics on a multi-touch device. These results clearly show that a physical keyboard outperforms the multi-touch platform, though post-study questionnaire data offers support for the potential adoption of the multi-touch device for everyday text entry tasks.

Our next study will attempt to improve on this performance by modifying *how* text entry is accomplished through a multi-touch platform.

3.4 Notes on Replication

This study cast a wide net – attempting to measure text entry performance for lowercase and mixed-case phrases, between platforms, with varying support of simultaneous input and by differing the task itself (copy versus memorization). We also analyzed the data through numerous metrics and examined the input style of the users. We felt that all of this was necessary because, to our knowledge, no other study has attempted to measure the text entry performance on a multi-touch platform. And while similar studies have been conducted to analyze, for example, *copy* versus *memorization* tasks [69], we were uncertain if those results would generalize across text entry technologies. Furthermore, we were not clear on *how* users would type on the device, and sought an understanding at this fundamental level.

With the aid of hindsight, we believe future replication can be significantly simplified and we offer these notes to aid in any such attempts. Indeed, we adopted many of the observations we are about to convey, for the additional experiments described in the next several chapters.

3.4.1 Copy versus Memorization

As stated in the results section, we found that participants typed faster when recalling from memory than when the phrase was constantly displayed. And while participants committed a similar number of errors for both tasks, they were more conscientious about fixing them for the copy task. These results support the original study by Soukoreff and MacKenzie [69]. Furthermore, we observed that for both of the values with statistical significance – speed and conscientiousness – the differences between the *copy* and *memorization* means were consistent. Specifically, the speed for the memorization task was always faster than that of the copy task. Likewise, users were always more conscientious during the copy task than during memorization. Interestingly, the average difference for both speed and conscientiousness, between the tasks, was 18%. However, we are not suggesting that one can merely assume an 18% difference between the two tasks and perform one task to derive approximations of the other. What we can confidently state, is that typing is faster when recalling from memory than when the phrase was constantly displayed; and users are more conscientious about fixing mistakes when they can readily compare their transcribed phrase to the original.

Moreover, we observed that it was easier to administer the copy task than memorization. Participants required less explanation of the copy task and generally appeared more confident in completing the experiment. Participants were commonly observed to audibly groan, or speak self-deprecatingly about their memory, when presented with the memorization task. Furthermore, on several occasions, participants would begin typing the phrase, thus dismissing the visual representation, and immediately forget what they were supposed to type. This required that the experimenter repeat the phrase audibly – which was only possible because of the real-time monitoring tool. The take away from this observation is that the experiment must provide a way to reconstitute a phrase, or to otherwise present another phrase without invalidating the participant's data. We concede that audibly repeating forgotten phrases introduces a confound in the data – though we believe the effects to be minor – and chose to accept this confound rather than discard an entire data set due to one phrase. In hindsight, we would have preferred a mechanism to discard the entire phrase and present a new one – thereby maintaining the integrity of the data. Thus, we conclude that the copy task, while perhaps not as natural as memorization, is easier to administer and has less potential for invalid data.

3.4.2 Phrase Case

There was little doubt that participants would type faster when the phrase-to-transcribe was composed entirely of lowercase letters – and indeed, the results affirmed this assumption. We chose to evaluate mixed-case phrases to establish baseline measurements for a variety of text entry modes. However, we recognize a number of flaws in the examination of mixed-case text entry. First, as we described in Section 3.2.1.6, we had different approaches for capitalizing words between the copy and memorization tasks. The final phrase database had an average of 4.22 capitalized characters per phrase for the copy task, and exactly 1 capitalized character per phrase for the memorization task. This almost certainly accounts for the difference in performance increase, between the copy (30%) and memorization tasks (17%), as the mixed-case memorization phrase was almost entirely lowercase. And while it is natural to capitalize the first letter in a sentence – our rationale for capitalizing this letter for the memorization task – being the first character to transcribe makes it somewhat suspect from a perspective of accuracy or speed. While cognitively more difficult for the participants, it may have been better to have used the same capitalization approach for the memorization task as for the copy task. Thus we submit that the copy task is likely more valid as a baseline measurement for entering mixed-case text on a multi-touch platform than the memorization task, though the usefulness of either measurement is subject to the intent of the experimenter.

3.4.3 Simultaneous Inputs

The 3M M2256PW [3] multi-touch monitor does not natively support the notion of single-touch interaction. We wrote a custom .NET wrapper (described in the next section) around the 3M multi-touch software development kit (SDK) which allowed us to: 1) consume the 3M multi-touch monitor events from within a C# application; and 2) force the device to behave as a single-touch input device. Specifically, the 3M multi-touch SDK sends events for each interaction point, and our wrapper associates such events with a unique “tracking” identifier. To force the device to send only one point of simultaneous interaction was a simple matter of suppressing events for more than one active identifier.

We created this mode in order to evaluate differences between single-touch and multi-touch monitors. Single-touch monitors have been popular for many years, though they have not made major inroads towards general-purpose computing. We asked the question: “would users prefer single-touch or multi-touch input with respect to text-entry?” Our results clearly indicate that our participants preferred multi-touch input and, therefore, our additional experiments do not include evaluations of the multi-touch platform restricted to single simultaneous inputs.

3.4.4 3M Multi-Touch SDK & .NET Wrapper

The 3M multi-touch software development kit (SDK) is a set of Microsoft Windows libraries written using the C programming language. This is a consideration for multi-touch experimenters who use software that does not natively consume C language libraries. As mentioned, the software we developed for our experiments was written using the C# programming language, which does not natively interoperate with C language constructs. Moreover, the 3M multi-touch SDK is not object-oriented and, to our thinking, is not terrifically “programmer friendly”. Thus, we developed a .NET wrapper, around the 3M multi-touch SDK, using C++/CLI. CLI stands for the Common Language Infrastructure, and is a specification that defines the core of the .NET environment (Microsoft’s recent programming environment within which C# operates). C++/CLI is a C++ programming language extension that allows for unmanaged C and C++ code to interoperate with managed C++ code. In short, C++/CLI libraries allow developers to bridge C/C++ libraries with .NET libraries – exposing unmanaged C/C++ constructs to .NET and .NET constructs to unmanaged C/C++. In this fashion, we created a C++/CLI wrapper around the 3M multi-touch SDK which, from the perspective of our C# applications, appears as a native .NET assembly.

In addition to the notion of consuming the 3M multi-touch SDK from other programming environments, we wish to point out the nature of the 3M Windows device drivers and the need for the 3M multi-touch SDK in the first place. 3M has provided a Windows 7 mouse emulator and support for Windows 7's multi-touch functionality. In particular, Windows 7 provides a multi-touch keyboard which can be used without special software development. One needs merely plugin in the 3M M2256PW, via Universal Serial Bus (USB), to their Windows 7 device and the virtual keyboard and mouse *should* operate via the touch screen. However, the Windows 7 keyboard and mouse emulators are not customizable, and do not provide any functionality that is not present with a traditional keyboard and mouse. Multi-touch gestural interaction, for example, is not available through the 3M Windows 7 drivers. Multi-touch experimenters will likely require lower-level access to the multi-touch events, which to our knowledge (and according to the 3M documentation) is not possible without the use of the 3M multi-touch SDK. This was our need, and is the reason why we have developed the C++/CLI .NET wrapper. Please note, it may be possible to access the 3M multi-touch device through the Windows 7 multi-touch Application Programming Interface (API). We have not investigated this.

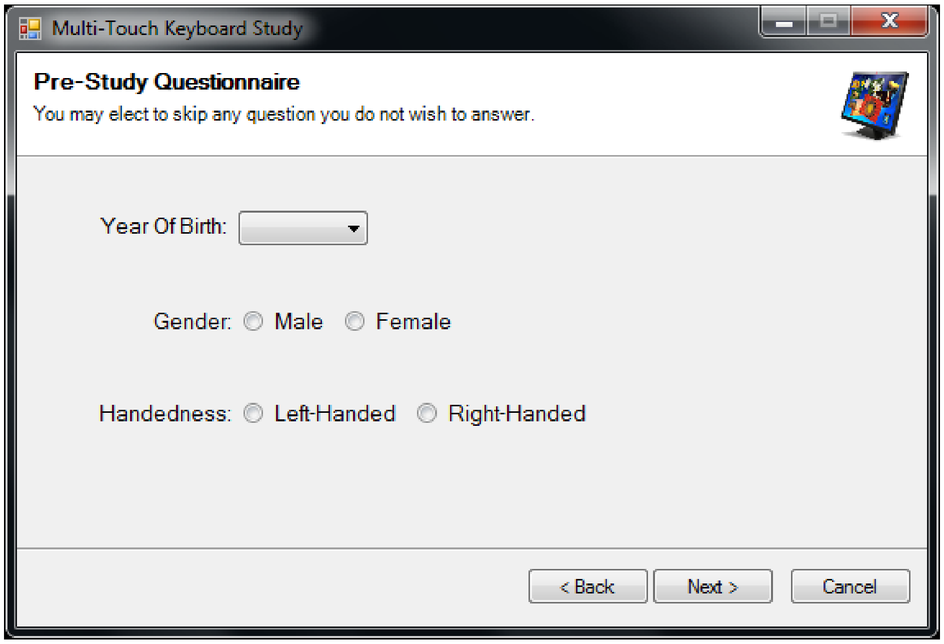
3.4.5 Choice of Platform

We are uncertain if the results found in this study will generalize across devices or multi-touch technologies. Indeed, such testing is planned in our future efforts. To comment on the choice of multi-touch platform for this study, we had three form-factors (that have made substantial inroads in commercial development/adoption) to choose from: tabletop, monitor, or hand-held/mobile. Of these, the tabletop form-factor was rejected because its design focused more on accommodating multiple simultaneous users than that of a single user. We rejected mobile devices such as the Apple iPad [5] or Smart Phones because of their smaller size, when compared to a traditional keyboard. Moreover, these devices are arguably less suitable for general-purpose computing than a laptop or desktop computer. The iPad is a platform that comes close to the capability of a desktop or laptop computer, but again, its size compared to a physical keyboard was the compelling factor to choose the 3M 22” multi-touch monitor over the iPad. The choice of the 3M device was also influenced by: its excellent capability at detecting simultaneous multi-touch interaction (20+ points of contact); form-factor (not too small or too large); adjustable orientation; relatively low-cost; availability; and ease of software development. It is our belief that an adjustable-orientation monitor-style device, of reasonably large size, is a decent representation of multi-touch devices that emphasizes individual use – the focus of our exploration in multi-touch and its application to general-purpose computing. Recall, from Section 3.2.1.2, that the 3M multi-touch monitor allows its orientation to be adjusted from 0-degrees (fully horizontal) to 90-degrees (fully vertical). It is our belief that an orientation of approximately 30-degrees serves as an ergonomic orientation that will benefit most users.

Also, with respect to the platform, we draw attention to the ergonomics of the desk and chair. In our experiments, we employed a drafting-style chair to elevate the user above the desk to accommodate the 3M multi-touch monitor’s stand. Better, would perhaps have been to modify the desk and monitor to embed the device into the desk – or otherwise lower the monitor to accommodate a normal workstation chair. In addition to integrating the device into a more natural workstation, such modification would allow the user to rest his/her forearms on the desk while typing. We received a considerable amount of feedback from the users in this regard. Specifically, users complained about fatigue and expressed opinions concerning the “unnatural” hand/arm posture required while typing on the multi-touch platform. We suspect that users will wish to rest their fingers and wrists upon the device as well, and this poses a considerable problem with respect to zero-force multi-touch platforms. We have more to say about this issue in Chapter 8.

3.4.6 Experimenting from Physically Separate Platforms

In this experiment, we employed three total computer systems: 1) the physical keyboard platform; 2) the multi-touch platform; and 3) the observer/experimenter’s platform. Our software design called for a centralized database, hosted on the experimenter’s computer, which applications on all three computers accessed. Recall from Section 3.2.1.3 that we had developed a custom C# application to administer the study along with a real-time monitoring application from which the experimenter could observe a participant's progress and record the participant's input style. Additionally, we developed online questionnaires (pictured in Figures 11 and 12) for collecting pre- and post-study information.


Multi-Touch Keyboard Study

Pre-Study Questionnaire

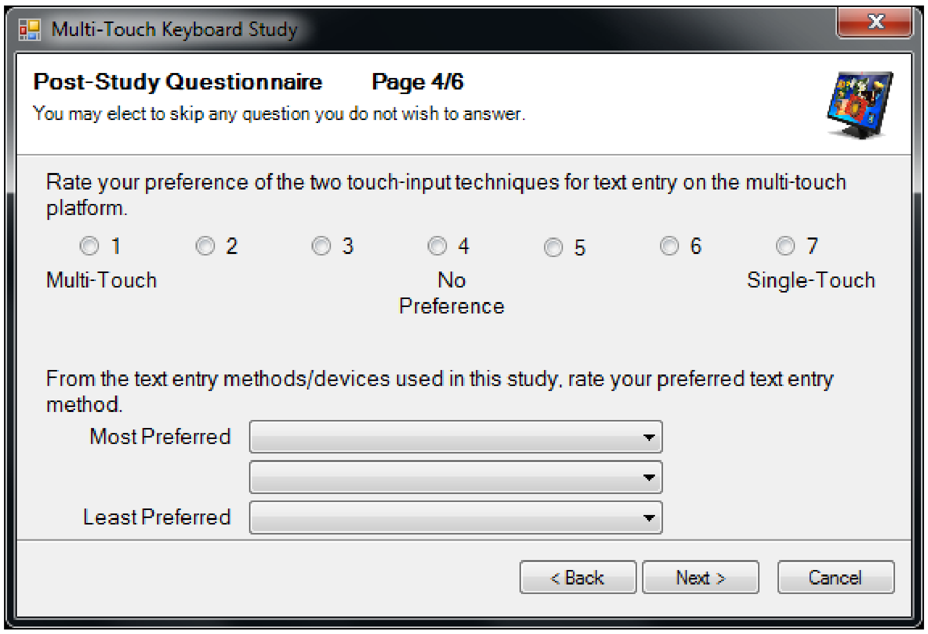
You may elect to skip any question you do not wish to answer.

Year Of Birth:

Gender: Male Female

Handedness: Left-Handed Right-Handed

Figure 11 Text entry study pre-study questionnaire screenshot


Multi-Touch Keyboard Study

Post-Study Questionnaire **Page 4/6**

You may elect to skip any question you do not wish to answer.

Rate your preference of the two touch-input techniques for text entry on the multi-touch platform.

○ 1	○ 2	○ 3	○ 4	○ 5	○ 6	○ 7
Multi-Touch			No Preference			Single-Touch

From the text entry methods/devices used in this study, rate your preferred text entry method.

Most Preferred

Least Preferred

Figure 12 Text entry study post-study questionnaire screenshot

Thus, in total, we had four distinct software applications. The Pre-Study questionnaire application was hosted on the physical keyboard platform, while the Post-Study questionnaire application was hosted on the multi-touch platform. Had we balanced the order of these platforms, we would have hosted both applications on both platforms. The study software was hosted on both platforms, with no modification whatsoever. As mentioned in Section 3.2.1.3 the study application and virtual keyboard operated with the 3M application or the physical keyboard without special software modification. For database purposes, a command-line flag was passed to the study application identifying the platform. The real-time monitoring application was hosted on the experimenter's platform.

Each of these four applications is a “client” of the database hosted on the experimenter’s computer. Because we employed a centralized database, with physically separate client applications accessing it, those computers had to be networked together. This introduces the potential for latency, which could have affected the results of the experiment, particularly speed. To mitigate latency, we physically wired the three computers to a common and isolated 100Mbps Ethernet switch. This switch was disconnected from any additional network connections. Through myriad pilots we could not detect any noticeable latency created by the network connections. Moreover, the input stream events were time stamped at the database, eliminating any time-in-flight recording differences or out-of-sync local computer times. Future experimenters might consider higher bandwidth networks (e.g. 1Gbps) though in our experience we were very far from taxing the bandwidth imitations of our 100Mbps switch. Likewise, our switch was a Commercial-Off-The-Shelf (COTS) ~$80 network switch and higher-priced switches would undoubtedly contain more memory and faster processors. But again, our experience did not indicate that latency was a bona-fide issue. Moreover, such latency would likely exist between both the physical and multi-touch platforms – essentially nullifying any impact. That said, an entirely different approach could have been taken to all-but-eliminate the concern of network latency: that of local recorded input streams. In such a design, a database would be hosted on each of the platforms responsible for collecting the input stream (in our case, these were the physical and multi-touch platforms). This database could be an Online Transaction Processing

(OLTP) database, such as what we used, or even a simple text file. Or one could maintain the entire input stream in volatile memory, for storage once a participant completed the study. With any of these designs, the input stream would be subsequently imported into the “main” database, or otherwise associated with the additional data collected for the participant. This adds some additional complexity, and potential for error, but would effectively eliminate network latency concerns. In our experiment, we felt that the latency concern was so minor that such designs were not required. Other experimenters may choose otherwise, particularly if their network environment is not as controlled as what we described. This question of network latency was moot in our additional experiments, as these studies employed only a single computer system.

A second consideration, for a design with a centralized database but physically separated platforms, is the phrase set from which to pull from. In particular, the study should separate the phrase set so that each platform has a unique subset of phrases from the greater set, or otherwise ensure that the phrases presented are not duplicated between platforms. This is not a terrible problem given a sufficiently large phrase set, and (assuming randomly selected phrases) a randomly generated seed. However, even with a fairly large phrase set, duplicated phrases will be presented between platforms if care is not taken. Another option is to prescribe specific phrases that every participant will transcribe – thus removing the notion of a randomly selected subset presented from a larger phrase database. However, prescribed phrases must be balanced between conditions and do not take into account failed data entry (a problem described in the next section).

3.4.7 Fault Tolerance

We can say, through some bitter experience, that errors within your experiment are all-but-inevitable. It is almost impossible to foresee every eventuality, for example: a power failure; the centralized database or host computer crashing; a network outage; the device driver of the multi-touch monitor crashing; a participant forgetting what phrase to enter; the participant immediately bypassing one or more phrases; the participant typing in an entirely different phrase with no resemblance to the presented text. All of these examples were encountered in our user studies. Thankfully, many such problems were mitigated through a solid software design and implementation. Others were uncovered during pilot testing. Unfortunately, there were still other problems that required us to discard entire sets of data. Such are the realities of experimentation. In this section we offer a few important lessons that may assist future experimenters.

First, it is important that you disallow extremely premature committal of a phrase. Specifically, in our experiment, the participant pressed the *Enter* key in order to complete a phrase and move to the next one. We correctly anticipated that participants may inadvertently press the *Enter* key multiple times between phrases. Without handling this scenario, these situations would have produced invalid data sets. Our solution was simply to prevent the committal of a transcribed phrase whose length was less than some percentage of the length of the presented phrase. Alternately, one might incorporate the notion of detecting these scenarios, automatically discarding the invalid data, and presenting a new phrase to complete the prescribed experiment. Indeed, this alternative might also be used as the solution to “skipping” a phrase that a participant forgot immediately after he/she dismissed the presented text (as with a memorization task).

Second, we found the use of real-time monitoring software to be invaluable – particularly with this first experiment. The monitoring software gave us confidence that our experiment was operating correctly, and did so as the experiment progressed – as opposed to post-study analysis that may have revealed invalid data only after many hours of experimentation had been wasted.

Third, make your data easy to analyze. Data analysis is difficult enough, without having to worry about processing your data into a form that can then be analyzed through statistical software. Avoid storing your input stream in an intermediate format (e.g. text file) which has to be parsed to be useful. Instead, store your data in a bona-fide database which can be queried and analyzed using commercial-off-the-shelf software. Our database of choice was Microsoft SQL Server, a system easily integrated with our custom C# applications and post-study application software (IBM's SPSS [32]).

Finally, we advise that, no matter what precautions you take, you will have problems with your experiment that will produce invalid data. Thus, we suggest a simple mechanism through which you can discard a participant's data set, even if you have collected many additional participants' data since the invalid run. Using an OLTP database system made this a simple matter of a few lines of SQL code. Without a bona-fide database system, be sure you design your software to collect data that is easy to discard without corrupting the entire set of collected data.

CHAPTER FOUR: TEXT ENTRY IMPROVEMENT STUDY

4.1 Introduction

In the first study, we observed a tendency for participants to “drag” their fingers across the multi-touch virtual keyboard to get to the next key. We suspected that this was a cause of mistakes made in entering text on the multi-touch platform. Specifically, the 3M M2256PW [3] multi-touch monitor requires no pressure to trigger the detection of a surface contact. This is in contrast to a typical physical keyboard, which allows the user to rest their fingers on the keys, or slide their fingers from key to key, without causing inadvertent text entry. Thus, we suspected that the participants of our study, all expert typists on a physical keyboard, retained their “sliding/dragging” behavior when typing on the multi-touch keyboard and this caused unintended key presses – errors.

4.2 Experiment

This experiment examines an attempt to improve the text entry performance of the multi-touch platform described in the original study. Specifically, we developed a technique that changes the manner in which key presses are detected by the multi-touch virtual keyboard. In the original implementation, key presses are detected at the moment the user makes physical contact with the multi-touch surface and within the bounds of a virtual key. In the technique studied in this experiment, key presses are detected once contact is made with a key and then released. This, we consider the key press “event” triggered on *Key Down* with the first implementation, and on *Key Up* with this new technique. Moreover, with the *Key Up* technique, the surface contact must originate within the key and, conversely, the contact must be released within the same key. If the user presses an area outside of a key and then drags their finger within the bounds of a virtual key, that key will not be pressed. Likewise, if the user makes contact with the surface, within the bounds of a virtual key, and then drags their finger outside of the key bounds and releases, the key will not be pressed.

This technique mimics the behavior of a button within the “window, icon, menu, pointing device” (WIMP) [79] paradigm of human-computer interaction (HCI). Under typical WIMP interaction, a button will only be triggered when the user left-clicks their mouse while hovering over the button and releases the mouse while still over the button. In this fashion, the user can “cancel” a button press and otherwise mitigate inadvertent button presses. Thus is our intention with the *Key Up* multi-touch text entry technique.

4.2.1 Materials and Methods

4.2.1.1 Participants

Twelve unpaid volunteers participated in this study (4 females, 8 males). All were right-handed and had no visual or physical impairments that prevented normal computer use (as reported by the participants). Their ages ranged from 22 to 32 with an average age of 25. The participants rated their familiarity with touch screen computer systems (including single-touch or multi-touch) from 1-5: 5 highest; their responses were 0%, 17%, 8%, 0%, and 67%, respectively.

4.2.1.2 Apparatus

This experiment employed a 3M M2256PW [3] pictured in Chapter 3, Figure 2. As with the original text entry study, this monitor displayed a resolution of 1680x1050 pixels. The monitor rested on a typical computer desk with a draft-table chair to compensate for the 9” mounting stand of the 3M multi-touch monitor. The 3M multi-touch monitor allows its orientation to be adjusted from 0-degrees (fully horizontal) to 90-degrees (fully vertical). Participants were instructed to adjust the orientation of the multi-touch monitor and chair height to best suit themselves.

4.2.1.3 Software

A custom C# software application was developed to present a virtual keyboard on the display and collect data for our study. This application is functionally identical to that used in the first multi-touch text entry study. However, the associated database was modified to accommodate the simplified experiment design. The pre- and post-study questionnaire applications, from the original study, were likewise adopted and modified to match the questions developed for this experiment.

4.2.1.4 Procedure

Participants initially performed three warm-up phrases on the multi-touch platform to familiarize themselves with the platform and task. They then completed two sets of ten phrases, using either the *Key Up* or *Key Down* technique – as ordered through a Latin Square. Finally, they performed two sets of ten phrases for the remaining technique. Participants were instructed to type "as quickly as possible as accurately as possible". The arrow keys, *Delete*, *Home,* and *End* keys were disabled, leaving *Backspace* as the only available correction key.

Each participant’s 20 total phrases were randomly selected from a set of 465 phrases. This was the same phrase database as was used in the first text entry experiment, derived from MacKenzie and Soukoreff’s database of 500 phrases [45]. The 35 removed phrases contained the word “I”.

After finishing the experiment, participants completed a post-study questionnaire.

4.2.1.5 Environment

The environment of the study was the same room as was used in the initial study – a small room housing the multi-touch platform (pictured in Chapter 3, Figure 6).

4.2.1.6 Design

This experiment employs a within-subjects single-factor design, where the factor is *Technique*, with levels: *Key Up* and *Key Down*. The multi-touch platform always allowed simultaneous inputs and phrases were presented entirely in lowercase. As mentioned in Section 4.2.1.4, the order in which the participants performed the two techniques was balanced through a Latin Square. Thus, 6 participants performed the

Key Up technique before *Key Down* and 6 performed *Key Down* first. In this fashion, we counterbalanced the conditions to mitigate learning effects.

Each participant completed the experiment in approximately 15 minutes.

4.2.2 Metrics

We employed the same metrics as were described in Section 3.2.2: Words per Minute (WPM), Minimum String Distance (MSD), Characters per Second (CPS), Key Strokes per Second (KSPC), MSD Error Rate, Total Error Rate, Not Corrected Error Rate, Corrected Error Rate, and Conscientiousness. As with the original study, we used the MSD, CPS, and MSD Error Rate metrics to support our analysis and do not include these within our results. Moreover, the KSPC metric is reported as “New KSPC” to indicate that we are reporting the newer of the KSPC metrics developed by Soukoreff and MacKenzie in 2003 [69].

Additionally, in the post-study questionnaire, we asked each of the participants which technique they preferred, allowing for the option of *No Preference.*

4.2.3 Results

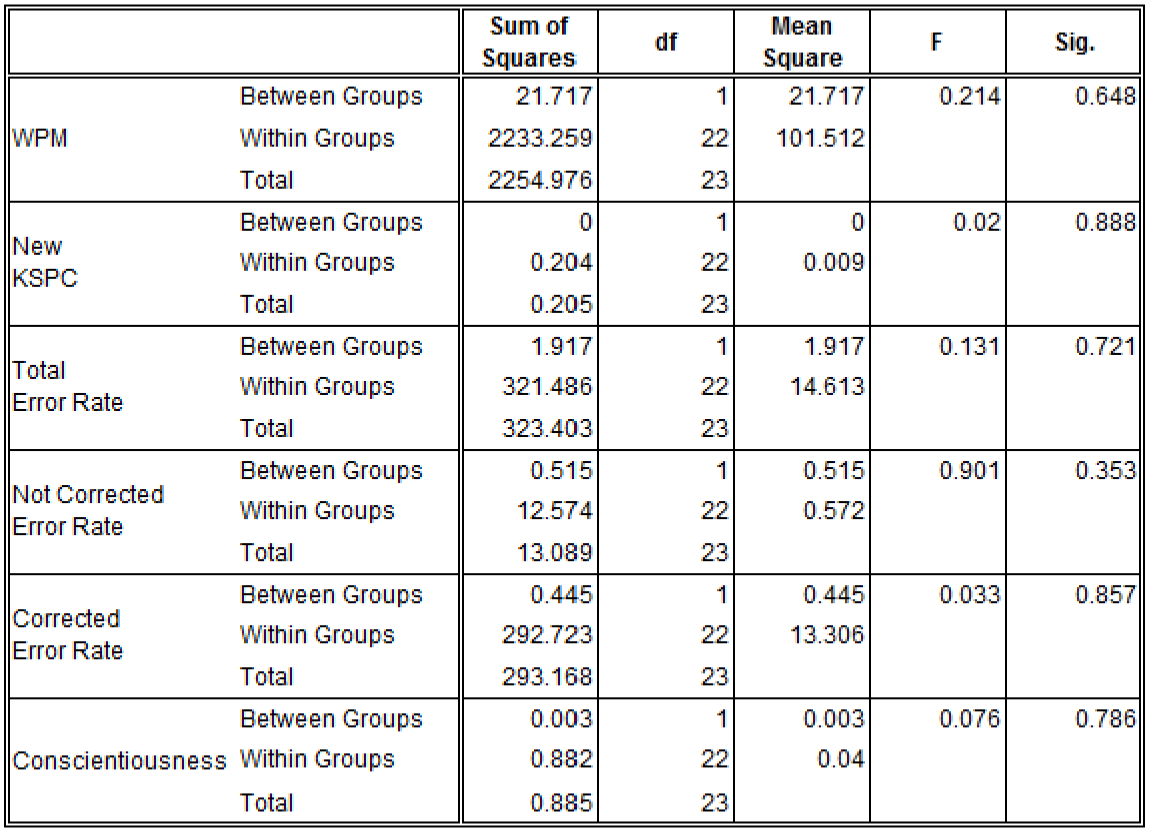
In total, participants entered 240 phrases: 120 for each of the 2 techniques; 20 for each of the 12 participants. The means and standard deviations for each technique and the six chief metrics are summarized in Table 5.

Table 5 Text entry improvement study summary of results

	WPM		New KSPC		Total Error Rate (%)		Not Corrected Error Rate (%)		Corrected Error Rate (%)		Conscientiousness	
Technique	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
KeyDown	48.63	10.06	1.14	0.09	6.44	3.49	0.68	0.63	5.76	3.57	0.78	0.24
KeyUp	46.72	10.09	1.15	0.10	7.00	4.13	0.97	0.86	6.03	3.73	0.80	0.15

To analyze the quantitative data, we ran a one-way ANOVA on each metric. The results are presented in Table 6. We found no significant differences for any of the six metrics.

Table 6 Text entry improvement study analysis results



		Sum of Squares	df	Mean Square	F	Sig.
WPM	Between Groups	21.717	1	21.717	0.214	0.648
	Within Groups	2233.259	22	101.512		
	Total	2254.976	23			
New KSPC	Between Groups	0	1	0	0.02	0.888
	Within Groups	0.204	22	0.009		
	Total	0.205	23			
Total Error Rate	Between Groups	1.917	1	1.917	0.131	0.721
	Within Groups	321.486	22	14.613		
	Total	323.403	23			
Not Corrected Error Rate	Between Groups	0.515	1	0.515	0.901	0.353
	Within Groups	12.574	22	0.572		
	Total	13.089	23			
Corrected Error Rate	Between Groups	0.445	1	0.445	0.033	0.857
	Within Groups	292.723	22	13.306		
	Total	293.168	23			
Conscientiousness	Between Groups	0.003	1	0.003	0.076	0.786
	Within Groups	0.882	22	0.04		
	Total	0.885	23			

4.2.3.1 Post Study Questionnaire Data

The single post-study question asked which, if either, of the techniques the participant preferred. The responses to this question are presented in Figure 13. We found no statistical significance within this data ($\chi_2^2 = 0.5, p = 0.779$).

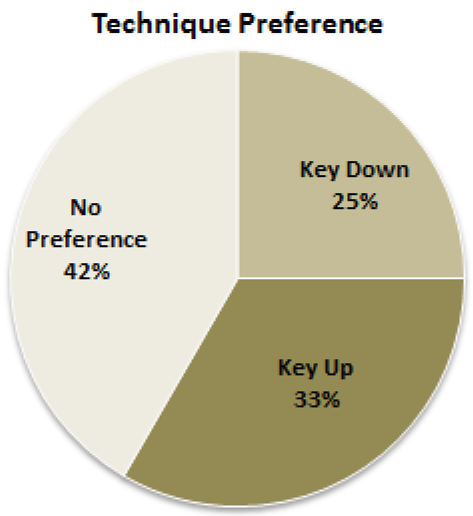


Figure 13 Text entry improvement study questionnaire responses on technique preference

4.2.4 Discussion

The results of this experiment are somewhat surprising, and a little disappointing. We found no statistical significance between the *Key Up* and *Key Down* techniques for any of the six chief quantitative measurements. Nor did we find statistical significance within the users' preference between the two techniques. We were hoping to find some difference between the two techniques, in either the objective or subjective measurements. Indeed, we had expected a reduction in the total error rate under the *Key Up* technique.

As is, the only conclusion we can draw is that the new *Key Up* technique didn’t significantly reduce performance, but it didn’t improve it either. We considered running additional participants through the experiment, but the statistics were so far from significance that we conceded the point as moot.

4.3 Summary

In this study we examined an attempt to improve text entry performance on the 3M M2256PW [3]. Specifically, we studied a technique that changes the manner in which key presses are detected by the multi-touch virtual keyboard – whereby keys are pressed once a surface contact is released. We dubbed this technique *Key Up* as opposed to the original detection behavior of triggering a key press when surface contact is initially made against a virtual key, on *Key Down*.

We collected transcribed phrases from twelve participants along with a post-study question asking which, if either, of the techniques the participants preferred. Our results found no significant difference between the two techniques, neither within the performance metrics nor the question of preference.

Our next study examines multi-touch text entry augmented with mouse-style input within the task of word processing.

4.4 Notes on Replication

This experiment was very simple: two techniques examined through a within-subjects design. We heavily leveraged the software developed for the first experiment along with the topics conveyed in Section 3.4. We draw attention to our simplified design that eliminates comparisons between *task*, *allowed inputs* and *phrase case*. A future experimenter may wish to evaluate this technique considering these additional conditions, or on different multi-touch hardware.

By only using a single multi-touch platform, one can host the database on the same computer through which the experiment is conducted. This eliminates the concerns described in Section 3.4.6. One can still employ a real-time monitoring system (as described in Section 3.2.1.3) though this system will be connected, via network, to the database hosted on the multi-touch platform. In such a scenario, there is no concern about network latency, as the monitoring system is primarily a read-only application and does not contribute time-sensitive data to the database.

CHAPTER FIVE: WORD PROCESSING TECHNIQUE STUDY

5.1 Introduction

In the first two studies, we examined plain-text entry on a multi-touch platform. Participants in those studies were not concerned with punctuation, formatting text, organizing text into paragraphs, or using computer interaction mechanisms outside of the keyboard. Such requirements are common within the task of word processing, and the intent of this experiment is to establish a foundation from which to examine word processing on a multi-touch platform.

Specifically, we wish to quantify the performance of word processing on a multi-touch platform for comparison against a computer with a physical keyboard and mouse. However, before we can properly compare word processing between a traditional computer and a multi-touch platform, we must investigate *how* one might accomplish word processing on a multi-touch system.

As we mentioned, word processing requires punctuation, text organization, and formatting. By *formatting*, we imply the concept of “rich-text” where text is stylized through, for example, font selection, bold, underline, or italics. And, while the word processing requirements of punctuation and organizing text into paragraphs can largely be accomplished through the keyboard, formatting is commonly applied through the use of a mouse-style pointing device. Therefore, when considering the examination of word processing through multi-touch, we recognize that we are exploring two modes of computer interaction: text entry through a keyboard, and mouse-style interaction. If we were to adopt the multi-touch virtual keyboard examined in the first two experiments, then we realized that what we were adding in this third experiment was the notion of mouse-style interaction to help users accomplish a rich-text entry task. Thus we asked the questions:

- How might users engage a formatting option, such as bold or italics, on a multi-touch platform?
- How might users trigger a WIMP-style button through a multi-touch platform?
- How might users select text on a multi-touch platform?

More generally, these thoughts can be summarized into the question: “how might a user perform mouse-style interaction on a multi-touch platform?” Borrowing work from the state of the art in multi-touch (described in Chapter 2) and by leveraging our own research in mouse-style interaction through multi-touch (detailed in Appendix A) we entertained two approaches toward answering the question: *Direct Interaction* and *Indirect Interaction*.

Direct Interaction indicates that the user touches directly onto the portion of the screen he/she wishes to interact with. For example, if the user wishes to press a WIMP-style button, he/she would touch the multi-touch screen within the visual bounds of the button. Indeed, direct interaction is employed by the virtual keyboard we have been studying. As another example, if the user wishes to select (or otherwise highlight) a bit of text from the screen, he/she would directly touch near the beginning of the text-to-select and then drag his/her finger to the end of the text-to-select. One might consider this technique a form of direct interaction mouse emulation, whereby the user's touch triggers a left-mouse-button click. The mouse need not be directly emulated in order to interact with GUI elements, and indeed our implementation of direct interaction does not trigger mouse movement. We mention the possibility of direct interaction mouse emulation to contrast *Indirect Interaction*.

Indirect Interaction denotes an emulated mouse on the multi-touch platform with the aid of a virtual mouse pad. With this technique, the user touches an area of the screen resembling a laptop trackpad. Our implementation of the virtual mouse pad is depicted in Figure 14.

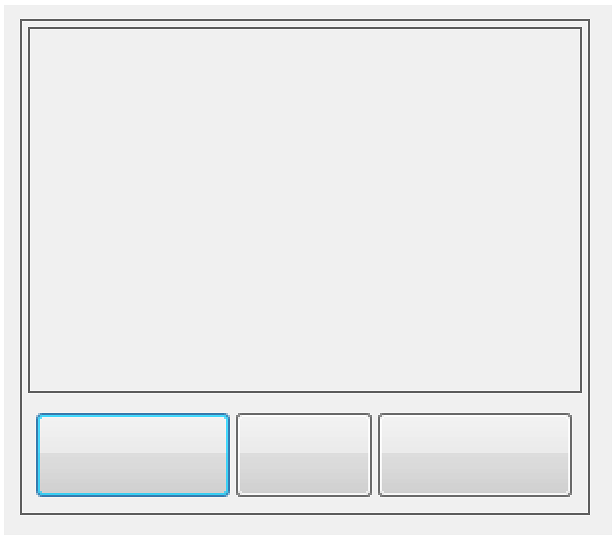


Figure 14 Word processing technique study virtual mousepad

With *Indirect Interaction* the mouse cursor is moved as the user drags his/her finger within the bounds of the virtual mouse pad. Movement scale is derived from the ratio of the resolution of the screen to the size of the virtual mouse pad. For example, a screen resolution of 1680x1050 pixels and a virtual mouse pad size of 285x248 pixels would scale mouse movement by a factor of approximately 5.9 on the horizontal axis, and 4.2 on the vertical axis.

At the bottom of the virtual mouse pad, we have included GUI-style buttons representing the left, middle, and right mouse buttons. Touching these buttons triggers the corresponding mouse click. Likewise, tapping on the virtual mouse pad produces a left-mouse-click and double tapping produces a double-click. In this fashion, our implementation of a virtual mouse pad closely approximates that of a laptop trackpad.

Because the *Indirect Interaction* technique truly emulates the input of the mouse, a WIMP-style application need not be altered to accommodate the multi-touch platform aside from the screen space required for the virtual mouse pad. One could either embed the virtual mouse pad within their application, or present the virtual mouse pad on-demand as an “on-top” application overlying any GUI program.

5.2 Experiment

This experiment examines the performance of a word processing task between *Direct Interaction* and *Indirect Interaction* mouse-style input techniques on a multi-touch platform. Participants are presented a series of rich-text paragraphs and asked to transcribe this text along with punctuation, organization, and formatting. Software records the corresponding input stream for subsequent analysis.

Text is in a consistent font, which the user does not modify, and may be bold or italicized. Toggle-style buttons for engaging bold or italics formatting are presented above the transcribed text-entry field. Common shortcut keys for bold or italics, such as

Ctrl-B and Ctrl-I, have been explicitly disabled so as to force the user to format the text using the prescribed buttons.

In this study, we seek an understanding of the word processing performance characteristics of a single-user multi-touch platform employing two mouse-style interaction techniques. Additionally, we query the participants for their preference between the methods. This information is intended to inform our next multi-touch word processing study, which will compare the preferred and/or higher performing technique against the same task performed on a traditional computing platform.

5.2.1 Materials and Methods

5.2.1.1 Participants

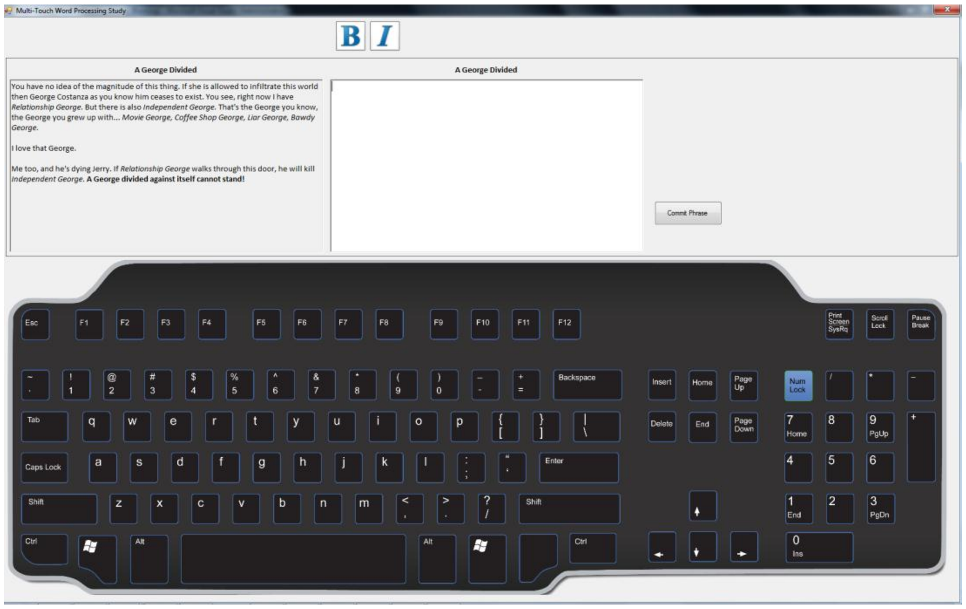
Twelve unpaid volunteers participated in this study (1 female, 11 males). All were right-handed and had no visual or physical impairments that prevented normal computer use (as reported by the participants). Their ages ranged from 22 to 42 with an average age of 30. The participants rated their familiarity with touch screen computer systems (including single-touch or multi-touch) from 1-5: 5 highest; their responses were 0%, 8.3%, 8.3%, 8.3%, and 75%, respectively.

5.2.1.2 Apparatus

This experiment employed a 3M M2256PW [3] pictured in Chapter 3, Figure 2. As with the first two studies, this monitor displayed a resolution of 1680x1050 pixels. The monitor rested on a typical computer desk with a draft-table chair to compensate for the 9” mounting stand of the 3M multi-touch monitor. The 3M multi-touch monitor allows its orientation to be adjusted from 0-degrees (fully horizontal) to 90-degrees (fully vertical). Participants were instructed to adjust the orientation of the multi-touch monitor and chair height to best suit themselves.

5.2.1.3 Software

A custom C# software application, with screenshots pictured in Figures 15 and 16, was developed to collect data for our study.

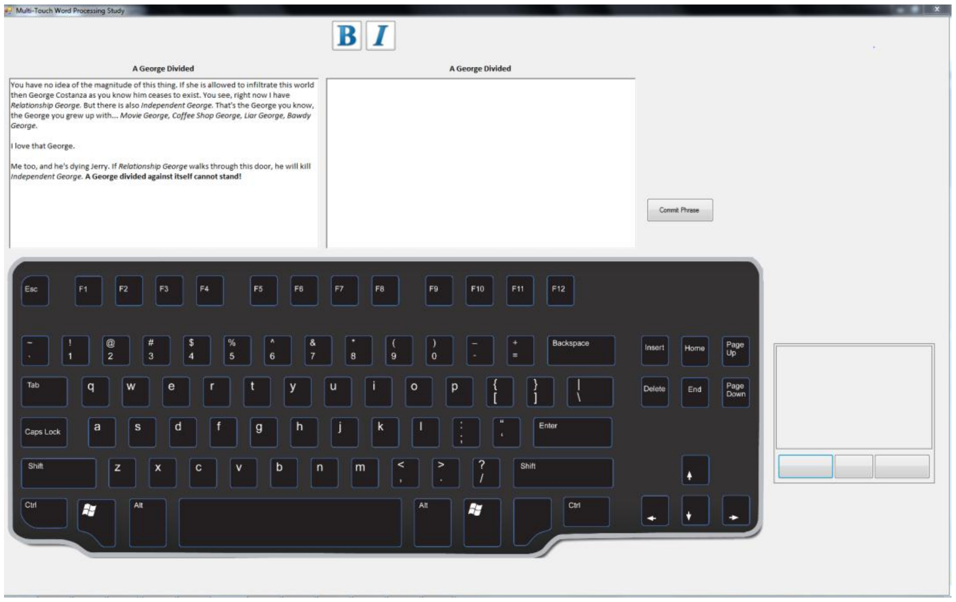


You have no idea of the magnitude of this thing. If she is allowed to infiltrate this world then George Costanza as you know him ceases to exist. You see, right now I have *Relationship George*. But there is also *Independent George*. That's the George you know, the George you grew up with... *Movie George, Coffee Shop George, Liar George, Bawdy George*.

I love that George.

Me too, and he's dying Jerry. If *Relationship George* walks through this door, he will kill *Independent George*. **A George divided against itself cannot stand!**

Figure 15 Word processing technique study direct interaction software



You have no idea of the magnitude of this thing. If she is allowed to infiltrate this world then George Costanza as you know him ceases to exist. You see, right now I have *Relationship George*. But there is also *Independent George*. That's the George you know, the George you grew up with... *Movie George, Coffee Shop George, Liar George, Bawdy George*.

I love that George.

Me too, and he's dying Jerry. If *Relationship George* walks through this door, he will kill *Independent George*. **A George divided against itself cannot stand!**

Figure 16 Word processing technique study indirect interaction software

The two figures depict the two interaction techniques: *Direct* and *Indirect Interaction*. For both modes, the upper portion of the screen is identical. This area of the screen contains the read-only presented text, and a rich-text field for transcription. Above the transcription area are the bold and italics formatting buttons. To the right of the transcription field is a “Commit Phrase” button which participants activate in order to complete an entry and continue to the next.

The virtual keyboards are contained in the lower portions of each screen. Within *Direct Interaction* mode, the keyboard is identical to that used in the first two text entry studies. Under *Indirect Interaction* the number pad is removed in order to allow the presentation of the virtual mouse pad. Additionally, the curved, empty portion of the full-size keyboard has been removed. Otherwise, the keys and their arrangement are identical between the two interaction modes.

As with the first two studies, the input stream is recorded to a database. This database was modeled after the previous studies, but was augmented in order to support the notion of formatted text. Additionally, the transcribed rich-text is saved to a file, for each phrase, in order to provide a backup in case of rich-text parsing errors during database serialization.

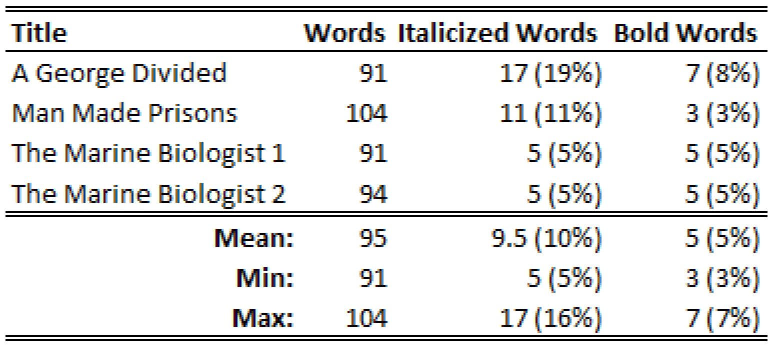
5.2.1.4 Procedure

Participants entered two paragraphs for both of the techniques under examination. A participant's starting technique was determined through a Latin Square. Participants were instructed to type "as quickly as possible as accurately as possible". Unlike the first two experiments, were the arrow keys, *Delete*, *Home,* and *End* keys were disabled, no keys were disabled in this experiment. Participants were free to transcribe and format text in the manner most comfortable for them and could utilize all available keys on the virtual keyboard.

The four paragraphs were taken from the television series *Seinfeld*, and are presented in Appendix B. All participants completed the same four paragraphs; however, the order in which the paragraphs were presented was balanced using two Latin Squares, one for each technique ordering. Each paragraph contained capitalization, punctuation, new lines, and bold and italics formatting. A summary of the makeup of each paragraph is presented in Table 7.

After finishing the experiment, participants completed a post-study questionnaire.

Table 7 Word processing techinque study paragraph summary



Title	Words	Italicized Words	Bold Words
A George Divided	91	17 (19%)	7 (8%)
Man Made Prisons	104	11 (11%)	3 (3%)
The Marine Biologist 1	91	5 (5%)	5 (5%)
The Marine Biologist 2	94	5 (5%)	5 (5%)
Mean:	95	9.5 (10%)	5 (5%)
Min:	91	5 (5%)	3 (3%)
Max:	104	17 (16%)	7 (7%)

5.2.1.5 Environment

The environment of the study was the same room as was used in the first two studies – a small room housing the multi-touch platform (pictured in Chapter 3, Figure 6).

5.2.1.6 Design

This experiment employs a within-subjects single-factor design, where the factor is *Technique*, with levels: *Direct Interaction* and *Indirect Interaction*. The multi-touch platform always allowed simultaneous inputs. As mentioned in Section 5.2.1.4, the order in which the participants performed the two techniques was balanced through a Latin Square. Thus, 6 participants performed the *Direct Interaction* technique before *Indirect Interaction* and 6 performed *Indirect Interaction* first. In this fashion, we counterbalanced the conditions to mitigate learning effects.

Each participant completed the experiment in approximately 20 minutes.

5.2.2 Metrics

5.2.2.1 Text Entry Metrics

We employed the same metrics as were described in Section 3.2.2: Words per Minute (WPM), Minimum String Distance (MSD), Characters per Second (CPS), Key Strokes per Second (KSPC), MSD Error Rate, Total Error Rate, Not Corrected Error Rate, Corrected Error Rate, and Conscientiousness. And as with the first two studies, we used the MSD, CPS, and MSD Error Rate metrics to support our analysis and do not include these within our results. Moreover, the KSPC metric is reported as “New KSPC” to indicate that we are reporting the newer of the KSPC metrics developed by Soukoreff and MacKenzie in 2003 [69].

5.2.2.2 Total Time, Total Key Stroke Time, and Completion Time

We also engaged a new metric, *Total Time*, to track the total amount of time the participants spent entering a phrase. This measurement was available during the first two experiments, but was of minimal use. In particular, the *Total Time* metric is a measurement of the speed at which the participant completes the task. With the first two experiments, this speed was better reported as WPM, without augmenting the metric with additional information. WPM is calculated using the time of the first key stroke to the time of the last key stroke, before a participant commits the phrase to the database. In this fashion, we deduct the amount of time that the participant spends waiting to commit the phrase. We dub the time between the first and last key strokes as *Total Key Stroke Time* and the time between the last key stroke and the moment of phrase committal as *Completion Time*. Completion time might indicate that the user is inspecting the transcribed phrase for final errors; or it might signify that the user is resting before the start of the next phrase. Alternately, completion time might simply imply that the participant forgot to commit the phrase.

Whatever the reason for additional Total Time/Completion Time, these metrics are not particularly important to the analysis of a plain text entry task. With our word processing task, however, these metrics might reveal information concerning *how* participants format text and if there is a difference between our *Direct* and *Indirect*

Interaction techniques. Specifically, participants can activate formatting *before* or *after* they enter text. Activating a formatting option *before* typing, toggles the formatting to its “on” state; where after the user types formatted text and then deactivates the formatting option to continue typing plain text. Alternately, the user might enter plain text and then select that text for subsequent formatting. Tracking the user’s total time completing a task might offer performance insights that could help differentiate between the two interaction techniques under study. Each of these metrics is reported in seconds.

5.2.2.3 Formatting Errors

In addition to these metrics, we developed two new measurements for tracking bold and italics formatting errors. At first glance, one might consider the notion of detecting formatting errors to be trivial; our experience, however, revealed that this was a fairly difficult problem to solve. Consider the following example:

The quick **brown** fox   (presented text)

```
The quick brown fox         (transcribed text)
```

In this example, a character-by-character comparison will show 5 formatting errors (the entire transcribed word ‘brown’ is not emboldened). However, a human might consider this as only 1 formatting error – the whole word. This concept is similar, though not the same as, the minimum string distance (MSD) for string comparison (described in Section 3.2.2). When determining the MSD, there are three primitive operations that can be used to transform one string into another: insertion, deletion, and substitution). When considering formatting errors there are potentially four primitive operations: *char-format, char-unformat, word-format*, and *word-unformat*. Similar to treating consecutive character formatting errors as one error, one might also consider consecutive word formatting errors as a single error. If so, we could expand our set of primitive formatting operations to include *phrase-format* and *phrase-unformat*.

Words that are partially formatted, either in the presented or transcribed text, complicate the application of *word* or *phrase* operations, to where an incorrectly formatted phrase could devolve into a set of *character* formatting operations. Indeed, the correct approach to partially formatted characters is unclear. Consider the following example:

The quick **br**ow**n** fox (presented text)

The quick brown fox (transcribed text)

In this, admittedly contrived, example, the “br” and the “n” of the presented word “brown” are bold and the transcribed text formats none of the characters. We have two possible approaches for measuring this error: 1) a character-by-character comparison; or 2) consecutive character formatting comparisons. Thus, in the above example, a character-by-character analysis would reveal three formatting errors, while a consecutive character formatting analysis would reveal only two (“br” would be considered a single formatting operation).

Another option is to consider formatting errors only at the word level – meaning, that any formatting errors within the word (a single character, multiple characters, or the entire word) should be considered as only a single formatting error. Thus the word is either formatted correctly, in its entirety, or is in error. One could extend this notion to the formatting of consecutive words – which seems reasonable considering that consecutively formatted words are rarely interrupted by single unformatted characters or words. However, this matter is complicated when considering mixed formatting modes – the combination of, for example, bold and italics formatting within a localized grouping of characters and words.

With six total formatting transformations, and the difficulties of partial and mixed-mode formatting, it is reasonable to question the necessity of measuring formatting errors in the same fashion as MSD measures spelling errors. Furthermore, format error measurement is complicated by misspelled words. A formatting analysis of a misspelled phrase could produce a grossly inaccurate measurement. Consider the following example:

```
The quick **brown** fox          (presented text)
The **brown** fox                (transcribed text)
```

A character-by-character analysis of the misspelled words in the above example would compare the unformatted “quick” to the formatted “**brown**” and produce 5 formatting errors; when there are, in fact, no formatting errors but rather an entirely missing (misspelled) word.

Instead, we can compare formatting against an optimal alignment string of the presented and transcribed phrase. Recall that optimal alignment strings, described in Section 3.2.2, incorporate the MSD primitive insertion, deletion, and substitution operations to create presented/transcribed string pairs that transform one string into the other. The set of alignment strings produced by the algorithm are always the same length, as each alignment string is generated through the optimal number of primitive transformations. The set will likely contain more than one alignment string, though which we choose is immaterial with respect to format comparison. However, regardless of which alignment string we use, it will likely be longer than the original presented or transcribed text. This is because the algorithm developed by Soukoreff and MacKenzie [43] inserts a '-' dash character, into the presented or transcribed alignment string component, to denote and insertion or deletion operation. This notion is important to our formatting comparison, as dash characters are unformatted, externally created components of the transcribed text that should be excluded from format testing. Thus, we must maintain synchronization between the alignment phrase pair and the corresponding formatting information stored for the originally transcribed phrase.

Collecting formatting information from the transcribed text also poses a challenge. Our metrics require that we record the users input stream – that is, that we track every key stroke produced by the user during transcription. However, even within the context of rich-text entry, the input stream does not yield accurate formatting information. Consider the following scenario: the user transcribes the phrase "the quick brown fox" and then selects the word "brown" and formats the selection using the bold or italics buttons. The text input stream never reveals that formatting took place.

Alternately, consider the same phrase wherein the user typed "the quick", then enabled the bold formatting feature, continued typing "brown" and then disabled the bold formatting before completing the phrase. As before, assuming mouse-style input triggered the formatting, the input stream does not reveal the formatting. Furthermore, even if keyboard formatting shortcuts, such as Ctrl-B (bold) or Ctrl-I (italics), were allowed one cannot rely on the user employing these shortcuts when a mouse-style formatting option exists. Moreover, one must track corrections properly – which is particularly difficult when the user can randomly access the transcribed-text field (as is the case in our experiment). No, the input stream cannot be reliably used to determine the formatting of transcribed text.

One could consider tracking formatting information through a combination of the text input stream and mouse-style actions. However, we believe this to be an unreliable, error-prone solution. In our implementation, we determine formatting as a post-process, triggered once the participant has committed a transcribed phrase to the database. Specifically, we store the transcribed text in memory as rich text, which maintains formatting alongside the plain text. Once the participant commits a phrase, our software parses the rich text and records bold and italics formatting data corresponding to every character in the input field. By storing the formatting information per character, we are free to employ any of the error measurement approaches previously discussed (e.g. *character-by-character*, *consecutive-character*, *whole-word*, or *whole-consecutive word* analysis).

As you can see, the complexity of measuring formatting errors is deceptive. What at first seems trivial becomes a fairly challenging problem under closer scrutiny. In our final implementation, we chose to track formatting errors *character-by-character*; meaning, we compare the formatting of each valid character, from the presented component of an optimal alignment string, against the transcribed character. If the formatting does not match we count an error, maintaining separate values for bold and italics errors.

5.2.2.4 Post-Study Questionnaire

In addition to the quantitative measures, we asked each of the participants to respond to the following statements in a post-study questionnaire:

- The multi-touch platform accurately detected my intended key presses (scale 1:5)
- The multi-touch platform accurately detected my intended mouse interactions (scale 1:5)
- Rate your preference of the two input techniques on the multi-touch platform (option included for “no preference”)
- Would you use the multi-touch device for everyday word processing activities?

5.2.3 Results

5.2.3.1 Text Entry Metrics

In total, participants entered 48 paragraphs of formatted text: 24 for each of the 2 techniques; 4 for each of the 12 participants. The means and standard deviations for each technique and the 6 chief text entry metrics are summarized in Table 8.

Table 8 Word processing technique study summary of results for text entry metrics

	WPM		New KSPC		Total Error Rate (%)		Not Corrected Error Rate (%)		Corrected Error Rate (%)		Conscientiousness	
Technique	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Direct	23.04	5.00	1.28	0.16	13.06	7.22	1.17	1.93	11.89	5.90	0.91	0.08
Indirect	23.12	5.74	1.33	0.18	14.19	6.33	0.56	0.45	13.63	6.18	0.95	0.05

To analyze the quantitative data, we ran a one-way ANOVA on each metric. The results are presented in Table 9. We found no significant differences for any of the six metrics.

Table 9 Word processing technique study analysis results for text entry metrics

		Sum of Squares	df	Mean Square	F	Sig.
WPM	Between Groups	0.039	1.000	0.039	0.001	0.971
	Within Groups	637.815	22.000	28.992		
	Total	637.854	23.000			
New KSPC	Between Groups	0.015	1.000	0.015	0.510	0.483
	Within Groups	0.633	22.000	0.029		
	Total	0.648	23.000			
Total Error Rate	Between Groups	7.608	1.000	7.608	0.165	0.689
	Within Groups	1014.793	22.000	46.127		
	Total	1022.401	23.000			
Not Corrected Error Rate	Between Groups	2.233	1.000	2.233	1.143	0.297
	Within Groups	42.971	22.000	1.953		
	Total	45.204	23.000			
Corrected Error Rate	Between Groups	18.086	1.000	18.086	0.496	0.489
	Within Groups	802.779	22.000	36.490		
	Total	820.865	23.000			
Conscientiousness	Between Groups	0.007	1.000	0.007	1.506	0.233
	Within Groups	0.099	22.000	0.004		
	Total	0.106	23.000			

5.2.3.2 Word Processing Metrics

We have grouped the *Bold Errors, Italics Errors, Total Time, Total Key Stroke Time,* and *Completion Time* metrics into a group titled *Word Processing Metrics*. The means and standard deviations of these metrics, for each technique, are summarized in Table 10.

Table 10 Word processing technique study summary of results for word processing metrics

	Bold Errors		Italics Errors		Total Time		Total Key Stroke Time		Completion Time	
Technique	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Direct	0.92	1.73	3.08	6.16	291.68	60.31	273.81	58.84	17.86	15.73
Indirect	0.50	1.00	3.17	6.44	319.57	84.91	280.29	69.44	39.28	25.22

To analyze this data, we ran a one-way ANOVA on each metric. The results are presented in Table 11. We found significant differences for only one of the word processing metrics, *Completion Time* ($F_{1,23} = 6.23$, $p < 0.05$) indicating that *Direct Interaction* required less *Completion Time* than *Indirect Interaction*.

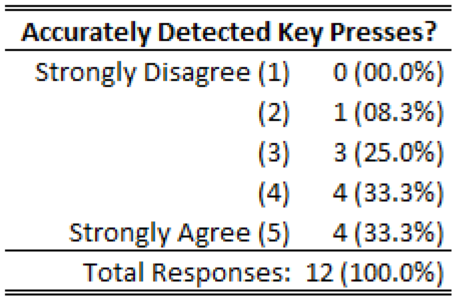
Table 11 Word processing technique study analysis results for word processing metrics

		Sum of Squares	df	Mean Square	F	Sig.
Bold Errors	Between Groups	1.042	1.000	1.042	0.522	0.478
	Within Groups	43.92	22.00	2.00		
	Total	44.96	23.00			
Italics Errors	Between Groups	0.04	1.00	0.04	0.00	0.97
	Within Groups	872.58	22.00	39.66		
	Total	872.63	23.00			
Total Time	Between Groups	4668.86	1.00	4668.86	0.86	0.36
	Within Groups	119323.45	22.00	5423.79		
	Total	123992.31	23.00			
Total Key Stroke Time	Between Groups	251.76	1.00	251.76	0.06	0.81
	Within Groups	91121.57	22.00	4141.89		
	Total	91373.33	23.00			
Completion Time	Between Groups	2752.28	1.00	2752.28	6.23	0.02
	Within Groups	9717.13	22.00	441.69		
	Total	12469.40	23.00			

5.2.3.3 Post Study Questionnaire Data

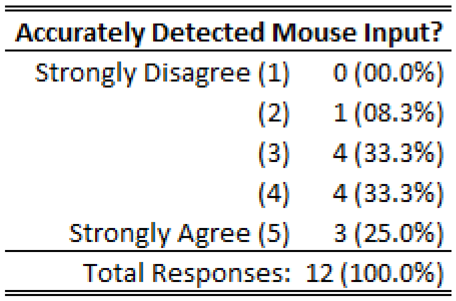
In the post-study questionnaire, we asked each participant Likert-scale questions pertaining to the accurate detection of intended key presses and mouse input. Tables 12 and 13 present the results of these questions, where the value 1 indicates *Strongly Disagree* and 5, *Strongly Agree.*

Table 12 Word processing technique study questionnaire responses on accurately detected key presses



Accurately Detected Key Presses?	
Strongly Disagree (1)	0 (00.0%)
(2)	1 (08.3%)
(3)	3 (25.0%)
(4)	4 (33.3%)
Strongly Agree (5)	4 (33.3%)
Total Responses:	12 (100.0%)

Table 13 Word processing technique study questionnaire responses on accurately detected mouse input



Accurately Detected Mouse Input?	
Strongly Disagree (1)	0 (00.0%)
(2)	1 (08.3%)
(3)	4 (33.3%)
(4)	4 (33.3%)
Strongly Agree (5)	3 (25.0%)
Total Responses:	12 (100.0%)

Figure 17 presents the responses to the participant's technique preference. The results were unanimous: all participants preferred *Direct Interaction*.

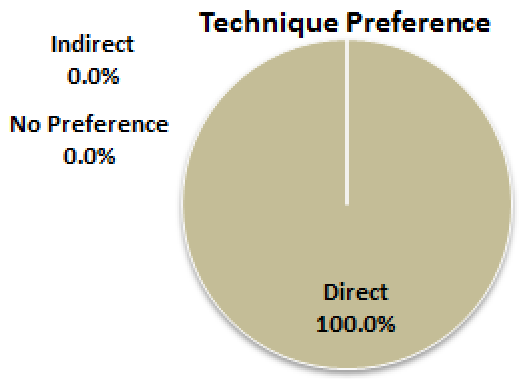


Figure 17 Word processing technique study questionnaire responses on technique preference

The final subjective question asked if the participant "would use the multi-touch device to for everyday word processing activities." The responses to this question are presented in Figure 18. We found no statistical significance within this data ($\chi_1^2 = 3.0, p = 0.083$) at a 95% confidence interval.

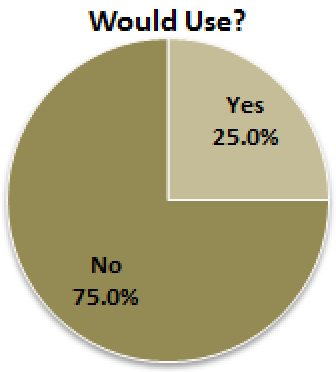


Figure 18 Word processing technique study questionnaire responses on "would use" multi-touch platform for everyday word processing activities

5.2.4 Discussion

The results of this experiment are split between the objective and subjective data. Of the 11 reported metrics, we found that *Completion Time* was the only measurement that revealed statistically significant performance differences between the *Direct Interaction* and *Indirect Interaction* techniques. However, the post-study questionnaire responses reveal a unanimous preference for *Direct Interaction*.

Looking at these results, from a text entry perspective, it is understandable why there would be little-to-no difference between the techniques. Indeed, both techniques used the same multi-touch platform and nearly identical virtual keyboards. As we described in Section 5.2.1.3, the virtual keyboard, used under the *Indirect Interaction* technique, differed only in that the number pad was omitted and that the upper "decorative" area of the keyboard was removed. Since none of the paragraphs contained numbers, participants apparently found no use for the number pad. Thus, we can consider the virtual keyboards substantively identical between the two techniques.

The five reported metrics: New KSPC, Total Error Rate, Corrected Error Rate, Not Corrected Error Rate, and Conscientiousness; are specific to the performance of text entry, and are therefore not influenced by the introduction of formatting to the text entry task. It is consequently, speed (measured in WPM) and the group of word processing metrics: Bold Errors, Italics Errors, Total Time, Total Key Stroke Time, and

Completion Time; wherein we should have found any difference between the two techniques, if any were to be found.

We find it unsurprising that there were no statistically significant differences between bold and italics errors. Indeed, the means of these metrics were 0.92 and 3.08 for bold and italics errors, respectively. The formatting breakdown of study paragraphs, described in Table 7, reveals that these errors constitute a formatting-error-to-formatted-word ratio of 0.92:5 (~18%) for bold formatting and 3.08:9.5 (~32%) for italics. If we use the traditional 5 characters per word we can present these ratios as 0.92:25 (~4%) and 3.08:35 (~6%) – formatting-error-to-formatted-characters. Thus, we recognize how few formatting errors the participants committed, irrespective of interaction technique.

What is somewhat surprising, are the significantly lower completion times between *Direct* and *Indirect* interaction. Recall that completion time measures the duration between the last key stroke and when the participant commits a transcribed phrase to the database. That the completion times were lower for *Direct Interaction* indicates that, for some reason, users spent less time "working" on the phrase after they finished entering text. Considering the unanimous user preference for the *Direct Interaction* mode, we postulate that this time was spent formatting text, and that users found this to be more quickly accomplished under the *Direct Interaction* technique.

Even if we had no quantitative evidence to suggest which technique we should employ for continued study, the overwhelming subjective support for the *Direct*

Interaction technique offers clear guidance for future multi-touch word processing experiments.

The subjective data also reveal that most of our participants felt that the multi-touch platform accurately detected their keyboard and mouse input. However, only 25% of our participants stated that they would use the multi-touch platform for everyday word processing tasks.

5.3 Summary

In this study we augmented our multi-touch text entry task by including rich text formatting, punctuation, organizing text into paragraphs, and using mouse-style interaction – requirements common within the task of word processing. Our intent was to establish a foundation from which to examine word processing on a multi-touch platform.

Specifically, our overarching goal is to quantify the performance of word processing on a multi-touch platform for comparison against a computer with a physical keyboard and mouse. However, before we could properly compare word processing between a traditional computer and a multi-touch platform, we had to investigate *how* one might accomplish word processing on a multi-touch system.

Thus, we examined two techniques for mouse-style interaction on a multi-touch platform: *Direct* and *Indirect Interaction*; within the context of a word processing task.

We collected transcribed paragraphs of rich text from twelve participants, along with post-study questionnaire responses. Our results found statistical significance in only one of the 11 objective measurements: *Completion Time*; revealing that *Direct Interaction* outperformed *Indirect Interaction*, at lease with respect to the time spent "working" on the phrase after entering text. The results also found unanimous agreement that *Direct Interaction* was the preferred interaction technique. This information provides clear direction for our fourth study: examining word processing between multi-touch and traditional computing platforms.

5.4 Notes on Replication

5.4.1 Formatting Error Metrics

This experiment was substantially different than the first two experiments on text entry. In this study, along with all of the software systems, we had to develop two new metrics for measuring formatting errors. A future experimenter may question our approach for these metrics, or at least may wish to replicate the measurements. Thus, we herein augment the discussion provided in Section 5.2.2.3.

Firstly, recall that our choice in presenting the formatting error metrics as “character-wise” statistics is not dependent on the method by which we record formatting meta-data. Specifically, we track formatting for each character within the transcribed text. Figure 19 presents the database schema for this portion of our implementation.

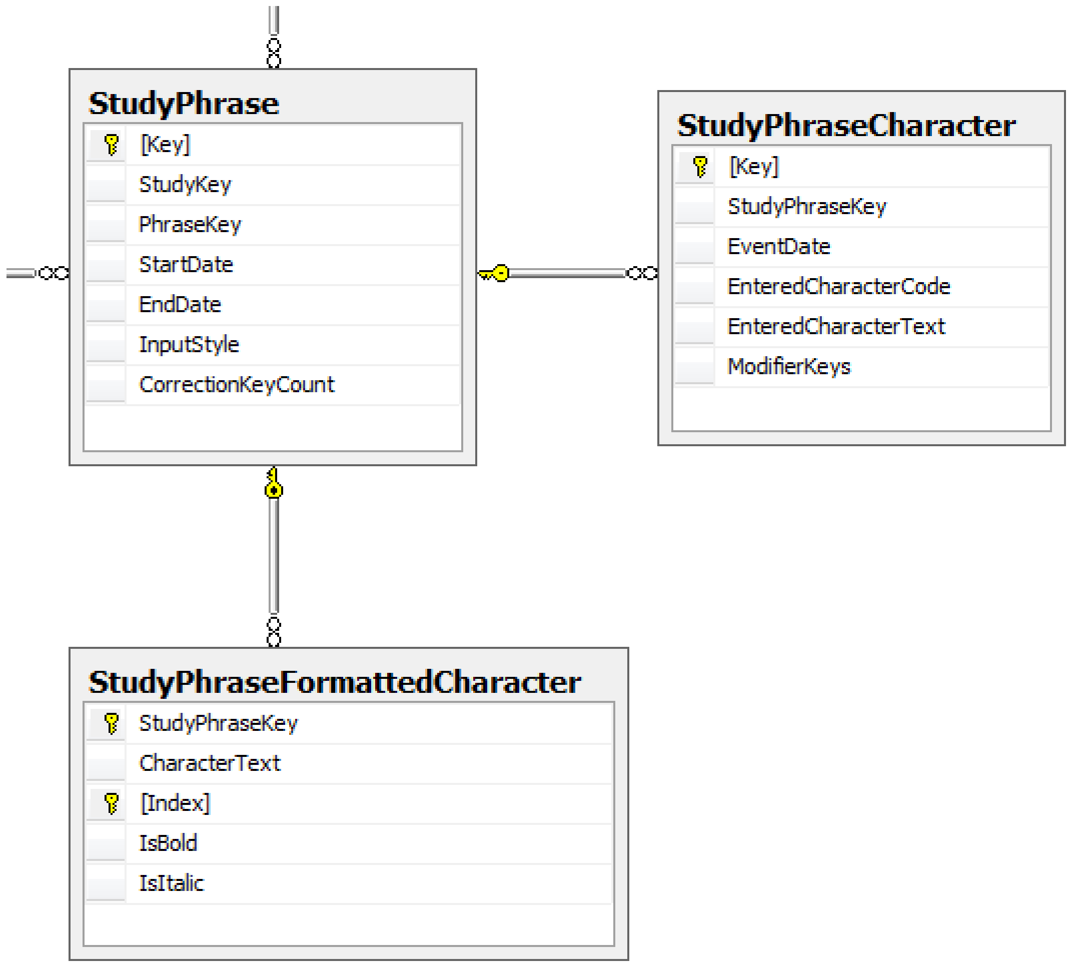
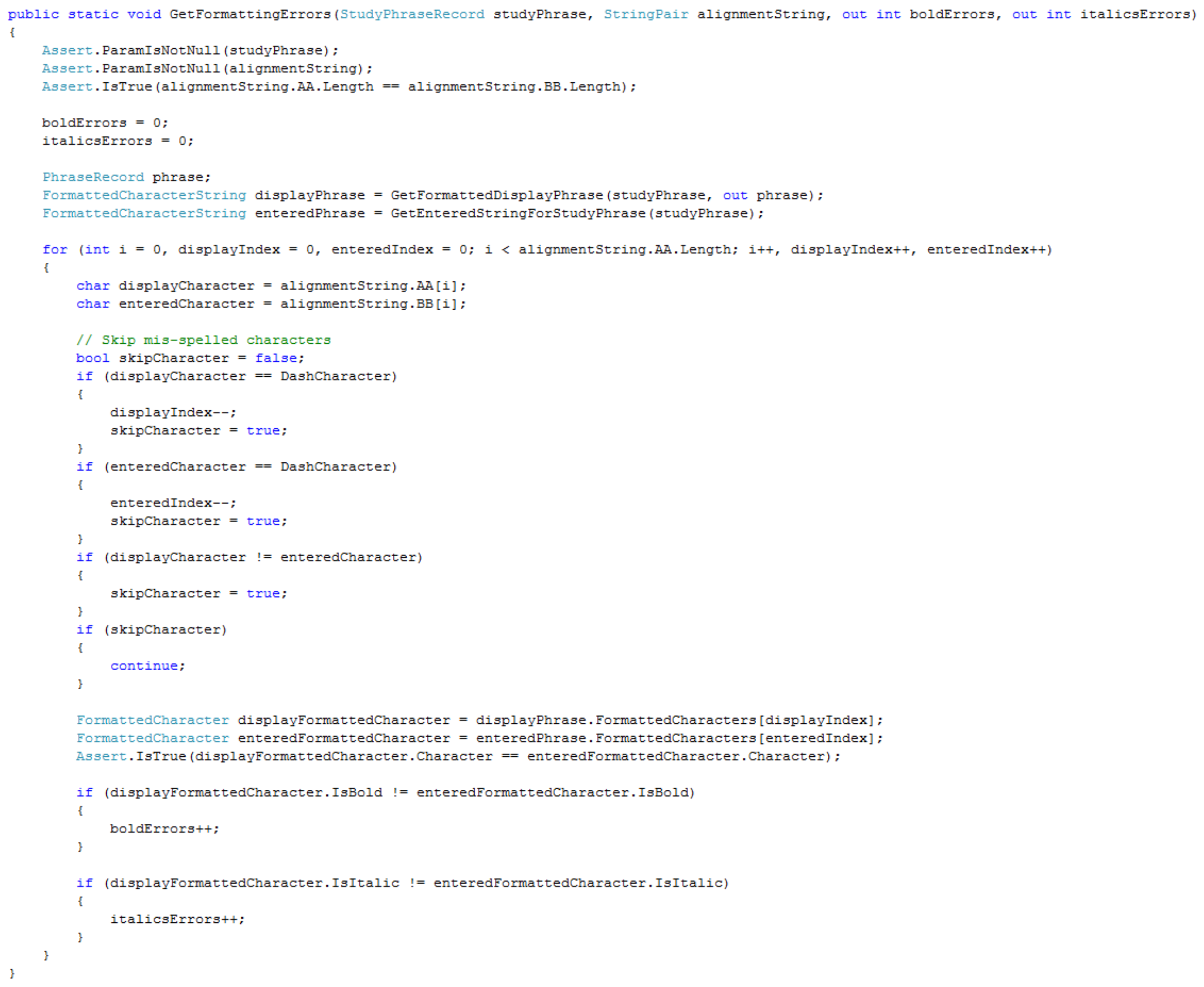


Figure 19 Word processing study formatting meta-data database schema

In this database design, an entry in the *StudyPhrase* table indicates a particular paragraph that a participant has transcribed. Entries in the *StudyPhraseCharacter* table record the input stream – each key pressed by the participant – even if those characters do not transfer to the final version of the transcribed text (i.e. the user removes characters with correction keys). The *StudyPhraseFormattedCharater* table stores the characters within final transcribed phrase, including Boolean values indicating the bold or italics formatting of the character.

While the *StudyPhraseCharacter* table is populated in real-time, as the participant enters text, the *StudyPhraseFormattedCharacter* table is filled only when the user commits a phrase to the database. At that time, we parse the rich text field containing the final version of the transcribed paragraph, and insert records in the *StudyPhraseFormattedCharacter* table. Note that the measurements for formatting errors depend on this data, but the method in which we calculate those metrics can vary. In our implementation, we chose to measure formatting errors *character-by-character*, but we suggested alternatives such as *consecutive-character*, *whole-word*, or *whole-consecutive word* analysis. Storing the formatting meta-data, for each character within the transcribed text, enables the exploration of these alternate measurements. Figure 20 presents the algorithm we developed for measuring bold and italics errors using the *character-by-character* method.



```
public static void GetFormattingErrors(StudyPhraseRecord studyPhrase, StringPair alignmentString, out int boldErrors, out int italicsErrors)
{
    Assert.ParamIsNotNull(studyPhrase);
    Assert.ParamIsNotNull(alignmentString);
    Assert.IsTrue(alignmentString.AA.Length == alignmentString.BB.Length);

    boldErrors = 0;
    italicsErrors = 0;

    PhraseRecord phrase;
    FormattedCharacterString displayPhrase = GetFormattedDisplayPhrase(studyPhrase, out phrase);
    FormattedCharacterString enteredPhrase = GetEnteredStringForStudyPhrase(studyPhrase);

    for (int i = 0, displayIndex = 0, enteredIndex = 0; i < alignmentString.AA.Length; i++, displayIndex++, enteredIndex++)
    {
        char displayCharacter = alignmentString.AA[i];
        char enteredCharacter = alignmentString.BB[i];

        // Skip mis-spelled characters
        bool skipCharacter = false;
        if (displayCharacter == DashCharacter)
        {
            displayIndex--;
            skipCharacter = true;
        }
        if (enteredCharacter == DashCharacter)
        {
            enteredIndex--;
            skipCharacter = true;
        }
        if (displayCharacter != enteredCharacter)
        {
            skipCharacter = true;
        }
        if (skipCharacter)
        {
            continue;
        }

        FormattedCharacter displayFormattedCharacter = displayPhrase.FormattedCharacters[displayIndex];
        FormattedCharacter enteredFormattedCharacter = enteredPhrase.FormattedCharacters[enteredIndex];
        Assert.IsTrue(displayFormattedCharacter.Character == enteredFormattedCharacter.Character);

        if (displayFormattedCharacter.IsBold != enteredFormattedCharacter.IsBold)
        {
            boldErrors++;
        }

        if (displayFormattedCharacter.IsItalic != enteredFormattedCharacter.IsItalic)
        {
            italicsErrors++;
        }
    }
}
```

Figure 20 Word processing study formatting error algorithm

5.4.2 Rich Text Parsing

Rich text denotes plain text that has been augmented with formatting information, and is commonly stored in the Rich Text Format (RTF) [53] – a proprietary specification developed by Microsoft for cross-platform document interchange. RTF is the format we employed to present our rich text paragraphs along with those transcribed by the participants. The participants required no knowledge of this specification; its use was entirely hidden from view. We comment on RTF here to assist future experimenters who may wish to replicate this experiment.

Storing the formatted text in RTF was natural, given our chosen software development environment (C# and .NET). This environment provides rich text controls that can by employed in a custom software application. Indeed, the Microsoft-provided rich text field can serialize and deserialize rich text very easily and offers great flexibility for customization – a necessity given our requirement to interact with the control via multi-touch. One consideration, however, when using RTF – and one which was not immediately evident – was the large amount of disparity between applications that created and consumed RTF. Specifically, RTF-enabled applications have varying support for the *version* of the RTF specification they support and also how much superfluous data they write when serializing a document in RTF format. Microsoft Word, for example, generates an enormous amount of RTF data for a given file, as compared to Microsoft WordPad or the Microsoft .NET rich text control. To illustrate this point, we compared the RTF file stored for our *A George Divided* paragraph (presented in Appendix B) and found the version constructed using Microsoft WordPad had a file size of 880 bytes, while the Microsoft Word version was 33,116 bytes. Visually, these files were identical, but the amount of disparity between the RTF authoring tools is incredible. If one merely presented RTF data visually, then this consideration is potentially minor. However, when programmatically accessing the data, such disparity poses a significant hurdle.

As mentioned, our implementation parses the RTF data, stored in memory, once the participant commits a transcribed phrase to the database. Thus, programmatically accessing the RTF is necessary in order to extract the required formatting information for the associated text. To our knowledge, there is no universal rich text parsing software available, or free of charge. Thus, we employed an open-source RTF parsing library, and unfortunately this system did not cope well with the heap of superfluous RTF information stored within Microsoft Word-generated RTF files. Moreover, the parsing library failed on some of the transcribed paragraphs entered by our participants through the .NET rich text control.

The lesson we wish to convey is thus: 1) know the type of RTF your authoring system generates; and 2) test your RTF parsing system exhaustively. To this second point, after discovering that the RTF parsing failed occasionally during pilot testing, we began saving the transcribed RTF data to disk before parsing. Thus, if the parsing system failed, we wouldn’t necessarily lose a participant’s data. Instead, we could attempt to debug the parsing error post-process.

CHAPTER SIX: WORD PROCESSING PLATFORM STUDY

6.1 Introduction

In this study we continue the examination of word processing on a multi-touch platform, leveraging the direction provided by the previous study. This experiment is similar to the very first text entry study, in that we wish to establish performance differences between the multi-touch platform and a traditional mouse and keyboard.

In particular, this study analyzes word processing on three computing platforms: a desktop computer with keyboard and mouse, a laptop with its integrated keyboard and trackpad, and a multi-touch platform.

6.2 Experiment

This experiment examines the performance of a word processing task between multiple computing platforms. As with the first word processing study, participants are presented a series of rich-text paragraphs and asked to transcribe this text along with punctuation, organization, and formatting. Software records the corresponding input stream for subsequent analysis.

6.2.1 Materials and Methods

<u>6.2.1.1 Participants</u>

Eighteen unpaid volunteers participated in this study (3 females, 15 males). All were right-handed and had no visual or physical impairments that prevented normal computer use (as reported by the participants). Their ages ranged from 22 to 42 with an average age of 26. The participants rated their familiarity with touch screen computer systems (including single-touch or multi-touch) from 1-5: 5 highest; their responses were 6%, 11%, 0%, 22%, and 61%, respectively.

<u>6.2.1.2 Apparatus</u>

Our three computing platforms denote separate interaction devices: a desktop keyboard and mouse; a laptop with integrated keyboard and trackpad; and a multi-touch monitor. Hereafter, we will refer to these platforms as *Desktop*, *Laptop*, and *Multi-Touch*, respectively. As with each of the previous experiments, this study employed a 3M M2256PW [3] multi-touch monitor pictured in Chapter 3, Figure 2. The *Physical* platform devices were a Dell SK-8135 keyboard, pictured in Chapter 3 Figure 1, and a Microsoft Wireless mouse pictured in Figure 21.

Figure 21 Microsoft wireless mouse used in word processing platform study

All of these devices were attached to a Dell Precision M6400 laptop pictured in Figure 22. The laptop's integrated keyboard and trackpad comprise the interaction devices used for the *Laptop* platform. Indeed, this laptop was the single computer system used during the experiment. Participants switched between interaction devices, as called for by the study, but the computer itself was common to all three platforms. The Dell Precision M6400's keyboard has 101 keys including a full number pad with a size of approximately 14.5”x5.0”. The laptop’s trackpad is roughly 3.125”x2.875” including buttons for the left-, middle-, and right-click functions.

Figure 22 Dell Precision M6400 laptop used in word processing platform study

Different monitors were employed for each of the platforms: the multi-touch monitor, a desktop-style flat screen monitor, and the integrated monitor of the laptop. All of the monitors operated at a resolution of 1680x1050 pixels.

6.2.1.3 Software

A custom C# software application was developed to collect data for our study. This application is functionally identical to that used in the previous multi-touch word processing study, and pictured in Chapter 5 Figures 15 and 16. However, the associated database was modified to accommodate the updated experiment design. The pre- and post-study questionnaire applications, from the original study, were likewise adopted and modified to match the questions asked in this experiment.

6.2.1.4 Procedure

Participants were assigned a starting platform through a Latin Square. Depending on the specified platform, participants sat at the desk in either a standard office chair or a drafting-style chair to accommodate the height of the multi-touch monitor. Participants were instructed to adjust their chair, the position of the keyboard/mouse/monitor, and the orientation of these devices to best suit themselves.

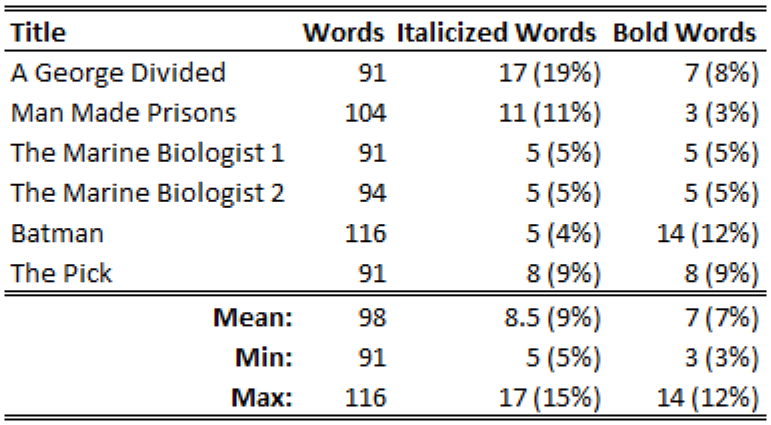
Once seated at the desk, participants entered two paragraphs on the specified platform. Afterwards, the participants and interaction devices were re-situated and the participants transcribed two additional paragraphs for the remaining platforms.

Participants were instructed to type “as quickly as possible as accurately as possible” and were free to transcribe and format text in the manner most comfortable for them.

The six paragraphs were taken from the television series *Seinfeld* – the same four as with the first word processing study, along with two additional paragraphs – and are presented in Appendix B. All participants completed the same six paragraphs; however, the order in which the paragraphs were presented was balanced using three Latin Squares, one for each platform ordering. Each paragraph contained capitalization, punctuation, new lines, and bold and italics formatting. A summary of the makeup of each paragraph is presented in Table 14.

After finishing the experiment, participants completed a post-study questionnaire.

Table 14 Word processing platform study paragraph summary



Title	**Words**	**Italicized Words**	**Bold Words**
A George Divided	91	17 (19%)	7 (8%)
Man Made Prisons	104	11 (11%)	3 (3%)
The Marine Biologist 1	91	5 (5%)	5 (5%)
The Marine Biologist 2	94	5 (5%)	5 (5%)
Batman	116	5 (4%)	14 (12%)
The Pick	91	8 (9%)	8 (9%)
Mean:	98	8.5 (9%)	7 (7%)
Min:	91	5 (5%)	3 (3%)
Max:	116	17 (15%)	14 (12%)

6.2.1.5 Environment

The environment of the study was the same room as was used in the first three studies – a small room housing the multi-touch platform (pictured in Chapter 3, Figure 6).

6.2.1.6 Design

This experiment employs a within-subjects single-factor design, where the factor is *Platform*, with levels: *Desktop*, *Laptop*, and *Multi-Touch* (Direct Interaction). As mentioned in Section 6.2.1.4*,* the order in which the participants performed on the three platforms was balanced through a Latin Square. Thus, 6 participants performed each of the unique platform orderings. In this fashion, we counterbalanced the conditions to mitigate learning effects.

Each participant completed the experiment in approximately 35 minutes.

6.2.2 Metrics

We employed the same metrics as were described in Section 5.2.2. This included the text entry metrics: Words per Minute (WPM), Key Strokes per Second (KSPC), Total Error Rate, Not Corrected Error Rate, Corrected Error Rate, and Conscientiousness. Likewise, this studies metrics included the word processing metrics: Bold Errors, Italics Errors, Total Time, Total Key Stroke Time, and Completion Time.

In addition to the quantitative measures, we asked each of the participants to respond to the following statements in a post-study questionnaire:

- The multi-touch platform accurately detected my intended key presses
- The multi-touch platform accurately detected my intended mouse interactions
- Rank your preferred word processing platform as (1, 2, 3; 1 highest)
- Would you use the multi-touch device for everyday word processing activities?

6.2.3 Results

6.2.3.1 Text Entry Metrics

In total, participants entered 108 paragraphs of formatted text: 36 for each of the 3 platforms; 6 for each of the 18 participants. The means and standard deviations for each platform and the 6 chief text entry metrics are summarized in Table 15.

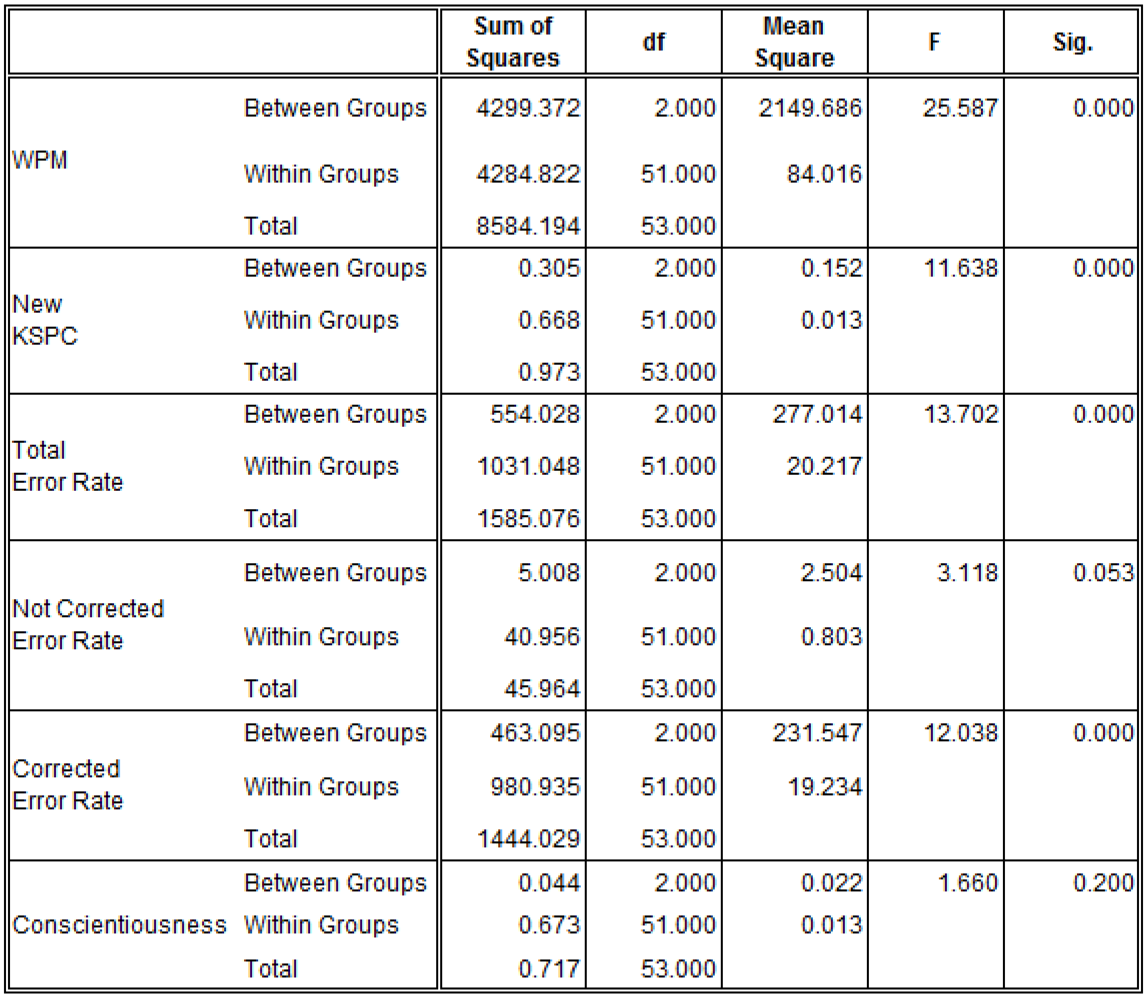
Table 15 Word processing platform study summary of results for text entry metrics

	WPM		New KSPC		Total Error Rate (%)		Not Corrected Error Rate (%)		Corrected Error Rate (%)		Conscientiousness	
Platform	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Desktop	42.81	10.81	1.13	0.07	6.85	3.38	1.08	0.89	5.77	3.25	0.81	0.16
Laptop	41.22	10.41	1.14	0.09	7.29	3.96	0.81	0.68	6.48	3.91	0.86	0.08
MultiTouch	23.14	5.18	1.29	0.16	13.85	5.79	1.55	1.08	12.30	5.64	0.87	0.08

To analyze the quantitative data, we ran a one-way ANOVA, on each text entry metric. Additionally, we ran performed a Bonferroni [30] post-hoc analysis, considering the three platform combinations: Desktop-to-Laptop, Desktop-Multi-Touch, and Laptop-to-MultiTouch. We found no statistically significant differences, within any of the metrics, between the desktop and laptop platforms. Comparing the desktop and multi-touch systems, we found significant differences for all of the metrics except for *Not Corrected*

Error Rate and *Conscientiousness*. Similarly, we found significant differences between the Laptop and multi-touch platforms for all but the *Conscientiousness* metrics. The results are presented in Tables 16-18.

Table 16 Word processing platform study analysis results for text entry metrics



		Sum of Squares	df	Mean Square	F	Sig.
WPM	Between Groups	4299.372	2.000	2149.686	25.587	0.000
	Within Groups	4284.822	51.000	84.016		
	Total	8584.194	53.000			
New KSPC	Between Groups	0.305	2.000	0.152	11.638	0.000
	Within Groups	0.668	51.000	0.013		
	Total	0.973	53.000			
Total Error Rate	Between Groups	554.028	2.000	277.014	13.702	0.000
	Within Groups	1031.048	51.000	20.217		
	Total	1585.076	53.000			
Not Corrected Error Rate	Between Groups	5.008	2.000	2.504	3.118	0.053
	Within Groups	40.956	51.000	0.803		
	Total	45.964	53.000			
Corrected Error Rate	Between Groups	463.095	2.000	231.547	12.038	0.000
	Within Groups	980.935	51.000	19.234		
	Total	1444.029	53.000			
Conscientiousness	Between Groups	0.044	2.000	0.022	1.660	0.200
	Within Groups	0.673	51.000	0.013		
	Total	0.717	53.000			

Table 17 Word processing platform study post-hoc analysis results for text entry metrics (WPM, New KSPC, and Total Error Rate)

						Interval	
Dependent Variable	(I) Platform	(J) Platform	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
WPM	Desktop	Laptop	1.586	3.055	1.000	-5.978	9.149
		MultiTouch	19.671	3.055	.000	12.108	27.235
	Laptop	Desktop	-1.586	3.055	1.000	-9.149	5.978
		MultiTouch	18.086	3.055	.000	10.522	25.649
	MultiTouch	Desktop	-19.671	3.055	.000	-27.235	-12.108
		Laptop	-18.086	3.055	.000	-25.649	-10.522
New KSPC	Desktop	Laptop	-.018	.038	1.000	-.112	.077
		MultiTouch	-.167	.038	.000	-.262	-.073
	Laptop	Desktop	.018	.038	1.000	-.077	.112
		MultiTouch	-.150	.038	.001	-.244	-.055
	MultiTouch	Desktop	.167	.038	.000	.073	.262
		Laptop	.150	.038	.001	.055	.244
Total ErrorRate	Desktop	Laptop	-.441	1.499	1.000	-4.152	3.269
		MultiTouch	-7.005	1.499	.000	-10.715	-3.295
	Laptop	Desktop	.441	1.499	1.000	-3.269	4.152
		MultiTouch	-6.563	1.499	.000	-10.274	-2.853
	MultiTouch	Desktop	7.005	1.499	.000	3.295	10.715
		Laptop	6.563	1.499	.000	2.853	10.274

Table 18 Word processing platform study post-hoc analysis results for text entry metrics (Not Corrected Error Rate, Corrected Error Rate, and Conscientiousness)

Dependent Variable	(I) Platform	(J) Platform	Mean Difference (I-J)	Std. Error	Sig.	Interval	
						Lower Bound	Upper Bound
Not Corrected Error Rate	Desktop	Laptop	.269	.299	1.000	-.470	1.009
		MultiTouch	-.468	.299	.371	-1.207	.272
	Laptop	Desktop	-.269	.299	1.000	-1.009	.470
		MultiTouch	-.737	.299	.051	-1.477	.002
	MultiTouch	Desktop	.468	.299	.371	-.272	1.207
		Laptop	.737	.299	.051	-.002	1.477
Corrected Error Rate	Desktop	Laptop	-.711	1.462	1.000	-4.330	2.908
		MultiTouch	-6.537	1.462	.000	-10.156	-2.918
	Laptop	Desktop	.711	1.462	1.000	-2.908	4.330
		MultiTouch	-5.826	1.462	.001	-9.445	-2.207
	MultiTouch	Desktop	6.537	1.462	.000	2.918	10.156
		Laptop	5.826	1.462	.001	2.207	9.445
Conscientiousness	Desktop	Laptop	-.056	.038	.443	-.151	.038
		MultiTouch	-.064	.038	.305	-.159	.031
	Laptop	Desktop	.056	.038	.443	-.038	.151
		MultiTouch	-.008	.038	1.000	-.102	.087
	MultiTouch	Desktop	.064	.038	.305	-.031	.159
		Laptop	.008	.038	1.000	-.087	.102

6.2.3.2 Word Processing Metrics

As with the word processing technique study, we have grouped the *Bold Errors, Italics Errors, Total Time, Total Key Stroke Time,* and *Completion Time* metrics into a group titled *Word Processing Metrics*. The means and standard deviations of these metrics, for each platform, are summarized in Table 19.

Table 19 Word processing platform study summary of results for word processing metrics

	Bold Errors		Italics Errors		Total Time		Total Key Stroke		Completion Time	
Platform	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
Desktop	0.61	1.09	3.17	3.55	157.39	34.76	149.34	33.51	8.06	7.21
Laptop	0.44	0.70	4.28	8.17	162.93	35.50	152.46	32.96	10.47	11.45
MultiTouch	1.72	4.13	3.83	4.30	297.03	85.68	281.54	77.16	15.50	18.61

To analyze this data, we ran a-way ANOVA on each word processing metric. Additionally, we ran performed a Bonferroni [30] post-hoc analysis, considering the three platform combinations: Desktop-to-Laptop, Desktop-Multi-Touch, and Laptop-to-MultiTouch.

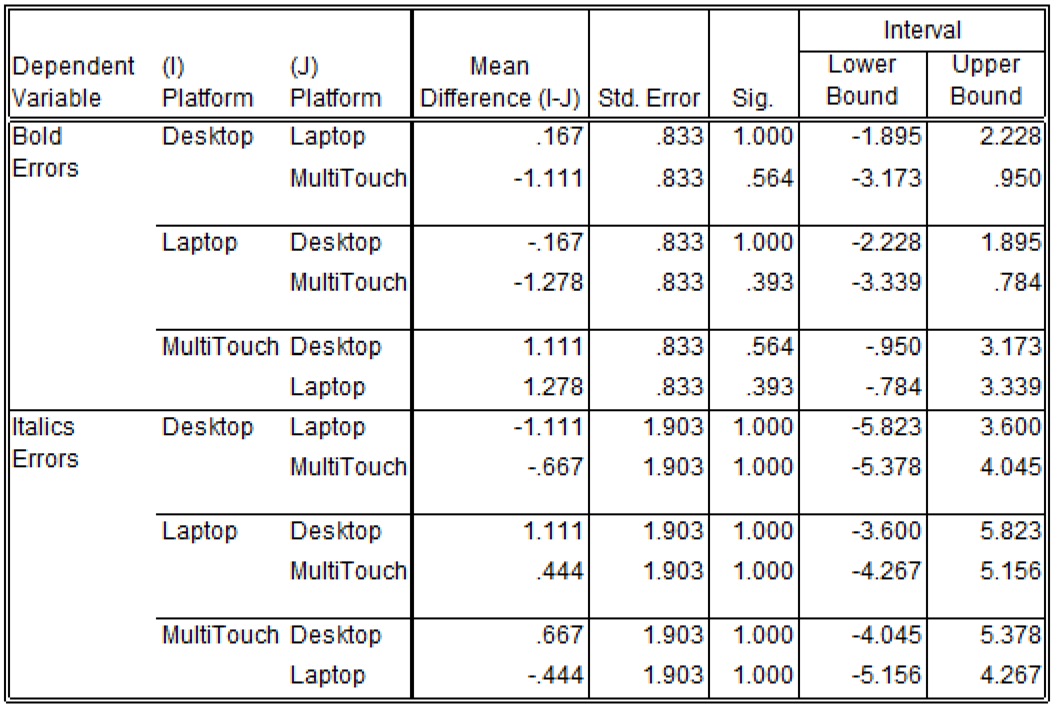
As with the text entry metrics, we found no statistically significant differences, within any of the word processing metrics, between the desktop and laptop platforms. We did, however, find significant differences for the *Total* Time and *Total Keystroke*

Time metrics between both the desktop-to-multi-touch and laptop-to-multi-touch comparisons. The results are presented in Tables 20-22.

Table 20 Word processing platform study analysis results for word processing metrics

		Sum of Squares	df	Mean Square	F	Sig.
Bold Errors	Between Groups	17.370	2.000	8.685	1.391	0.258
	Within Groups	318.333	51.000	6.242		
	Total	335.704	53.000			
Italics Errors	Between Groups	11.259	2.000	5.630	0.173	0.842
	Within Groups	1662.611	51.000	32.600		
	Total	1673.870	53.000			
Total Time	Between Groups	225081.603	2.000	112540.802	34.421	0.000
	Within Groups	166745.392	51.000	3269.517		
	Total	391826.995	53.000			
Total Key Stroke Time	Between Groups	204882.857	2.000	102441.428	37.644	0.000
	Within Groups	138785.779	51.000	2721.290		
	Total	343668.636	53.000			
Completion Time	Between Groups	518.614	2.000	259.307	1.469	0.240
	Within Groups	9000.369	51.000	176.478		
	Total	9518.983	53.000			

Table 21 Word processing platform study post-hoc analysis results for word processing metrics (Bold Errors and Italics Errors)



						Interval	
Dependent Variable	(I) Platform	(J) Platform	Mean Difference (I-J)	Std. Error	Sig.	Lower Bound	Upper Bound
Bold Errors	Desktop	Laptop	.167	.833	1.000	-1.895	2.228
		MultiTouch	-1.111	.833	.564	-3.173	.950
	Laptop	Desktop	-.167	.833	1.000	-2.228	1.895
		MultiTouch	-1.278	.833	.393	-3.339	.784
	MultiTouch	Desktop	1.111	.833	.564	-.950	3.173
		Laptop	1.278	.833	.393	-.784	3.339
Italics Errors	Desktop	Laptop	-1.111	1.903	1.000	-5.823	3.600
		MultiTouch	-.667	1.903	1.000	-5.378	4.045
	Laptop	Desktop	1.111	1.903	1.000	-3.600	5.823
		MultiTouch	.444	1.903	1.000	-4.267	5.156
	MultiTouch	Desktop	.667	1.903	1.000	-4.045	5.378
		Laptop	-.444	1.903	1.000	-5.156	4.267

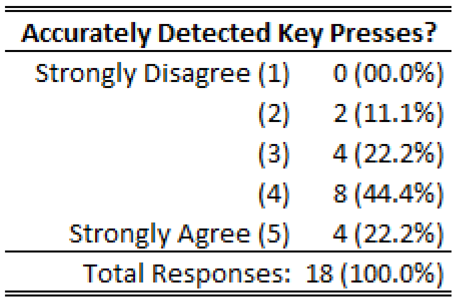
Table 22 Word processing platform study post-hoc analysis results for word processing metrics (Total Time, Total Key Stroke Time, and Completion Time)

Dependent Variable	(I) Platform	(J) Platform	Mean Difference (I-J)	Std. Error	Sig.	Interval	
						Lower Bound	Upper Bound
Total Time	Desktop	Laptop	-5.538	19.060	1.000	-52.721	41.645
		MultiTouch	-139.641	19.060	.000	-186.824	-92.458
	Laptop	Desktop	5.538	19.060	1.000	-41.645	52.721
		MultiTouch	-134.102	19.060	.000	-181.285	-86.919
	MultiTouch	Desktop	139.641	19.060	.000	92.458	186.824
		Laptop	134.102	19.060	.000	86.919	181.285
Total Key Stroke Time	Desktop	Laptop	-3.127	17.389	1.000	-46.173	39.919
		MultiTouch	-132.201	17.389	.000	-175.247	-89.156
	Laptop	Desktop	3.127	17.389	1.000	-39.919	46.173
		MultiTouch	-129.074	17.389	.000	-172.120	-86.028
	MultiTouch	Desktop	132.201	17.389	.000	89.156	175.247
		Laptop	129.074	17.389	.000	86.028	172.120
Completion Time	Desktop	Laptop	-2.411	4.428	1.000	-13.373	8.551
		MultiTouch	-7.439	4.428	.297	-18.401	3.523
	Laptop	Desktop	2.411	4.428	1.000	-8.551	13.373
		MultiTouch	-5.028	4.428	.784	-15.990	5.934
	MultiTouch	Desktop	7.439	4.428	.297	-3.523	18.401
		Laptop	5.028	4.428	.784	-5.934	15.990

6.2.3.3 Post Study Questionnaire Data

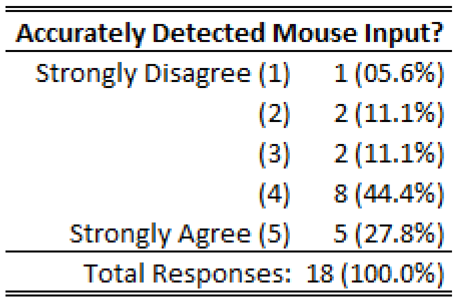
In the post-study questionnaire, we asked each participant Likert-scale questions pertaining to the accurate detection of intended key presses and mouse input. Tables 23 and 24 present the results of these questions, where the value 1 indicates *Strongly Disagree* and 5, *Strongly Agree.*

Table 23 Word processing platform study questionnaire responses on accurately detected key presses



Accurately Detected Key Presses?	
Strongly Disagree (1)	0 (00.0%)
(2)	2 (11.1%)
(3)	4 (22.2%)
(4)	8 (44.4%)
Strongly Agree (5)	4 (22.2%)
Total Responses:	18 (100.0%)

Table 24 Word processing platform study questionnaire responses on accurately detected mouse input



Accurately Detected Mouse Input?	
Strongly Disagree (1)	1 (05.6%)
(2)	2 (11.1%)
(3)	2 (11.1%)
(4)	8 (44.4%)
Strongly Agree (5)	5 (27.8%)
Total Responses:	18 (100.0%)

We also asked participants to rank their preferred word processing platform. The results are presented in Figure 23 and clearly show that the desktop platform was the most preferred, with the laptop and multi-touch taking the second and third rankings, respectively. We found statistical significance for each preference ranking (Most Preferred: $\chi_2^2 = 13.0,\ p < 0.05$,Mid: , $\chi_2^2 = 6.333,\ p < 0.05$ Least Preferred: $\chi_2^2 = 16.333, p < 0.05$).

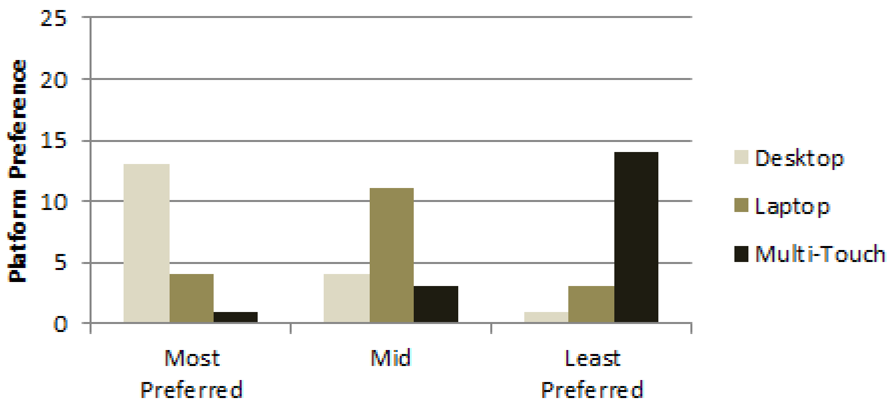


Figure 23 Word processing platform study questionnaire responses on method preference

The final subjective question asked if the participant "would use the multi-touch device to for everyday word processing activities." The responses to this question are presented in Figure 24. We found no statistical significance within this data ($\chi_1^2 = 5.765, p = 0.056$) at a 95% confidence interval.

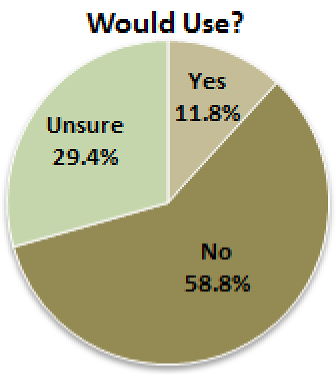


Figure 24 Word processing platform study questionnaire responses on "would use" multi-touch platform to for everyday word processing activities

6.2.4 Discussion

The results of this study are generally unsurprising – the desktop and laptop platforms outperform the multi-touch platform for our word processing task. Specifically, except for the *Not Corrected Error Rate* metric (whose statistical significance was split between the desktop-to-multi-touch and laptop-to-multi-touch comparisons), *Conscientiousness* was the sole text entry metric that held no significant difference between the three platforms. Thus, we can state that participants were just as conscientious about fixing typographical errors irrespective of platform. Otherwise, the multi-touch platform underperformed the desktop and laptop systems on every other text entry metric.

What was moderately surprising was that there were no significant differences between the desktop and laptop platforms. We expected that the desktop would outperform the laptop, and it did not.

The word processing metrics *Total Time* and *Total Keystroke Time* add support to the text entry performance metrics, as they too reveal that the desktop and laptop platforms outperformed the multi-touch system. However, the remaining word processing metrics: *Bold Errors*, *Italics Errors*, and *Completion Time* showed no significant differences between any of the three platforms. Thus we could conclude that (for this task) the multi-touch platform performed at least as well as the traditional platforms with respect to formatting and post-transcription validation.

As with the first user study on multi-touch text entry, we intuitively expected that the desktop and laptop systems would outperform the multi-touch platform. Our question was always “by how much?” The results of this experiment establish a baseline from which to evaluate future efforts in word processing on a multi-touch platform.

Furthermore, the post-study questionnaire responses offer some support for the usefulness of the multi-touch platform as a word processing device. 2 of the 18 respondents (11.8%) stated that they would use the multi-touch platform for everyday word processing activities. Although this is decidedly lower than the 10 participants (58.8%) who explicitly said that they would not use the platform, the number is not zero. Moreover, 5 participants (29.4%) were unsure if they would use the platform or not, offering hope that they *might* use the multi-touch platform. Additional support comes from the platform preference ranking. 1 participant (6%) ranked the multi-touch platform as their most preferred system for the word processing task and 3 participants (17%) selected the platform for their second rank.

Interestingly, the “would use” results could be considered worse than those collected for the word processing technique study. In that study, 25% of the participants (3 of 12) indicated that they would use the multi-touch platform for everyday word processing activities compared to the 11.8% (2 of 18) in this study. However, we must consider that, in the technique study, the multi-touch system was not directly pitted against any other platform. This may have acted to skew the results of the “would use” question in the multi-touch platform’s favor.

Additionally, the “would use” results are lower than the 18 of 47 participants (38%), from the original text entry study, who said that they would use the multi-touch platform to enter text for everyday computer activities. These results imply that fewer people would use the multi-touch platform for word processing than they would for plain-text entry. We suspect that the transcribed paragraphs contribute to this trend. Specifically, the phrases transcribed within the original text entry study were short (an average of 6 words in length) and contained no punctuation. Conversely, the paragraphs used in the word processing studies were fully punctuated. And while the total amount of transcription was roughly the same between studies, the individual paragraphs were much longer (approximately 100 words apiece) than the individual plain-text phrases. An interesting follow-up study would be to examine the “would use” statistics for plain-text entry against the same paragraphs (un-formatted) as were used in the word processing studies.

Finally, we feel that we’ve introduced bias *against* the multi-touch platform within this and the word processing technique studies. Specifically, we have taken pains to replicate the GUI-style interface systems employed by traditional word-processing systems. We have explicitly not leveraged multi-touch-specific features including, for example, gestural interaction. One could imagine text formatting and/or text selecting accomplished through gestures and not at all performed through button-style interaction. While the word processing technique study indicated that the *Direct Interaction* technique was preferred over *Indirect Interaction* for GUI-style button pressing, this is certainly not the only technique one could employ to trigger rich text formatting. Perhaps leveraging multi-touch-specific features would improve the adoption potential of the platform for word processing tasks.

6.3 Summary

In this study, we examined the performance of word processing on a multi-touch platform and compared it against traditional desktop and laptop computer systems. Similar to our very first text entry experiment, our goal in this study was to establish baseline performance measurements from which we could explore improvements in multi-touch word processing.

The results of our experiment were unsurprising – the traditional desktop and laptop platforms outperformed the multi-touch system; but this experiment was primarily concerned with the question “by how much?” And, furthermore, “how does the inclusion of mouse-style interaction affect someone’s acceptance of the multi-touch platform as a general-purpose computing device?”

Our results indicate that fewer people would adopt the multi-touch platform for word processing, as compared to plain-text entry. However, we suspect that these opinions might be skewed by the differences in punctuation between the two experiments.

6.4 Notes on Replication

By leveraging our experiences from the first three studies, we avoided most of the issues discussed in the *Notes on Replication* sections of those chapters. However, one new issue appeared that we feel warrants discussion. In this experiment, as with the Word Processing Technique study, we employed the Minimum String Distance (MSD) algorithms developed by Soukoreff and MacKenzie [68] and discussed in Chapter 3.2.2. In particular, we reused the notion of optimal alignment strings to identify the constituents of the input stream – namely, Correct (C), Incorrect Not Fixed (INF), Incorrect Fixed (IF), or Fix Keys (F).

The paragraphs used in the Word Processing Technique and Platform studies are approximately 100 words apiece – significantly longer than the 6 word phrases used in the two text entry studies. Thus, the set of optimal alignment strings was larger – sometimes exceedingly larger – than any of the sets generated for the first two experiments. For most of the transcribed phrases in the Word Processing Platform study, this was not an issue. Indeed, we did not uncover this error in the Word Processing Technique study (presumably because we only collected 48 paragraphs in that experiment). However, for 5 of the 108 transcribed paragraphs in this fourth study, the input was so grossly in error that the alignment set was prohibitively large (e.g. greater than 100,000 alignment strings). Visually scanning the input-in-question, we found myriad extra spaces, typos, and even whole words or sentences transposed. If we could have broken the input into smaller chunks, the set of alignment strings would have been manageable; but taken as a whole, the set becomes unwieldy, to the point that we considered the data invalid.

CHAPTER SEVEN: ADDITIONAL INVESTIGATIONS

In addition to the studies presented in Chapters 3 through 6, we have pursued a number of additional topics in the area of general-purpose multi-touch computing. This chapter describes some of the more noteworthy of these efforts.

7.1 General-Purpose Multi-Touch Platform (GPMT)

Throughout our investigations into general-purpose computing and multi-touch, we have speculated on what such a platform might look like. As previously mentioned, multi-touch platforms, to date, have not targeted general-purpose computing. Their form factors have most frequently accommodated mobile computing (e.g. smart phones) or multi-user collaboration (e.g. tabletops). Our overarching goal seeks to determine the viability of multi-touch as a single-user general-purpose computing platform – one which would operate in the place of today's traditional desktop computer. In this section, we present our vision of such a general-purpose multi-touch platform (GPMT).

There have been several commercially successful platforms that work in the direction of a GPMT computing platform. The platform we might consider the "spiritual predecessor" to a GPMT platform is the Tablet PC [91]. Made popular by Microsoft in

2001, a Tablet PC is a mobile computing device equipped with a touchscreen and/or a stylus. Essentially, Tablet PCs add pen-based user interaction to a laptop computing platform, enabling handwriting capture and 2D gestural input. Though mobile, Tablet PCs typically fall under the previously described “desktop” computing category, because of their processing power and capability as general-purpose computers. Tablet PCs do not replace the mouse and keyboard paradigm, and instead augment this interaction capability with a stylus/touchscreen. Indeed, many Tablet PCs support a physical keyboard either directly integrated or through a detachable peripheral. Tablet PCs fall short of our definition of a general-purpose multi-touch platform because they typically do not support multiple points of simultaneous contact.

A second platform, similar in nature to the GPMT, is the HP TouchSmart [31]. This device is a desktop computer with a traditional mouse and keyboard, but also has a touchscreen display that allows for two simultaneous points of contact. This display is oriented vertically, in the fashion of a traditional monitor, and the user can perform simple gestures to interact with the computer. HP provides a multi-touch “shell” which encapsulates multi-touch friendly applications, and a mouse emulator for interacting with the Microsoft Windows desktop. While this is an interesting device, which provides visibility for multi-touch into the home, it perhaps should not be considered a bona-fide multi-touch platform. We make this statement primarily because the touchscreen is not the device’s primary mechanism for interaction – which remains the mouse and keyboard. This is more akin to the Tablet PC, which augments the physical mouse and keyboard with an additional input modality. Furthermore, we suggest that the vertical orientation of the touchscreen prohibits long-term-single-session use and that only two points of simultaneous contact limits the usefulness of the multi-touch display.

More commonly considered as true multi-touch platforms are tabletop sized displays – which are often designed to accommodate multiple users. Such devices include the Mitsubishi DiamondTouch [14] and the Microsoft Surface [23]. Although work exists documenting such platforms for individual use [87], these systems have typically targeted special-purpose applications and not daily computer use. We speculate that the size and orientation of these devices is the dominant factor with respect to their applicability to general-purpose computing. Specifically, these platforms do not integrate well into the typical workstation – that of a desk and chair. Indeed, if at all viable as an individual's workstation, such tabletop platforms serve as the desk itself, as with [87] where the multi-touch platform was mounted akin to a drafting table.

7.1.1 Design

Our design of a GPMT platform is pictured in Figure 32. This pictorial is meant to convey a proximal multi-touch device (PMTD) that is positioned and oriented in a fashion similar to a conventional keyboard. The stylus positioned in front of the PMTD denotes the pen-based interaction allowed with this platform, though the user is intended to use his/her hands as the primary interaction mechanism. The distal display is either a traditional, non-interactive flat-screen monitor, or a single-touch or multi-touch display. The distal display is positioned and oriented similar to a conventional mouse+keyboard+monitor platform and serves as the primary display. Thus, the posture of the user is in the same fashion as a traditional computing platform – the user’s hands rest before him/her and his/her visual focus remains orthogonal to the input device. As with a conventional computing platform, the user is expected to periodically glance down at the input device – which in this design serves as a secondary display as well – but his/her attention is primarily on the distal display.

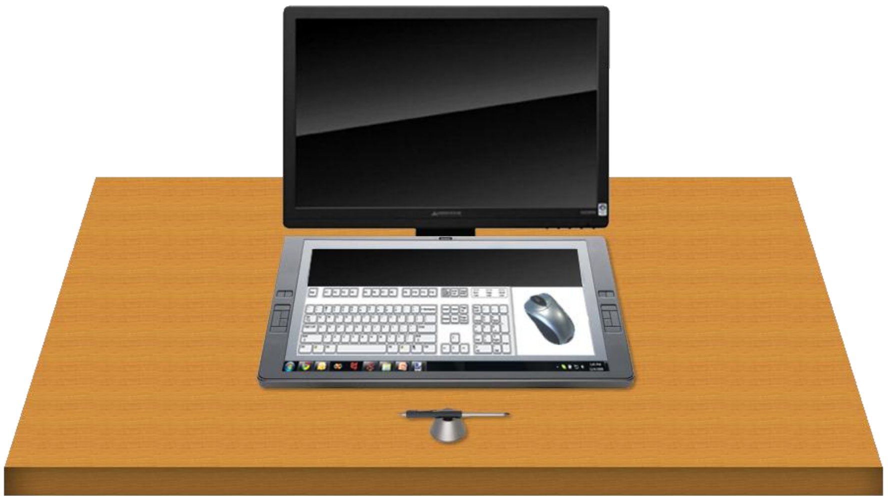


Figure 25 General-purpose multi-touch design concept

A closer look at the layout of the proximal multi-touch device is pictured in Figure 26. This device is intended to integrate the input modalities of keyboard, mouse, pen-based interaction, and gesturing into a single hardware system. The device is approximately 22” wide, 17” high and 2” deep with an adjustable-angle stand. There are three primary interaction areas, each providing a different input paradigm. These are the keyboard, mouse, and gestural interaction areas depicted in the image. The pictured layout is positioned for right-handed use; however, because the platform is entirely software-driven, user customization is easily accomplished. Indeed, the choice of keyboard layout, geographical region, size, and position on screen can all be configured by the user. The mouse style interaction area functions in a fashion similar to a laptop, multi-touch trackpad.

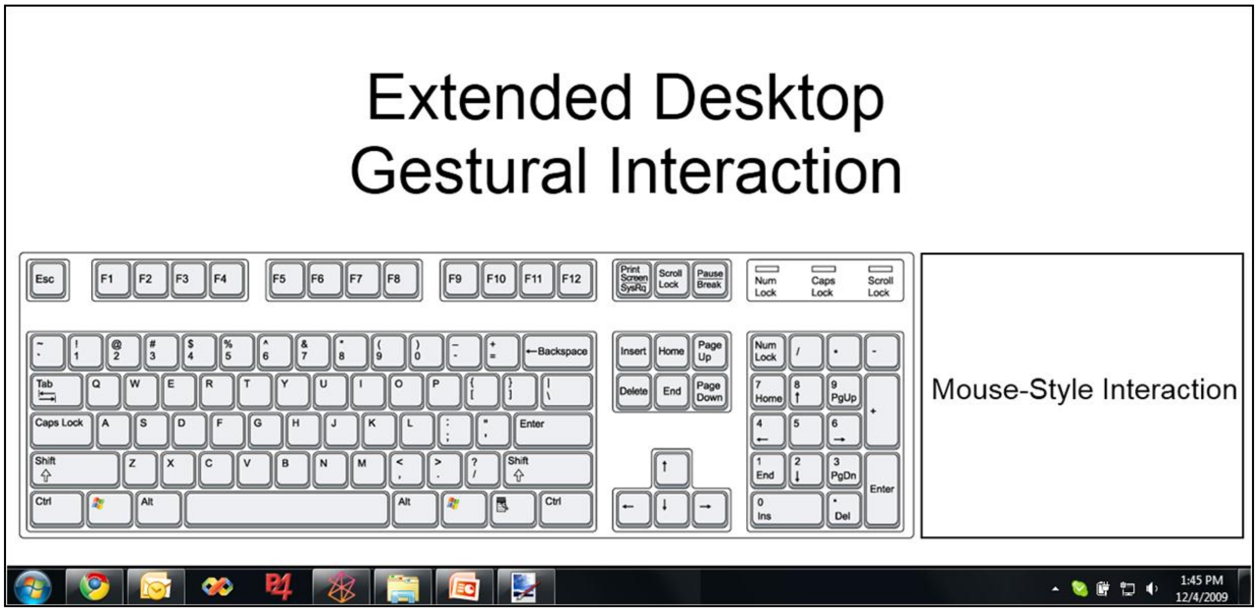


Figure 26: Proximal multi-touch device design concept

Additionally, the PMTD extends the computer’s virtual desktop to allow interaction between the primary and secondary displays as a seamless desktop. Virtual objects can be transferred between the displays through dragging and “flick” gestures. With an object opened on the proximal display, the user’s focus can shift away from the distal display and allow for direct interaction to the virtual object through fingers or a stylus. This interaction mechanism has commercial roots established by the Tablet PC

[91] and the Wacom Cintiq [9] and is particularly useful for artists who work with a stylus for creating and editing digital images. The entire PMTD acts as a region for gestural interaction. These gestures might include short-cuts for launching applications, sending email, organizing active windows, and opening/closing the mouse or keyboard regions. However, when the keyboard and mouse regions are “active”, these regions take precedence to other multi-touch input. Thus, when the user begins a gesture he/she should do so in an unoccupied region of the display.

At the very bottom of the PMTD concept image is an area depicting the Microsoft Windows application taskbar. This is to indicate that the PMTD is indeed an extended desktop display (a second monitor) and could be configured to house the taskbar. With the taskbar on the PMTD, the user can switch between applications by directly touching that area of the screen – without relying on the mouse region.

7.1.2 Rationale

When formulating this design, we sought to work within the existing desktop paradigm. Specifically, we chose an evolutionary path for the desktop platform instead of a wholesale departure from what has been the successful interaction design for decades. At the core of our design was the precept that a general-purpose computing platform must foster long-term-single-session use and accommodate efficient text input.

We felt that this platform must enhance the user's experience – not simply replicate the mouse+keyboard input devices in an integrated platform. Multi-touch excels at pen-style interaction and gestural commands. Thus, our design seeks to incorporate the successful and established interaction paradigms for text input, pointing, pen-computing, and gesturing into a single platform that robustly integrates with a modern operating system. Moreover, extending the virtual desktop allows for direct interaction with virtual objects upon the proximal display and for transferring objects between the output devices. If the distal display is single- or multi-touch enabled, object transferring can be accomplished through drag or flick gestures instigated on either display. Furthermore, the notion that all input is software-driven allows for extensive user customization – a feature generally missing from physical input devices.

7.2 Tactile Feedback

Throughout our investigations of multi-touch text entry, a glaring deficiency has been the lack of tactile feedback. We have oft speculated that the performance of text entry would significantly improve if the multi-touch platform "felt" more like a traditional keyboard. Thus we considered what aspects of tactile feedback were missing from a multi-touch platform and hypothesize that the two most important elements are: pressure sensitivity and the physical boundaries presented by traditional keys.

7.2.1 Pressure Sensitivity

Pressure sensitivity refers to the behavior of a traditional keyboard, whereby the user must exert a small amount of force to activate a key. The 3M M2256PW [3] multi-touch monitor, used in the experiments presented in Chapters 3 through 6, detects surface contacts with no force required – a common trait for capacitive-touch platforms. With such systems, the user cannot rest his/her fingers on the virtual keyboard without triggering inadvertent surface interaction. Software, such as the *Key Up* interaction technique we developed and discussed in Chapter 4, might mitigate unwanted interactions, but a bona-fide pressure sensitive device would more aptly address this issue. As discussed in *Chapter 2: State of the Art*, Interpolated Force-Sensitive Resistance (IFSR) platforms offer exciting potential for detecting pressure through multi-touch platforms. Unfortunately, these devices have not transitioned from the lab to commercial availability, as of this writing.

Vision-based multi-touch technology can attempt to detect pressure, and we ourselves have investigated this through our experiments with multi-touch form factors (described in Appendix A.3). Some vision technologies, such as Frustrated Total Internal Reflection (FTIR) operate by detecting light reflected from an emitter through user interaction. With FTIR, infrared light trapped within a waveguide escapes when the user touches the surface. The more surface area under contact, the more light can escape and be detected by the camera. Surface area under contact is then, to some extent, a function of pressure. The harder a finger is pressed onto the multi-touch surface, the larger the apparent “blob” detected by the camera. We could interpret the size of the blob as pressure – assuming that the blob is representing a finger and not an alternate input device (e.g. stylus). Calibration could be performed to determine the size of the user’s finger, and the associated size of blobs, under various pressures.

7.2.2 Physical Features

Perhaps even more important than pressure sensitivity, are the physical aspects of the physical keyboard. In particular, we call attention to four elements: key shape/size, the physical separation of the keys, the boundaries of the keyboard, and the rebound of a pressed key.

Keys on the keyboard have a particular shape. Most of the keys share a common shape and size – the character and number keys, for example. However, several keys have different shapes and sizes which help the user orient themselves on the keyboard without needing to look. The space bar, for instance, has a completely unique and unmistakable shape. Such tactile cues are absent from a multi-touch device.

Keys have spacing which provides tactile cues when moving from one key to the next. Moreover, some keys have nubs to identify a particular key – commonly the ‘J’ key or the ‘5’ key on a number pad. Furthermore, keyboards have traditionally organized their keys into rows, and these rows likewise help orient the user. The top row of keys, for example, is commonly reserved for numbers; and above the number row are the function keys. Certainly, the specifics of each row are particular to a brand of keyboard. But the notions of consistent space and key rows are common across almost all commercial keyboards.

The third physical element, we call attention to, are the boundaries of the keyboard. This refers to the edges of the outermost keys and the physical housing supporting the keyboard. The outermost keys form a tactile boundary that can assist the user in orienting themselves by touch. Consider the left-most set of keys on a typical QWERTY keyboard. From top-to-bottom, these keys are: Escape, Tilde, Tab, Caps Lock, Left-Shift, and Left-Ctrl. If the user is aware of this arrangement, he/she need rarely glance down at the keyboard to verify his/her position. Moreover, the boundaries provide the unmistakable area within which text entry is conducted. While, a multi-touch platform certainly has a containing frame, the inner contents cannot be determined through touch alone; and no such “text entry boundary” is commonly employed on a multi-touch platform. Previous work [73] has explored a transparent rubber material that can be fashioned into a keyboard shape, and which transmits a surface contact on a touch screen. However, we believe that such semi-permanent peripherals are cumbersome, requiring application and removal. Moreover, they violate our tenet that the multi-touch platform not be augmented by external peripherals.

The final physical keyboard aspect, that we draw attention to, is the rebound of a pressed key. To date, most multi-touch platforms employ a rigid surface which does not provide the notion of a key being depressed and rebounding on contact.

7.2.3 Summary

This section is intended to call attention to some of the physical differences between multi-touch platforms and traditional keyboards. While not the focus of our research, we feel these are open topics that could significantly impact the performance of text entry on a multi-touch platform. We encourage electrical engineers and likeminded researchers to provide such tools to those of us focused on the application of their technology.

7.3 Audible Feedback

The studies presented in this work, have focused on the task of quantifying the performance of text entry on a multi-touch platform. This is a natural first step, as it establishes a baseline from which we can compare future efforts in improving this performance – assuming the investigations identified deficiencies as compared with text entry on a traditional keyboard. However, it has always been our suspicion that the multi-touch platform would underperform with respect to a physical keyboard; our true question has been: “by how much?” Furthermore, our ultimate goal is the determination of the viability of multi-touch as a general-purpose computing platform. Thus, we have continuously had our eye on improving multi-touch text entry performance.

Indeed, as discussed in the previous section (*7.2: Tactile Feedback*), we have hypothesized a number of hardware-related factors that we believe could significantly impact multi-touch text entry performance. Unfortunately, our collective skillset does not include electrical engineering. Thus, we have been unable to directly implement and evaluate the suggestions offered in section 7.2. However, we have made attempts at improving multi-touch text entry performance, purely through software. Our first endeavor was the *Key Up* technique described and studied in Chapter 4.

In this section, we describe another approach – using audible feedback – one which we implemented and informally studied, and then discarded. Nevertheless, we present this work and encourage future researchers to evaluate this approach themselves.

Audible feedback refers to the sounds produced as a result of text entry on the multi-touch platform. Our virtual keyboard has integrated audible feedback from the outset – playing a recording of a physical key press when a virtual key is activated. Indeed, we received a considerable amount of positive response, from our study participants, with respect to this aspect of the virtual keyboard. Thus, we pursued two alternate approaches to audible feedback in an effort to improve multi-touch text entry performance: *voice* and *bin-based audio*.

7.3.1 Voice Feedback

Voice Feedback refers to playing a recording of a spoken letter, when a virtual key is pressed, rather than the original “tack-tack” sound of a physical keyboard.

Specifically, we implemented voice feedback through two sets of sounds files: *organic* and *synthesized*.

The first set, *organic*, is that of an English-speaking woman saying each letter of the English alphabet. The set contains one audio file for each letter of the alphabet, and the virtual keyboard software played the file associated with the pressed key. These audio files have an average length of approximately 400 milliseconds. The second set, *synthesized*, employs the same structure, but was generated through the Microsoft Text-To-Speech synthesizer. These synthesized sound bites, each speak a letter in approximately 200 milliseconds. We produced the two sets of audio files, with the suspicion that shorter files would not "overlap" one another as dramatically as longer files. However, we recognized that letters spoken in short duration were more likely to sound hurried or *clipped*.

To (informally) test the audio sets, we reused the software developed for the first two text entry experiments. In particular, the user typed the phrase: "the quick brown fox"; first in a slow and deliberate fashion, and next in a continuous typing mode. As the user typed, the audio system "spoke" the associated letter – first using one audio set and then the other. Non-character keys, such as the *Space* or *Backspace*, produced the original "tack-tack" sound recorded from a physical keyboard.

Videos of these tests can be found online at: http://www.bespokesoftware.org/UCF/Dissertation/ in the files Sonification1.mp4 (organic audio set) and Sonification2.mp4 (synthesized audio set). In both sets, the voice feedback was clear, as long as the user typed slowly. Under the user’s normal typing speed, the spoken letters ran together and produced a cacophony of sound. We found this to be the case for both audio sets. Indeed, we strongly felt that, not only did the voice feedback provide no improvement to text entry performance, but rather, it added a distraction that could reduce performance. Thus, after producing these videos, we decided to abandon the notion of voice feedback. Certainly, we could have developed a formal experiment to analyze the performance of this technique. However, we found the anecdotal evidence to be so strong, that bona-fide experimentation was unnecessary.

Another approach to voice feedback would be to reject the notion of playing each letter individually – and instead, announce a complete word and/or an “error” sound if the system did not find the word in a dictionary. Commercial spell checking systems exist that take exactly this approach. While we did not explore any such system, in conjunction with multi-touch text entry, it could be an interesting examination.

7.3.2 Bin-based Audio

The second audible feedback approach we considered was that of *bin-based audio*, based on work by Lumsden and Gammell in 2004 [40]. Their technique, used “a combination of stereo panning and pitch to represent stylus position” within the writing pad of a mobile text entry system. [40]. Essentially, they organized the writing pad into a grid of rows and columns, wherein the cells (“bins”) were associated with specific tones. As the user transcribed a gesture on the writing pad, sounds were produced; and each gesture was represented by a distinct audio signature. We considered applying a similar design to multi-touch text entry, whereby the virtual keyboard would be organized into a grid with associated sounds for each cell.

Ultimately, we rejected this approach and did not even implement a pilot system as we did with voice feedback. While we think this is an interesting idea, our trouble with this system is twofold: text entry speed and vocabulary. While bin-based audio may work well for handwritten gestures, we are concerned with the speed differential between handwriting (15 wpm [46]) and typing on a physical keyboard (56 wpm [57]). At almost 3-times slower, a user has much more time to process audio cues associated with handwritten text, than he/she does through keyboard text entry. At expert typing speeds, we are concerned that such a system would produce the same cacophony of sounds that we found with voice feedback.

Additionally, the audio design presented by Lunsden and Gammell is concerned with associating audio with the 26 letters of the English alphabet. Thus, should our design follow suit, and provide bin-based representations of the 101 keys on the keyboard? Or should our system attempt to provide distinct audio representations of the tens of thousands of words in the English vocabulary? We find it unlikely that the typical user will recall 101 distinct sounds and their corresponding character, let alone an extremely large vocabulary of words. Moreover, we are uncertain if we could produce 101 audio signatures that are distinguishable as different by a typical user.

In conclusion, we were initially enthusiastic about the potential for bin-based audio, and concede that more time could be spent investigating this approach. Indeed, future work could at least attempt a pilot implementation, not dissimilar to the work we performed on voice feedback. To be sure, such an investigation may produce results that contradict our expectations. However, as we identify research directions, we recognize that the body of work far outweighs available time. Thus, we must be particular as to how we proceed, and perhaps lean towards techniques that we believe have a greater potential for success.

7.4 Summary

In this chapter, we have examined a number of additional investigations in the area of general-purpose multi-touch computing. We discussed our vision for a general-purpose multi-touch platform, its design and rationale. We presented our considerations of tactile feedback for multi-touch. And, finally, we discussed our investigations of audible feedback. While not the focus of the research presented in the previous chapters, these investigations supplement the principal body of work.

CHAPTER EIGHT: CONCLUSION

8.1 Summary of Research

In Chapter 1, we introduced our work in general-purpose multi-touch computing. We defined multi-touch as: *a classification of computer input device that detects multiple simultaneous points of contact on a two dimensional surface.* We defined general-purpose computing as: *the “every-day” work that is accomplished, through the assistance of a desktop computer, in a typical office environment.* And we proposed that, in order to support general-purpose computing, a platform must: 1) facilitate long-term-single-session use; and 2) accommodate efficient text input.

We observed that: *multi-touch platforms have, thus far, been special-purpose devices;* and this observation led us to ask “can multi-touch, without a text entry peripheral, provide a platform for efficient text entry? And, by extension, is such a platform viable for general-purpose computing?” With these questions in hand, we introduced our work through two research thrusts:

1. Examination of existing text entry methods

2. Empirical evaluation of text entry and word processing methods

For our initial thrust, we presented a tailored set of evaluative criteria for examining existing text entry methods, and applied them to identify a category of potentially viable input systems – on-screen soft keyboards.

In Chapter 2, we presented a state-of-the-art review of multi-touch, including a discussion of the origin of multi-touch, hardware, software, open research topics, and resources available for researchers interested in the field.

In Chapter 3, we detailed our first empirical examination of plain-text entry on a multi-touch platform – a study which established baseline performance characteristics of multi-touch text entry through an on-screen implementation of the QWERTY character layout. We intuitively expected the physical keyboard to outperform the multi-touch platform – and indeed the results of the experiment confirmed our intuition – but, our true question was always “by how much?” This study provided those statistics. Furthermore, the post-study questionnaire responses offered support for the usefulness of the multi-touch platform as a text entry device. 38% of the participants stated that they would use the multi-touch platform to enter text for everyday computer activities – only 2% lower than those who explicitly said they would not.

We continued our examination of text entry in Chapter 4, where we discussed our second user study. This study measured the impact of our *Key Up* technique, a multi-touch text entry method intended to mitigate errors produced when users “dragged” their fingers across the multi-touch virtual keyboard, to get to the next key. We suspected that this was a cause of mistakes made in entering text on the multi-touch platform. Specifically, the 3M M2256PW [3] multi-touch monitor, used in each of our studies, requires no pressure to trigger the detection of a surface contact. This is in contrast to a typical physical keyboard, which allows users to rest their fingers on the keys, or slide their fingers from key to key, without causing inadvertent text entry. Thus, we suspected that the participants of our study, all expert typists on a physical keyboard, retained their “sliding/dragging” behavior when typing on the multi-touch keyboard and this caused unintended key presses. Surprisingly, while the *Key Up* technique didn’t significantly reduce performance, it didn’t improve it either. Likewise, we found no statistical significance within user preference between the original and newly introduced multi-touch text entry techniques.

In the next chapter, we added mouse-style interaction into the mix, in the context of a word-processing task. Specifically, we sought to understand if the inclusion of the new interaction style – one that the multi-touch platform presumably excelled at – would impact users’ potential adoption of the platform. However, before we could examine performance and adoption-potential of the multi-touch platform, in the context of a word-processing task, we first had to investigate how best to implement word-processing on the system. Chapter 5 detailed our analysis of two approaches: *Direct* and *Indirect Interaction*. *Direct Interaction* indicates that the user touches directly onto the portion of the screen he/she wishes to interact with. For example, if the user wishes to press a WIMP-style button, he/she would touch the multi-touch screen within the visual bounds of the button. *Indirect Interaction* denotes an emulated mouse on the multi-touch platform with the aid of a virtual mouse pad. With this technique, the user touches an area of the screen resembling a laptop trackpad. Under *Indirect Interaction* the mouse cursor is moved as the user drags his/her finger within the bounds of the virtual mouse pad, and mouse-button interaction is performed through GUI-style buttons or by tapping on the mouse pad itself.

The results of the study found unanimous preference for the *Direct Interaction* technique – offering clear guidance for the word processing platform study. Furthermore, this research introduced newly developed metrics for tracking formatting errors within a word-processing task and expanded the set of performance metrics, including the notion of *Completion Time*.

In Chapter 6, we continued our investigation of word processing through an analysis of performance and user preference between multi-touch, desktop, and laptop computing platforms. This experiment was similar to our very first text entry experiment, in that our goal was to establish baseline performance measurements from which we could explore improvements in multi-touch word processing. The results of our experiment were unsurprising – the traditional desktop and laptop platforms outperformed the multi-touch system; but again, as with the first text entry study, this experiment was primarily concerned with the question “by how much?” And, furthermore, “how does the inclusion of mouse-style interaction affect someone’s acceptance of the multi-touch platform as a general-purpose computing device?” Interestingly, our results indicated that fewer people would adopt the multi-touch platform for word processing, as compared to plain-text entry. However, we suspect that these opinions might have been skewed by the differences in punctuation between the two experiments.

Finally, in Chapter 7, we detailed a number of additional topics we investigated in the area of general-purpose multi-touch computing. We presented a vision of a general-purpose multi-touch platform, its design and rationale. This platform exhibits a “dual-screen” system composed of a proximal multi-touch device and a distal display (either interactive or non-interactive). The design is fashioned after the conventional mouse+keyboard+monitor platform where the posture of the user is the same as with a traditional computing platform – the user’s hands resting before him/her with his/her visual focus orthogonal to the input device. The implementation and evaluation of such a platform is left to future efforts.

We also discussed our opinions concerning tactile feedback for multi-touch platforms, specifically identifying *pressure sensitivity* and *physical boundaries* as the two most important tactile elements missing from modern multi-touch platforms. While we briefly discussed our own investigations into detecting pressure on a vision-based platform, we left the implementation of such systems to future work.

Finally, we discussed a brief exploration into audible feedback as a potential mechanism for improving text entry performance on a multi-touch platform. Our virtual keyboard has integrated audible feedback from the outset – playing a recording of a physical key press when a virtual key is activated. Indeed, we received a considerable amount of positive responses, from our study participants, with respect to this aspect of the virtual keyboard. Thus, we pursued two alternate approaches to audible feedback in an effort to improve multi-touch text entry performance: *voice* and *bin-based audio*.

The remainder of this thesis offers conclusions to our research and discusses future work. Additionally, Appendix A*,* describes a body of work that we have contributed to the multi-touch field over the last several years. This research provided a foundation from which to explore multi-touch for general-purpose computing.

8.2 Conclusions

The studies presented in the previous chapters, have focused on the task of quantifying the performance of text entry on a multi-touch platform. This is a natural first step in an examination of multi-touch for general-purpose computing. From the results of these studies, we conclude that traditional desktop computers outperform our selected multi-touch platform. We anticipated this outcome, but the studies were necessary in order to establish a baseline from which we can compare future efforts toward improving this performance. Indeed, these results provide a clear performance “upper-bound” and underscore the performance gaps between platforms across text entry and word processing tasks. However, our results suggest, at least for some of our participants, that the performance was “good enough” that they would consider adopting the multi-touch platform for these everyday tasks.

Improving performance, through purely software techniques, proved elusive. Our *Key Up* technique had no significant impact on plain-text entry performance. Nor were our efforts in audible feedback successful. This is not to suggest that no performance improvement is possible, through software-only modifications, but we conclude that the attempts described in this thesis were unsuccessful. Furthermore, we propose that software-only modifications are perhaps less important to improving performance than hardware modifications to the multi-touch platform. In particular, we identified two elements, for tactile feedback, that we believe have strong potential for improving text entry performance: *pressure sensitivity* and *physical features*.

We also developed new metrics for measuring formatting errors made during word processing tasks – and conclude that such measurement is not trivial. In particular, we detailed the problem measuring formatting errors and offered guidance to future researchers on collecting the necessary data. Moreover, we suggested a number of measurement techniques, including: *character-by-character*, *consecutive-character*, *whole-word*, or *whole-consecutive word* analysis. We also presented our algorithm for measuring bold and italics errors using the *character-by-character* method.

Finally, we offered a vision for a general-purpose multi-touch platform and conclude, at least, that the pursuit of such a platform is a worthwhile endeavor.

8.3 Future Work

While extensive, the research described in this thesis is but a first step in answering the questions: “can multi-touch, without a text entry peripheral, provide a platform for efficient text entry? And, by extension, is such a platform viable for general-purpose computing?” We concluded that, for some, the answer to these questions is certainly “yes”. However, those disinclined to adopt the platform outnumbered those who would use the multi-touch system for everyday text entry and word processing tasks. Thus, further efforts are required to determine if the platform is viable for the majority of users.

First, our existing text entry and word processing studies could be expanded to provide additional results. In particular, we suggest longitudinal studies to determine the peak performance of text entry and word processing on the multi-touch platform along with the corresponding learning curves. Furthermore, we are uncertain if our results will generalize between different multi-touch technologies. We suspect that devices within a given technology, capacitive multi-touch, for example, will yield similar results under the same studies. But we are not clear that these results will be consistent between technologies or form factors. Thus we recommend a similar battery of tests against various multi-touch devices.

Also, we concede that the comparison of adoption-potential between the first and fourth studies is potentially flawed. Recall that the results of the “would use” question of the fourth study revealed that fewer people would adopt the multi-touch platform for word processing, as compared to plain-text entry. However, we suspect that these results might have been skewed by the differences in punctuation between the phrases/paragraphs presented within the two experiments. To clarify these results an additional study could be run utilizing the word processing paragraphs, in plain-text, within the text entry experiment, or vice versa.

We also propose future study of audible and tactile feedback. It is our opinion, that tactile feedback offers the greatest potential for improving text entry performance on a multi-touch platform. Indeed, other researchers have been working on the question of the physical flat-screen multi-touch platform and its affordance for input. As we mentioned in Section 2.2.2.9, we consider Interpolating Force-Sensitive Resistance [64] a significant innovation for multi-touch, particularly with respect to general-purpose computing, because of its support for pressure detection. Furthermore, a new effort by Findlater, Wobbrock, and Wigdor [17] (pending publication) examines typing patterns on a flat surface, with the goal of informing future designs of touch screen keyboards.

Finally, we look to the development of a multi-touch platform whose *purpose* is general-purpose computing – perhaps a platform similar to the design we proposed in section 7.1. In fact, new work by Xi, et al. [7] (pending publication) implements and evaluates a design very similar to what we have proposed.

If we were to consider a “blue-sky” multi-touch platform (one perhaps not yet practical or possible to develop) it would include tactile feedback in a fashion whereby a semi-permanent physical keyboard would be embedded into a display and could be invoked or dismissed upon command. This keyboard would physically emerge from the display upon activation and seamlessly dissolve back into the display upon dismissal. Moreover, the physical attributes of the display – its key arrangement, sensitivity, shape, and layout – would be customizable through software. Such a system would still support pen-based computing and gestural interaction, but could provide the text entry performance of a desktop keyboard.

Multi-touch has already proven itself as a commercially successful computing platform – excelling at natural user interaction for certain tasks. Our question is will the general public adopt these systems as their chief interaction system? Can multi-touch provide such a compelling platform, that it displaces the desktop mouse and keyboard? We have asserted that general-purpose depends upon efficient text entry. And we have quantified the current text entry performance of a multi-touch platform. Even if this is the highest text entry performance we can achieve on the platform, do the other features offered by multi-touch compensate for the reduced text entry performance? Features such as gestural interaction, tangible interfaces, pen-based computing, and complete customization – are these features enough? Is multi-touch the next revolution in human-computer interaction?

APPENDIX A: SUPPORTING RESEARCH

In addition to the research presented in the previous chapters, we have produced a significant body of work in the field of multi-touch interaction over the last several years. These efforts have provided a foundation from which to explore multi-touch for general-purpose computing. This appendix describes some of the more noteworthy of these efforts.

A.1 The Bespoke Multi-Touch Research Testbed

A.1.1 Introduction

The pending proliferation of multi-touch technology, which allows interaction with a surface through multiple simultaneous points of contact and from multiple concurrent users, has the potential to radically change Human-Computer Interaction. However, unless the research community can answer fundamental questions about optimizing interface designs, and provide empirically driven guidelines, the technology can become just another I/O modality looking for a purpose. Unfortunately, there has been limited availability of hardware and software to support this research. Thus, investigators must overcome a number of technical challenges to develop a multi-touch platform before they can begin research in this area.

Through an extensive literature and state-of-the-art review, an analysis of the requirements for a multi-touch research platform revealed two categories of components: those essential for basic multi-touch research; and secondary components necessary for extensive investigation and longer-term research projects. These components, listed in Table 25, emphasize a low-cost, do-it-yourself approach.

Table 25 Essential and secondary components for a multi-touch research platform

Essential Components	Description
Multi-touch surface	Constructed using commercial-off-the-shelf hardware and requiring near zero pressure to detect an interaction point
Software Hit-Testing	Ability to determine the presence and location of each point of surface contact; supporting at least four users
Software Point Tracking	Identifying a continuous point of contact and reporting its velocity and duration

Secondary Components	Description
Application Programming Interface (API)	A software system upon which multiple multi-touch applications can be developed
Multi-Platform Support	The ability to access multi-touch interaction data from different computing environments (e.g. languages, OS's, etc.)
Reconfiguration	Modifying the software system without recompilation
Software Service	Allowing multiple applications to access multi-touch interaction data simultaneously, including over a computer network
Presentation-Layer Independence	Isolating the multi-touch interaction data from the system to graphically present such data, allowing any GUI to be employed when
Mouse Emulation	Support for controlling traditional 'Window, Icon, Menu, Pointing Device' interaction through a multi-touch surface
Tangible Interfaces	The ability to detect and interact with physical devices placed on or near the multi-touch surface
Customizable Gesture System	Support for training arbitrary multi-touch gestures and mapping them to software events

A multi-touch platform is made up of two primary components: a physical interaction surface, and a software system for collecting and interpreting points of contact. The hardware and software systems each require significant investments of time and effort to construct. While advances in hardware and multi-touch software provide interesting research opportunities themselves, they are a barrier to entry for researchers who want to focus on higher-level interface issues or the development of novel applications.

This section presents the Bespoke Multi-Touch Research Testbed, a hardware and software system for enabling research in multi-touch interaction. A detailed discussion is provided on hardware construction, pitfalls, design options, and software architecture to bridge the gaps in the existing literature and inform the researcher on the practical requirements of a multi-touch research testbed. This includes a comprehensive description of the vision-based image processing pipeline developed for the Bespoke software library, which makes surface interactions available to multi-touch applications. Furthermore, this section explores the higher-level functionality and utility of the Bespoke software library and how researchers can leverage the system to investigate multi-touch interaction techniques.

A.1.2 Hardware

Although there are a variety of multi-touch technologies, we believe that Frustrated Total Internal Reflection (FTIR) offers a robust and affordable hardware solution. FTIR surfaces share a set of common components: 1) an optical waveguide for conducting infrared (IR) light; 2) a supporting structure for holding the waveguide; 3) an IR sensing camera; 4) a projector and diffuser; 5) an IR emission source; and 6) a computer. The hardware design for our testbed is pictured in Figure 27. For the optical waveguide, we have chosen a 32”x24”, ½” thick sheet of clear acrylic. The dimensions of the surface match the 4:3 aspect ratio of most modern, short-throw projectors. The supporting structure for the acrylic is a 37” high table and includes a 6” wide border around the waveguide, for placing materials (or resting elbows) that will not interact with the surface. The IR camera and projector are placed below the acrylic, and are contained within the table. This design supports collaboration with up to four seated or standing users. The IR camera is a Microsoft LifeCam VX-6000, a commercial-off-the-shelf webcam that has been modified to allow IR light while filtering visible light. The projector is a Mitsubishi XD500U-ST short-throw projector, capable of producing a 60” diagonal from only 33” away. The diffuser is a sheet of Rosco Gray 7mm thick PVC, rear-projection material. For IR emission, we chose a set of 32 Osram 485 LEDs. These are split into 4 chains (connected in parallel) of 8 LEDs (connected in serial) running to a common 12V power supply. Lastly, we have chosen a small-footprint MicroATX computer for operating the multi-touch surface software. Several of these components are discussed in more detail below.

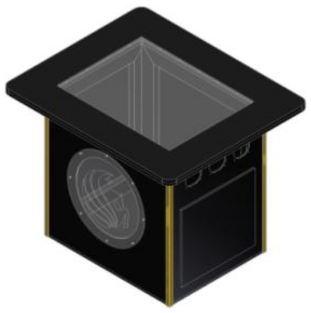


Figure 27: Bespoke multi-touch research testbed hardware design

A.1.2.1 Acrylic Waveguide & Infrared LEDs

Acrylic has optical properties conducive to Total Internal Reflection [24]. It is also quite durable, inexpensive, and can be manufactured in a number of sizes. Thickness is a consideration at large dimension, as pressure on the surface causes the acrylic to noticeably deflect. At 32”x24” we found a ½” thick sheet of acrylic to be a nice compromise between rigidity and cost.

When placing the LEDs around the acrylic, the primary concerns are: 1) providing enough light to fill the waveguide, and 2) fully distributing that light within the waveguide. We initially drilled 5mm wide depressions, into the acrylic edges, to house the LEDs. This works quite well, and does not require polishing the acrylic edge to introduce light into the waveguide. A drawback to this approach is that the LEDs are semi-permanently affixed to the acrylic. Working with the waveguide (for example, to pour on a silicone rubber compliant surface) often requires removing the LEDs. In our final design, we chose to surround the acrylic with LEDs, equally spaced, and abutting the acrylic edges. Again, we found that polishing the acrylic edges was not required. The choice of 8 LEDs per side is more than sufficient for our surface size, given their 40-degree viewing angle. In fact, we found quite reasonable results with only two adjacent edges lit. To determine if enough light is being introduced into the waveguide, one can point the IR camera at an edge opposite to the illumination. The camera should detect a solid bar of IR – the light escaping the edge. If this bar of light is segmented, or significantly varies in intensity, then the illumination is not being fully distributed throughout the waveguide and additional LEDs are required.

A.1.2.2 Projector & Diffuser

Short throw projectors, capable of displaying large images from very close distances, have become increasingly available in recent years. A short throw is necessary for maintaining small table depth. If the multi-touch system has no depth requirement (e.g. a wall-display) then a traditional projector can help reduce cost. Strictly speaking, a traditional projector, with a system of mirrors for increasing focal length, can be used in a depth-limited multi-touch surface. However, this adds complexity to the hardware design.

The diffuser is the surface upon which the projected image will be displayed. Aside from the choice of materials, a chief concern for the diffuser is its placement – either above or below the waveguide. Placing the diffuser below the waveguide causes a slight disparity (of the thickness of the waveguide) between the interaction surface and the projected display. Furthermore, a pliable diffuser material, placed below the waveguide, will sag, deforming the projected image if not otherwise supported. Moreover, the diffuser may absorb some of the IR light passing through it. While this is a benefit for reducing ambient IR light, the diffuser will also absorb some of the light being reflected through FTIR when placed below the waveguide. For these reasons, we suggest placing the diffuser above the waveguide. This approach also protects the waveguide from scratches and oils from the users' fingers. Unfortunately, placing the diffuser above the waveguide negatively impacts the coupling between the users' fingers and the waveguide, decreasing the light reflected through FTIR. A material, placed between the diffuser and the waveguide, is required to improve coupling and thereby increase the amount of IR light directed to the camera while decreasing the force necessary to reflect it.

A.1.2.3 Compliant Surface

A finger makes an imperfect connection with acrylic. Micro air-gaps form between the user's fingers and the waveguide and maintain the original acrylic-to-air interface, thus supporting Total Internal Reflection. Moistening fingertips or pressing firmly on the surface can improve coupling, but these are not viable long-term solutions. A coupling material is required to permit low-force, high-quality surface interaction.

After much trial-and-error, we settled on a 1mm thick layer of SORTA-Clear 40 – a translucent silicone rubber – placed between the acrylic and the diffuser. SORTA-Clear 40 is a liquid that, when mixed with its catalyst, will cure at room temperature into a firm (40A Shore hardness) material that provides very good coupling with limited hysteresis. However, mixing the rubber with its catalyst creates air bubbles, which will cause considerable noise in the captured image. Placing the mixed rubber into a vacuum chamber can help remove these bubbles, but there is limited time before the material begins to cure, and the pouring and smoothing process will reintroduce some air. Placing the entire surface into a vacuum, after the material has been poured, may be the best option for removing bubbles – if a large enough vacuum chamber is available. We found good results, in mitigating air bubbles, simply by keeping the thickness of the rubber very small (e.g. <=1mm) and by pouring and smoothing slowly and deliberately. Applying a small amount of heat, from a hair dryer or heat gun, can help remove any stubborn bubbles before the rubber cures.

While pre-cured layers of rubber are available, we found them difficult to adhere to the acrylic without introducing a large number of air pockets. Pouring the silicone rubber directly onto the surface produced the best results.

A.1.3 Software

The chief function of the Bespoke software library is to collect and interpret multi-touch surface input. The core components of the library do not specify how this data is used or displayed. Thus, the library is presentation-layer independent and graphical user interface (GUI) systems such as Windows Presentation Foundation (WPF), Windows Forms (WinForms), and Microsoft XNA can all be used to develop front-end applications that utilize the multi-touch framework. To support various presentation systems, the Bespoke software framework maintains two modes of data communication: polling and events. Traditional WinForms applications use events to communication object information; whereas polling is more common for simulations or video games, where input devices are continuously queried.

A.1.3.1 Image Processing

At the core of the software, is an image processing system that converts raw camera data into points of interaction. The image processing system runs in its own software thread, and captured frames are sent through the processing pipeline depicted in Figure 28.

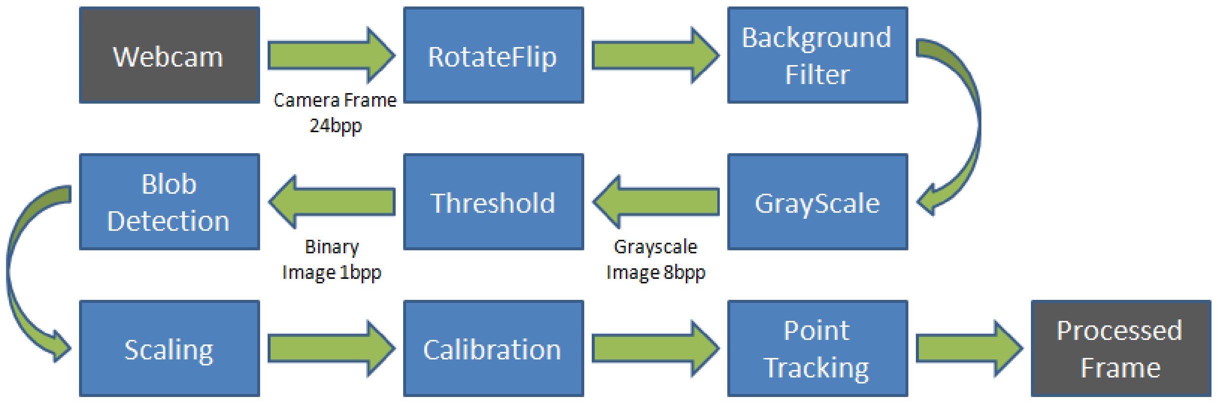


Figure 28: Image processing pipeline

Processing begins by capturing an image from a video source – a Microsoft DirectShow compatible device. The camera's device enumeration, frame rate, and resolution are specified through the framework's XML configuration file. The *RotateFlip* step transforms the image vertically and horizontally, as specified in the configuration file, orienting the image to match the projector.

During initialization, the software library captures a set of frames, while the surface is quiescent, and combines them to form a background image. The *Background Filter* step subtracts this background image from the active frame, thus removing noise from the image. Noise originates from ambient infrared light, “hot-spots” produced by the projector, oils and debris on the waveguide and compliant surface, and from light unintentionally escaping the waveguide. The Bespoke software library allows the user to recapture the background image at any time and does not force the restart of the image processing system to compensate for a dynamic lighting environment.

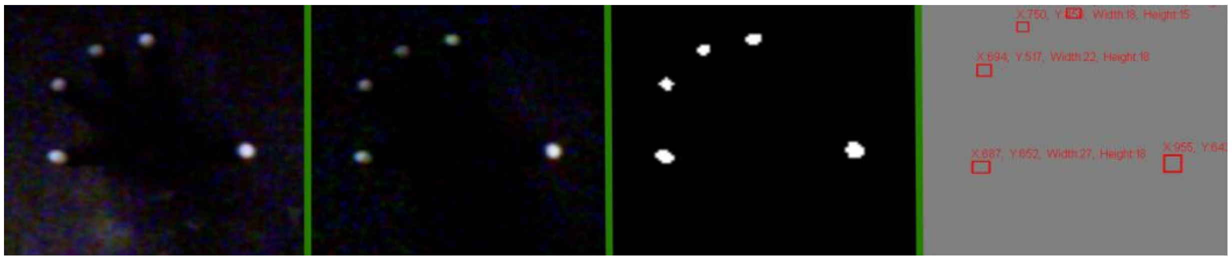
Most webcams capture images in color, typically at 24 bits per pixel (bpp). The *Grayscale* action converts a color frame to gray scale (8bpp). This step can be removed if the camera natively captures images in gray scale – the format required for the subsequent *Threshold* filter, which further isolates pixel values to black or white (fully *on* or fully *off*). Pixels below the configurable threshold value are treated as *off* and pixels above as *on*. The resulting 1bpp black & white image is sent to the *Blob Detection* process, which groups neighboring *on* pixels into *blobs*. Blobs are the regions of the image that we consider for potential points of surface interaction. The blob detector filters out blobs below a minimum width and height, as specified in the configuration file.

Scaling adjusts the dimensions of the image to correspond to the resolution of the image projected onto the multi-touch surface. The *Calibration* step then adjusts for differences between the interaction surface, projector, and the camera that create discrepancies between the points touched on the surface and the location of those points in the camera frame. By sampling points at the four corners of the surface, we can construct a transformation matrix and generate a lookup table with the corrected location for every point of interaction. This table can be serialized, and the resulting file specified in the XML configuration for automatic loading by the framework.

The final phase of the image processing pipeline is *Point Tracking*, which takes the detected, scaled, and calibrated blobs and abstracts them into *FtirPoint* objects. An FtirPoint object has attributes including: a globally unique identifier (GUID), location, timestamp, bounding box, speed, direction, and duration. Each FtirPoint represents a single point of interaction with the surface, and it is this data that is most useful for multi-touch applications. With the location and bounds of an FtirPoint we can perform hit testing – testing an area on the screen for an interaction point – through simple rectangle intersection. The point tracking process also labels each FtirPoint with a GUID that is maintained as long as the interaction point is present. To track a point across frames, we again perform rectangle intersection between the previous and current frame's FtirPoints. Points that intersect are considered the same point, and differences in location and time are used to calculate the point's speed, direction, and presence duration. Detected points that do not intersect with previous points are assigned a new GUID. This process allows points to split and merge, and enables gestural interaction with the surface. However, the performance of this technique is tied to the quality of the camera and the compliant surface.

The purpose of the compliant surface is to improve coupling between the user’s fingers and the waveguide. If that coupling is poor, the user’s fingers will “stutter” across the surface, and will not continuously reflect IR to the camera. The gaps between images would cause the point tracking system to re-label what would otherwise be the same point. The framework provides a stutter-correction system that tracks points within a time-window for label reuse. Tracked points that become absent from a camera frame are transferred to a “pending disposal” collection for a user-configurable time (250 millisecond default). Newly detected points are matched against this collection, again through rectangle intersection, before they are assigned a new GUID. In this fashion, the framework will reconstitute a point that becomes briefly disconnected from the surface. Stutter mitigation, however, does not address a camera with a slow frame rate. If the user’s fingers move across the surface faster than the camera can track, the software library will label the points as disconnected. Future work on the library will attempt to address this through point prediction.

Exiting the image processing pipeline is the set of currently detected FtirPoints. Figure 29 shows a camera frame as it is passed through the image processing pipeline. Specifically, the figure displays: the raw camera frame (a), the background filtered image (b), the threshold image (c), and the fully processed FtirPoints displayed on the multi-touch surface (d).


X:750, Y:, Width:18, Height:15
X:694, Y:517, Width:22, Height:18
X:687, Y:652, Width:27, Height:18
X:955, Y:64

(a) (b) (c) (d)

Figure 29: A camera frame passed through the image processing pipeline: raw camera frame (a), background filtered (b), threshold (c), processed points (d)

A.1.3.2 Framework Features

While the image processing system forms the heart of the Bespoke multi-touch software framework, there are a number of additional features that can aid in the creation of multi-touch applications including: multi-platform communication, 2D/3D graphics, pen/writing-style interaction, and gesture recognition.

The Bespoke multi-touch software library is built on two open-source libraries: the Bespoke Open Sound Control Library (OSC) [81] and the Bespoke 3DUI XNA

Framework [80]. OSC is an open, lightweight, message-based protocol that enables, for example, multi-touch data to be transmitted over a network. The Bespoke 3DUI XNA Framework is a software library for enabling research in game development and 3D user interaction (3DUI). The Bespoke multi-touch software framework employs this library as a presentation layer for multi-touch applications; allowing games and simulations to be constructed with multi-touch input.

Another interesting feature of the Bespoke multi-touch software system is its support of pen/writing-style interaction. Pen-style computing refers to human-computer interaction through the digital representation of ink or writing, typically input through a computer stylus [9]. Ordinarily, pen-computing is single-touch – where input is collected from only one location at a time. This is a degenerate case of multi-touch, where we constrain the input and treat interaction points as digital ink. The Bespoke multi-touch software library collects ink data into *Stroke* objects which can be used for 2D recognition. The Bespoke framework provides a machine-learning system for training and classifying stroke data based on work by Rubine [66]. A detailed discussion of this feature is provided in Section A.2.

A.1.4 Case Studies

The Bespoke software library has been utilized in the creation of many multi-touch applications, and across a variety of domains. This section discusses four projects built with the framework, pictured in Figure 30: SurfaceCommand (a), InkDemo (b), Waterfall (c), and the Bespoke mouse emulator (d).

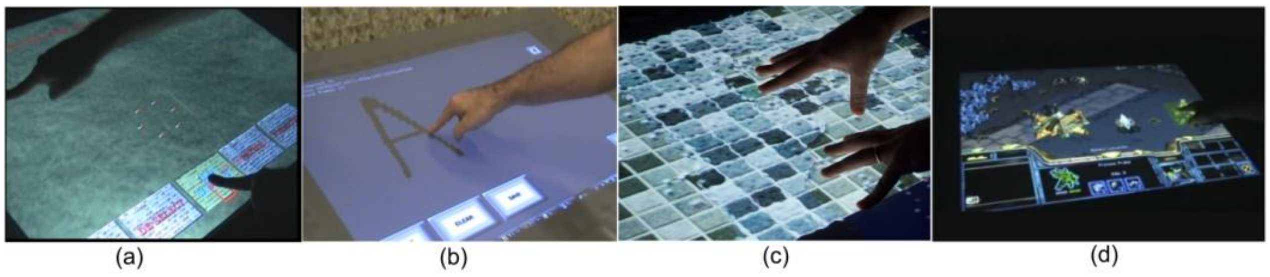


Figure 30: Multi-Touch applications: SurfaceCommand (a), InkDemo (b), Waterfall (c), and the Bespoke mouse emulator (d)

SurfaceCommand, pictured in Figure 30a, is a multi-touch demonstration styled after real-time strategy games. Built using the Bespoke 3DUI XNA Framework, with the Bespoke multi-touch extensions, SurfaceCommand presents a 3D battlefield viewed through an orthographic virtual camera. The user can pan around the battlefield by sliding two or more fingers across the display and zoom into and out of the map with “pinch” gestures – a motion whereby two interaction points are moving in roughly opposite direction, either toward or away from each other. Spaceships within the battlefield can be selected and moved about the map with single interaction points; and multiple ships can be selected and deselected with mode buttons along the bottom of the display. This simple application explores techniques that could be used within a real-time strategy video game and was developed in just four days.

InkDemo, pictured in Figure 30b, demonstrates the pen-style interaction of the Bespoke multi-touch framework. Stroke data is collected when the user provides only a single point of interaction. As the user slides a finger across the display, simple block-style lines are generated to visualize the underlying stroke data. A set of strokes can be labeled and committed to the Bespoke symbol recognition system. With a sufficient number of training samples, the user can then classify an unlabeled set of strokes.

Our third application, Waterfall, is an example of multi-platform communication and the rapid development capability of the Bespoke software system. The application is a fluid-dynamics demonstration, where simulated water flows down an inclined surface and can be perturbed by multi-touch surface interaction. Users interact with the water to form “dams” with their hands and fingers. The simulation was developed in C++ and rendered with the OGRE game development platform [76]. The multi-touch input was serialized via Open Sound Control, as described in Section A.1.3.2, and received by the simulation using an open-source C++ OSC implementation. The integration effort took only three days from start-to-finish.

The last example demonstrates the Bespoke mouse emulator – an application that allows the use of a multi-touch surface to control traditional, mouse-driven Windows applications. The mouse emulator associates a set of gestures with common mouse commands, as listed in Table 26.

Table 26 Mouse emulator gesture mappings

Function	**Gesture Description**
Left Click	Quick tap on the surface with one finger.
Left Click (alternate)	While holding down a finger, tap another finger to the left side of the first.
Drag	Perform a Left Click (alternate) but do not release the left side press. Drag both fingers to the destination and release.
Right Click	While holding down a finger, tap another finger to the right side of the first.
Double Click	Tap two fingers at the same time.
Mouse Wheel Scroll	While holding down a finger, drag another finger vertically and to the right side of the first. Dragging up scrolls the mouse wheel up and vice versa.
Alt-Tab	While not a mouse command, the Alt-Tab command is a useful Windows feature that switches between applications. To perform an Alt-Tab, hold down a finger and drag another finger horizontally above the first. Dragging to the left moves backward through the list of active applications and dragging to the right moves forward.

Figure 30d shows the emulator in use with the commercial video game *Starcraft* by Blizzard Entertainment. This popular real-time strategy game is typically controlled through the mouse and keyboard; but using the Bespoke mouse emulator, one can play

Starcraft through a multi-touch surface. Videos of these, and other Bespoke multi-touch demonstrations, can be found at http://www.bespokesoftware.org/.

A.1.5 Summary

In summary, while multi-touch technology has generated considerable excitement and offers the potential for powerful new interaction techniques, the researcher must overcome a significant obstacle for entry into this field – obtaining a multi-touch hardware and software research platform. Few commercial options exist, and there are deficiencies in academic literature on constructing such a platform. This section described the requirements of a multi-touch research system and presented the Bespoke Multi-Touch Research Testbed, a hardware and software testbed that enables research in multi-touch interaction. The testbed discussed offers insight into the construction of a robust, low-cost multi-touch surface and the development of an extensible software system for the rapid creation of multi-touch applications.

A.2 2D Symbol Recognition with Multi-Touch Surface Input

A.2.1 Introduction

Recognizing 2D symbols through machine learning has had a long history of academic research and commercial investment. We have based much of our machine learning approach on work by Rubine [66], who identified 13 features useful in classifying 2D symbols. Microsoft, Apple, Wacom, and myriad other hardware and software makers have made considerable investments in pen-based technology since the early 1990s. Early pen-based products include: the Apple Newton, the IBM ThinkPad, and the QBE Vivo Tablet PC. Since then, Microsoft Windows XP Tablet PC Edition (2002), Windows Vista (2007), and Windows 7 (2009) have included pen support directly into the operating system, establishing a common software platform for developing single-touch, pen-based user interfaces. When developing our multi-touch platform we recognized the benefit of including single-touch, pen-style interaction. This concept is a degenerate case of multi-touch, where single contact points can be combined into ink-style stroke information useful for training and classifying 2D symbols.

As discussed in the previous section, the Bespoke Multi-Touch Research Testbed provides software services to capture and interpret simultaneous contacts with a multi-touch surface. At the core of this process is an image processing pipeline which identifies and labels interaction points from frame-to-frame. When considering the problem of 2D symbol recognition, we can treat these points as digital ink data to reproduce the pen-and-paper analogy found in pen-based applications. A *Stroke* object is made up of a collection of interaction points and represents a single surface contact from press to release. The bounds of a stroke are defined by the extents of all of the contained points, and each stroke is assigned a globally unique identifier (GUID). An *InkSegment* object is a collection of strokes and can be used for symbol recognition. It is within an InkSegment object that we implement Rubine's set of 13 features [66] for identifying 2D symbols; and, before such identification, these objects can be considered "unlabeled" symbols. The ink processing pipeline and InkSegment feature set are displayed in Figure 32 and Table 27, respectively.

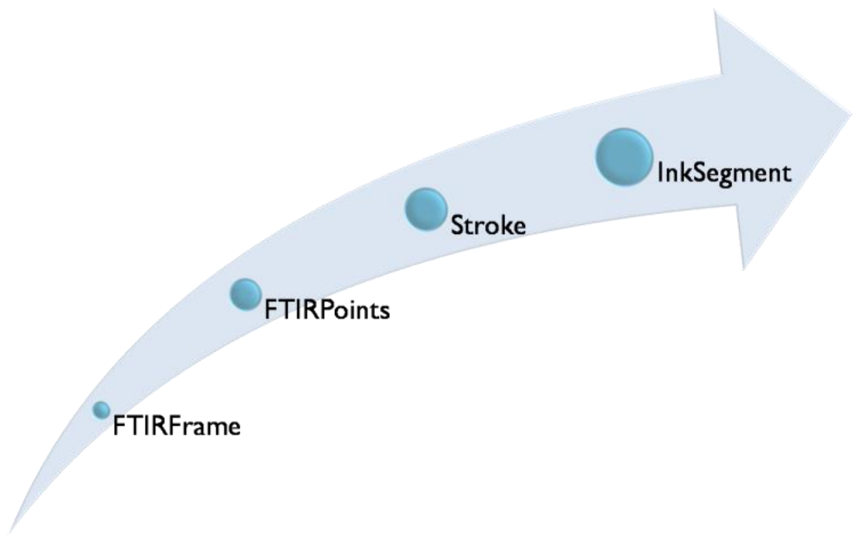


Figure 31: Ink processing

Table 27 InkSegment feature set

Feature
InitialAngleCosine
InitialAngleSine
BoundingBoxLength
BoundingBoxDiagonalAngle
TotalStrokeDistance
FirstLastAngleCosine
FirstLastAngleSine
TotalSegmentLength
TotalAngleTraversed
TotalAngleTraversedAbsolute
TotalSquaredAngleTraversed
MaximumSpeedSquared
Duration

A.2.2 Machine Learning

With a hardware and software platform capable of accepting ink data from a multi-touch surface, we move to the task of comparing machine learning algorithms to identify an appropriate solution for training and classifying multi-touch ink symbols. We can describe the machine learning problem as follows:

- Task *T*: Recognize handwritten symbols collected as a series of strokes from a multi-touch surface.
- Performance Measure *P*: Percent of symbols correctly classified
- Training Experience *E*: A database of labeled symbols
- Target Function *V*: V: InkSegment → Real Number (mapped back to a labeled symbol according to the classifier)
- Target Function Representation: Transform an InkSegment into a feature vector (array of real numbers)

We chose three algorithms for comparison: 1) a Linear classifier; 2) AdaBoost; and 3) an Artificial Neural Network evolved using Neuro-Evolution of Augmenting Topologies (NEAT) [72]. To compare these systems, we developed a software system (pictured in Figure 32) which allows the user to specify and commit samples of 2D symbols to a database. We developed a second application to test the collected data sets against each machine learning algorithm and record performance results. A video of the multi-touch ink application used for data collection can be found at

http://www.bespokesoftware.org/wordpress/?p=43.

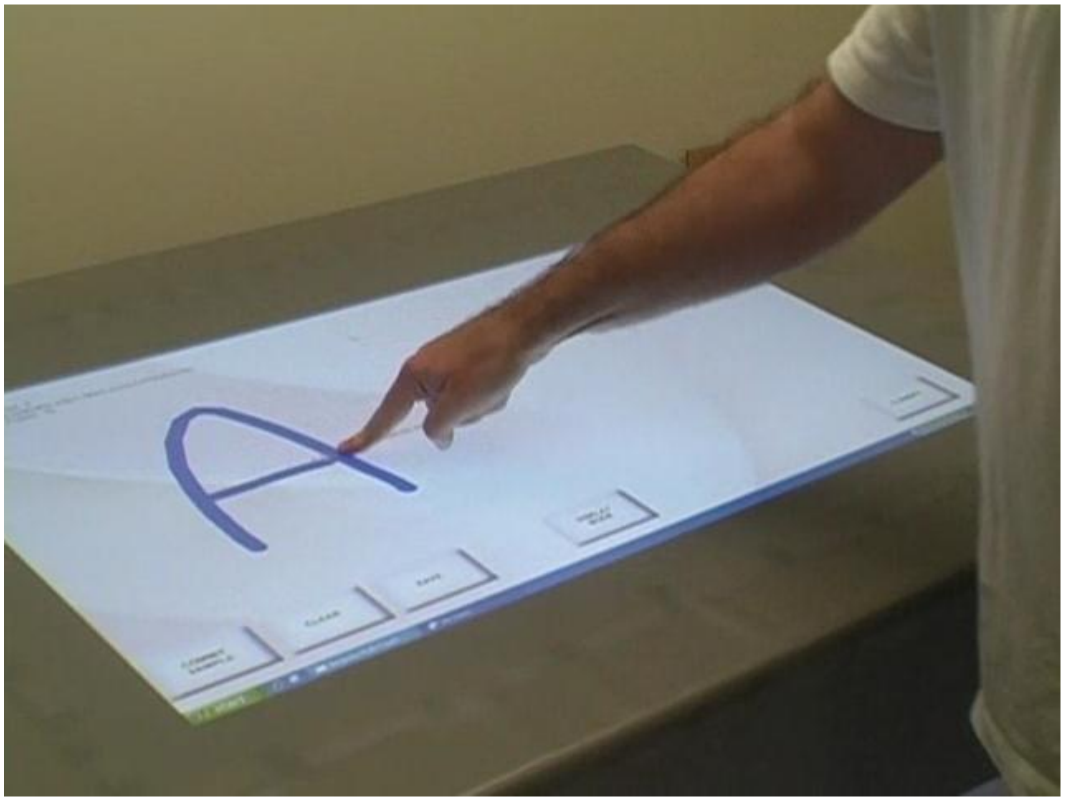


Figure 32: Multi-touch ink capturing software

A.2.2.1 Linear Classifier

With the linear classifier, we calculate an output score of an unlabeled symbol by modulating the symbol's feature vector (the 13 attributes) by a weight vector associated with each trained (labeled) symbol. The highest output is considered the recognized symbol. The weight vectors are computed through a common covariance matrix.

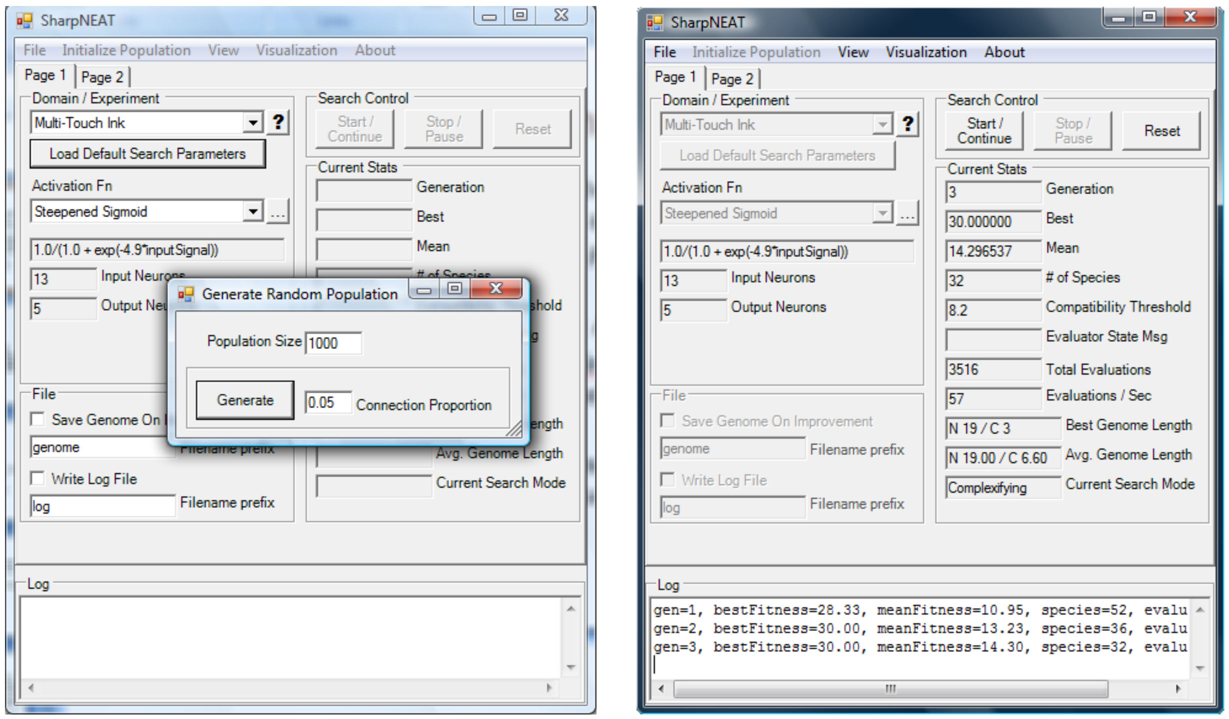
A.2.2.2. AdaBoost

AdaBoost – or Adaptive Boosting – is a machine learning algorithm developed by Freund and Schapire [19]. "(Adaptive Boosting) calls a weak classifier repeatedly in a series of rounds. For each call, a distribution of weights is updated that indicates the importance of examples in the data set for the classification. On each round, the weights of each incorrectly classified example are increased (or alternatively, the weights of each correctly classified example are decreased), so that the new classifier focuses more on those examples [88]." Here again the highest output from weighted sum of the learned recognizers and the unlabeled feature vector is used to determine the correct symbol.

A.2.2.3 Artificial Neural Network (using NEAT)

An Artificial Neural Network represents a group of interconnected nodes. The topology of the network and weighting of the edges between nodes typically changes through input data as it flows through the network during the training phase. Classification is performed by passing an unlabeled input vector through the network and reading the highest corresponding output node. Each output node corresponds to a labeled symbol.

“Neuro-evolution is the evolution of Artificial Neural Networks (ANNs), and in the case of NEAT this means the evolution of both connection weights and network structure, thus NEAT is a Topology and Weight Evolving Artificial Neural Network (TWEANN) strategy [72].” Figure 33 pictures the user interface for evolving an ANN using an open-source NEAT implementation called *SharpNEAT*. By specifying a fitness evaluator specific to our training set, we were able to use this tool to generate an ANN used for subsequent classification.



Domain / Experiment: Multi-Touch Ink
Activation Fn: Steepened Sigmoid
1.0/(1.0 + exp(-4.9*inputSignal))
13 Input Neurons
5 Output Neurons
File: Save Genome On Improvement; genome Filename prefix; Write Log File; log Filename prefix

Generate Random Population
Population Size 1000
0.05 Connection Proportion

Current Stats:
3 Generation
30.000000 Best
14.296537 Mean
32 # of Species
8.2 Compatibility Threshold
Evaluator State Msg
3516 Total Evaluations
57 Evaluations / Sec
N 19 / C 3 Best Genome Length
N 19.00 / C 6.60 Avg. Genome Length
Complexifying Current Search Mode

Log
```
gen=1, bestFitness=28.33, meanFitness=10.95, species=52, evalu
gen=2, bestFitness=30.00, meanFitness=13.23, species=36, evalu
gen=3, bestFitness=30.00, meanFitness=14.30, species=32, evalu
```

Figure 33: ANN evolution with SharpNEAT

A.2.3 Results

While the chief performance measure of the machine learning algorithm is its accuracy, our analysis also considered the time required to train and classify. In the comparison we chose four variations on the data: 1) 12 samples per symbol with 5 symbols; 2) 24 samples per symbol with 5 symbols; 3) 12 samples per symbol with 10 symbols; 4) 24 samples per symbol with 10 symbols. We used 75% of the data to train the algorithms and 25% for validation. The results are listed in Tables 28-31.

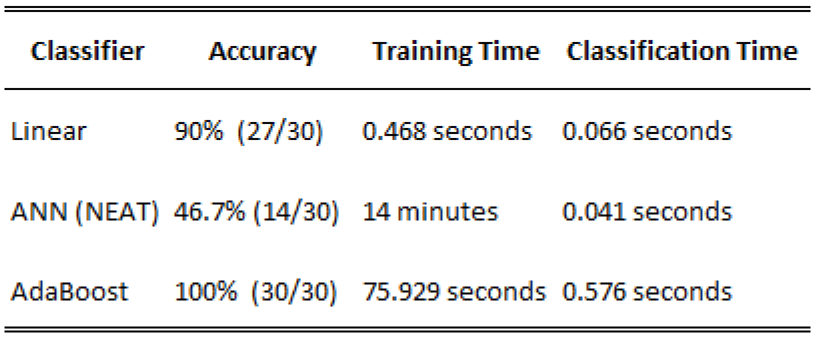
Table 28 2D symbol recognition results: 12 samples per symbol (5 symbols)

Classifier	Accuracy	Training Time	Classification Time
Linear	100% (15/15)	0.273 seconds	0.025 seconds
ANN (NEAT)	66.7% (10/15)	7 minutes	0.039 seconds
AdaBoost	100% (15/15)	21.701 seconds	0.066 seconds

Table 29 2D symbol recognition results: 24 samples per symbol (5 Symbols)

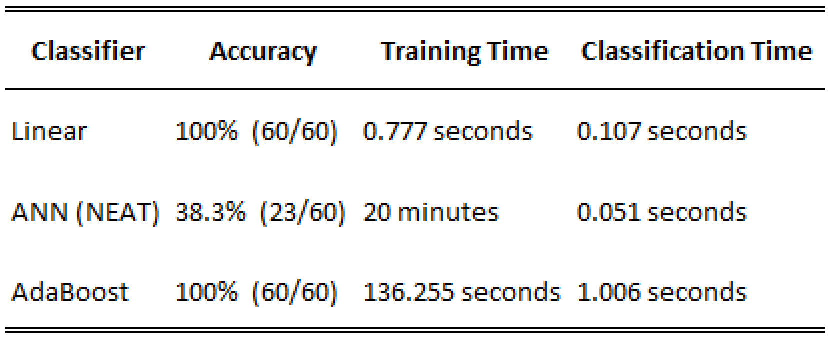
Classifier	Accuracy	Training Time	Classification Time
Linear	100% (30/30)	0.461 seconds	0.037 seconds
ANN (NEAT)	96.7% (29/30)	15 minutes	0.042 seconds
AdaBoost	100% (30/30)	35.147 seconds	0.118 seconds

Table 30 2D symbol recognition results: 12 Samples per symbol (10 Symbols)



Classifier	Accuracy	Training Time	Classification Time
Linear	90% (27/30)	0.468 seconds	0.066 seconds
ANN (NEAT)	46.7% (14/30)	14 minutes	0.041 seconds
AdaBoost	100% (30/30)	75.929 seconds	0.576 seconds

Table 31 2D symbol recognition results: 24 Samples per symbol (10 Symbols)



Classifier	Accuracy	Training Time	Classification Time
Linear	100% (60/60)	0.777 seconds	0.107 seconds
ANN (NEAT)	38.3% (23/60)	20 minutes	0.051 seconds
AdaBoost	100% (60/60)	136.255 seconds	1.006 seconds

A.2.4 Discussion

The most notable observation is the poor performance of the ANN evolved using NEAT. The training of the ANN was stopped once the maximum fitness was found, or at a maximum of 50 generations or 20 minutes (whichever came first). The maximum fitness was calculated as *maxFitness = Number of Symbols * 10* (e.g. 5 symbols could produce a maximum fitness of 50). Maximum fitness was only reached for the first data variation: 12 samples per symbol with 5 symbols. For the remaining treatments, the maximum evolved fitness was found to be 47.78/50, 48.89/100, and 40/100, respectively. These low fitness values account for the poor accuracy of the ANN/NEAT classifier.

The AdaBoost algorithm performed at 100% accuracy across all data variations, with the Linear classifier only missing this mark for treatment 3. However, the classification time for AdaBoost was consistently slower than the Linear classifier as was the training time. For these reasons, we choose the Linear classifier as the best overall choice for accuracy, training time, and classification time.

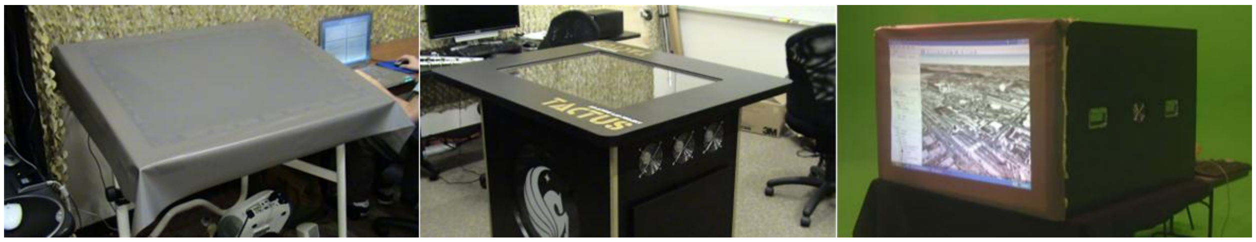
A.2.4 Summary

Multi-touch technology offers exciting research opportunities for human-computer interaction. We can leverage the academic and commercial investments in single-touch computing to propel multi-touch. This is what we have done in the work presented in this section. We have adapted digital ink collection to a multi-touch platform, in order to support pen-based user interfaces and symbol recognition. We have found an appropriate machine learning algorithm for training and classifying stroke data through a comparison of three algorithms and four data variations.

Future efforts include the development of user-definable areas of a multi-touch surface that can be designated as symbol recognition areas. We would also like to incorporate the symbol recognition system with a customizable gesture vocabulary for performing common multi-touch operations.

A.3 Experimentation with Multi-Touch Form Factors

Prior to the work on general-purpose multi-touch computing, we constructed three multi-touch hardware platforms, each with a different form factor. These platforms, pictures in Figure 34, support different usage patterns and chronicle our advancements with multi-touch hardware as each design incorporated knowledge gained from the previous efforts. All of these platforms employ Frustrate Total Internal Reflection (FTIR) for detecting surface contacts, and the Bespoke Multi-Touch Framework for developing software applications.



(a) (b) (c)

Figure 34: Multi-touch hardware platforms: single-user drafting table (a), multi-user sit/stand table (b), multi-user tileable wall display (c)

A.3.1 Single-User Drafting Table

Pictured in Figure 34a, this table was our first attempt at developing a multi-touch hardware platform. By repurposing a traditional drafting-table, we developed a height-and-angle adjustable surface with affordance for a single user. The FTIR waveguide is a 2'x3'x0.5" sheet of cast acrylic supported by a 2.5'x3.5'x2" medium-density fibreboard (MDF) frame. An array of 32 infrared LEDs is embedded into holes, drilled at consistent intervals into the edges of the acrylic. The quantity, and distribution of LEDS, produces ample light to fully permeate the waveguide and produce readily detectable surface contacts. Draped over the surface is a sheet of Rosco Gray, 7mm thick PVC, rear-projection material (the diffuser). Sandwiched between the acrylic and the diffuser is a 1mm thick layer of SORTA-Clear 40 silicone rubber to help couple the user's fingers to the acrylic. Beneath the table are a short-throw projector and an off-the-shelf webcam modified to detect infrared light. Both elements are oriented orthogonally to the waveguide but are only coarsely calibrated. The software calibration system within the Bespoke Multi-Touch Framework corrects for orientation offsets between the camera and projector.

This platform was developed as a prototype with the purpose of educating us on the requirements of constructing a multi-touch surface. As such, it is not conducive to commercial deployment, but it served its purpose as a learning tool and offered insights into the ergonomics of a single-user platform. Particularly, we observed that, while the height of the table (adjustable from 2'-4') allowed the user to sit or stand at the platform, standing was the more common usage scenario. This is an important observation when considering the application of multi-touch to general- purpose computing, because a standing position is not ideal for long-term use. Thus we pose the question: "what about this form factor led the user to prefer standing to sitting?" The observation that users preferred to stand at the table (even when a seat was available) was an anecdotal one and requires a formal study to validate. However, even without a formal study, and with the assumption that the observation is valid, we can speculate on the preference to a standing position:

- The environment of the multi-touch surface may have discouraged the user from taking a seated position.
- The novelty of the multi-touch platform may have discouraged a seated position.
- This particular platform, being visibly unpolished and representative of a prototype, may have discouraged a seated position.
- The application of multi-touch differs from drawing/drafting so much that the affordances of the form factor do not transfer.
- The specific software applications running on the multi-touch table did not encourage the user to take a seated position.
- The duration of using the multi-touch table did not encourage the user to take a seated position.

Recall that we repurposed a typical drafting table for this multi-touch platform. With the popularity and long-term adoption of this form-factor, for the applications of drawing or drafting, we can assume that the drafting table form factor is not, in itself, the reason that users shied away from sitting at the table. Again, without a formal study to answer this question, we are left with only speculation. However, it is clear that there are affordances to a seated position, and anecdotally, this initial multi-touch hardware platform did not provide them. Furthermore, we propose that a seated position is necessary for long-term-single-session use of a general-purpose computing platform, and that a successful implementation of a general-purpose multi-touch platform must therefore encourage the user to sit.

A.3.2 Multi-User Sit/Stand Table

Our second multi-touch hardware platform, pictured in Figure 34b, was a professionally crafted table with sit or stand affordances for up to four simultaneous users. This platform encased the short-throw projector, camera, and computer within a sturdy cabinet that easily supports leaning elbows, kicking or bumping. The core design of the FTIR configuration remained largely unchanged from our first prototype (Figure 34a). However, this platform was much more carefully crafted and presents a clean, professional and polished face. Indeed, this platform has been used as a showpiece for numerous private and public demonstrations of the technology.

We anecdotally observed different user behavior when demonstrating the platform versus using the device for longer periods of time. Specifically, we noted that – even with height-adjustable chairs provided – demonstrators and associated users almost always worked with the platform from a standing position. Software developers, on the other hand, while writing software or experimenting with applications on the device, tended to sit at the table. This observation supports the notion that the environment of the device (both physical environment and the format in which it was being used) affects its usage. Moreover, we speculate that the sturdy, professional appearance of the platform emboldened longer-term users to sit at the platform – where they did not do so with the prototype. With the core hardware elements protected within the cabinet, the user could feel confident that their usage would not break the device or misalign the projector or camera.

A.3.3 Multi-User Tileable Wall Display

Our third hardware platform, pictured in Figure 34c, addressed an entirely different use case – that of standing users interacting with a wall display. This device was intended for use in a museum and for up to two simultaneous users. The FTIR waveguide is a 4’x3’x0.5” sheet of cast acrylic with the projector, camera, and computer housed within an approximately 4’x3’x4’ cabinet. The design also provides for multiple, identical cabinets to be stacked alongside and atop each other. For example, a 2x2 array would present an 8’ wide, 6’ high interactive display. The venting fans were designed to create a channel of air through a row of stacked cabinets and dissipate heat from the computers and projectors. The cabinets can be operated as stand-alone displays or as a single integrated input/output system with a combined display size. To treat the input from each cabinet as part of an integrated whole, we designate one of the cabinet computers (or an external computer) as the master device. All secondary cabinets transmit their multi-touch input, using the Bespoke Multi-Touch Framework and Open-Sound Control Library, to the master for integration into a single frame of data. In this fashion, the task of vision-based surface contact detection is distributed to the secondary cabinets, while the master operates the user-facing software application. Rendering to the array can be distributed in one of several ways: 1) the master device can contain multiple video outputs (1 for each cabinet in the array); 2) daisy chain a GPU sync cable to each cabinet computer (GPU vender specific); license a third-party distributed rendering system such as Mersive [74]. While we have designed the systems necessary for tiling the cabinets, to date we have only constructed one multi-touch wall display.

The use case for this platform was never intended for general-purpose computing. Thus, we cannot make many observations of this device along these lines. However, constructing a third cabinet, and for a distinctly different use from the first two, provided an opportunity to refine our techniques in multi-touch hardware and software development.

APPENDIX B: WORD PROCESSING PARAGRAPHS

A George Divided

You have no idea of the magnitude of this thing. If she is allowed to infiltrate this world then George Costanza as you know him ceases to exist. You see, right now I have *Relationship George*. But there is also *Independent George*. That's the George you know, the George you grew up with... *Movie George*, *Coffee Shop George*, *Liar George*, *Bawdy George*.

I love that George.

Me too, and he's dying Jerry. If *Relationship George* walks through this door, he will kill *Independent George*. **A George divided against itself cannot stand!**

Man Made Prisons

What are you thinking about Jerry? *Marriage? Family?* They're prisons. Man made prisons. You're doing time. You get up in the morning. She's there. You go to sleep at night. She's there. It's like you gotta ask permission to use the bathroom. *Is it alright if I use the bathroom now?*

Really?

Yeah, and you can forget about watching TV while you're eating.

I can?

Oh, yeah. You know why? Because it's dinner time. And you know what you do at dinner? You talk about your day. How was your day today? Did you have a good day today or a bad day today?

The Marine Biologist 1
So I started to walk into the water. I won't lie to you boys, **I was terrified**! But I pressed on and as I made my way passed the breakers a strange calm came over me. I don't know if it was *divine intervention* or the kinship of all living things but I tell you Jerry at that moment I was a *marine biologist*! *George* I was just reading this thing in the papers, **it's amazing**! I know I was just telling them the story. Come on George, finish the story.

The Marine Biologist 2
The sea was angry that day my friends, like an *old man* trying to return soup at a deli! I got about fifty-feet out and then suddenly the **great beast** appeared before me. I tell ya he was ten stories high if he was a foot. As if sensing my presence he gave out a **big bellow**. I said, *"Easy big fella*!" And then as I watched him struggling I realized something was obstructing his breathing. From where I was standing I could see directly into the eye of the great fish! **Mammal.** Whatever.

Batman

Then everybody is screaming, because *the driver*, he's passed out from all the commotion... **the bus is out of control**! So, I grab him by the collar, I take him out of the seat, I get behind the wheel and now I'm driving the bus.

You're Batman!

Yeah. Yeah, I am *Batman*. Then *the mugger*, he comes to, and he starts choking me! So I'm fighting him off with one hand and I kept driving the bus with the other. Then I managed to open up the door, and I kicked him out the door with my foot, you know, at the next stop.

You kept making all the stops!?

Well, people kept ringing the bell!

The Pick

But there was no *pick*! I did not *pick*! There was no *pick*!

I gotta go.

No! No *pick*!

And what if I did do it? Even though I admit to nothing, and never will. What does that make *me*? And I'm not here just defending myself but all those *Pickers* out there who've been caught. Each and every one of them, who has to suffer the shame and humiliation because of people like you.

Are we not *human*?! If we pick, do we not *bleed*?! **I am not an animal!**

APPENDIX C: INSTITUTIONAL REVIEW BOARD DOCUMENTS

C.1 Pre-Study Questionnaire

This questionnaire was used for all four user studies.

1. Year of Birth: __________

2. Gender: Male_________ Female____________

3. Handedness: Left Handed_________ Right Handed_________

4. Do you have any physical impairment that limits you from normal computer use?

   Yes______ No______

5. Do you have any visual impairment that limits you from normal computer use?

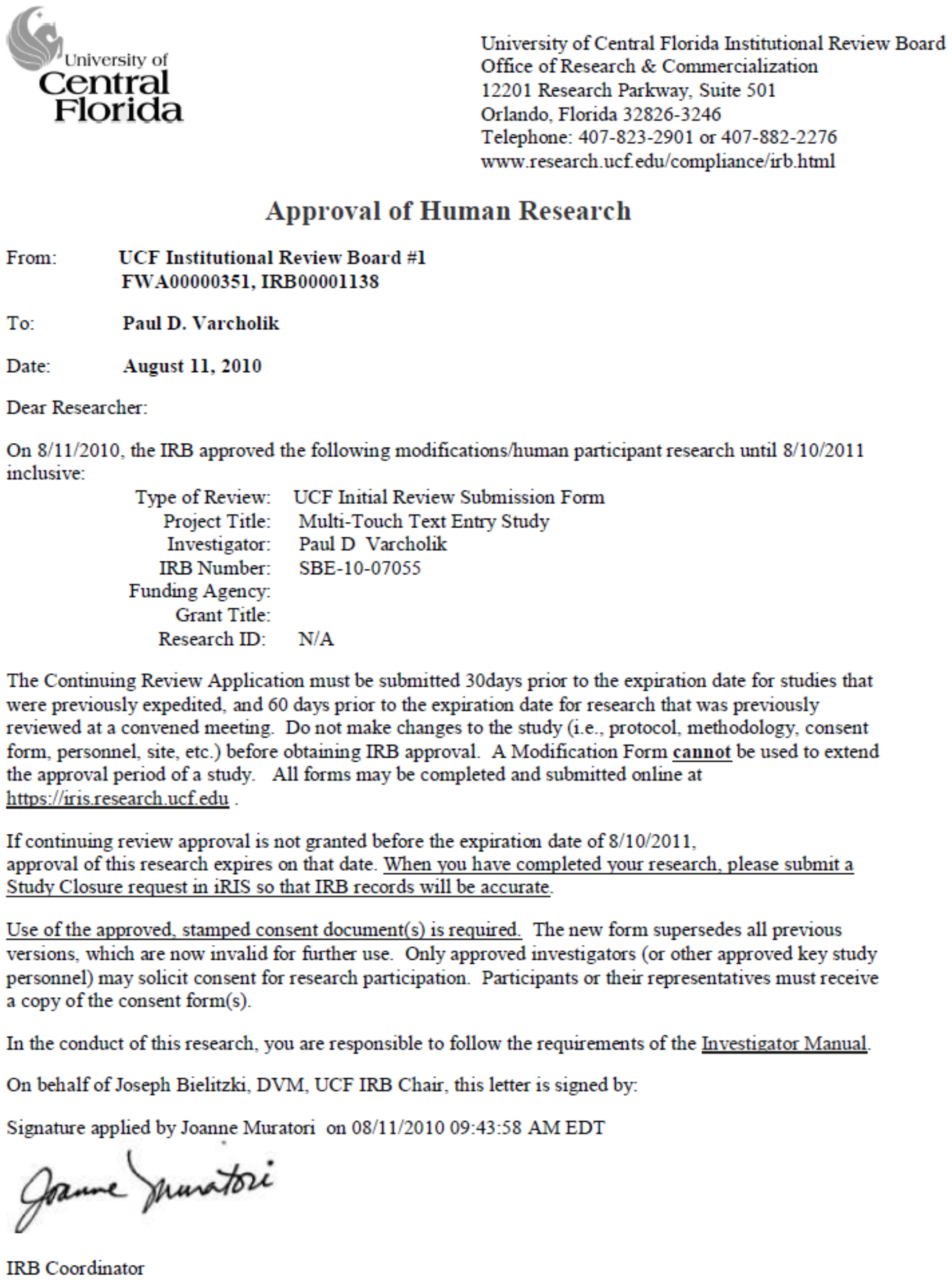
   Yes_______ No________

6. Rate your familiarity with touch screen computer systems (including single-touch or multi-touch platforms):

1	2	3	4	5
Never Used	Used once or twice	Use once a month	Use once a week	Use a few times a week

C.2 Text Entry Study

C.2.1 Approval of Human Research



University of Central Florida

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Paul D. Varcholik**

Date: **August 11, 2010**

Dear Researcher:

On 8/11/2010, the IRB approved the following modifications/human participant research until 8/10/2011 inclusive:

Type of Review:	UCF Initial Review Submission Form
Project Title:	Multi-Touch Text Entry Study
Investigator:	Paul D Varcholik
IRB Number:	SBE-10-07055
Funding Agency:	
Grant Title:	
Research ID:	N/A

The Continuing Review Application must be submitted 30days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu .

If continuing review approval is not granted before the expiration date of 8/10/2011, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Joseph Bielitzki, DVM, UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori on 08/11/2010 09:43:58 AM EDT

Joanne Muratori

IRB Coordinator

C.2.2 Post-Study Questionnaire

1. Rate the difficulty of entering text on the physical keyboard:

1 Very Easy	2	3	4	5 Very Difficult

2. Rate the difficulty of entering text on the multi-touch platform in single-touch mode:

1 Very Easy	2	3	4	5 Very Difficult

3. Rate the difficulty of entering text on the multi-touch platform in multi-touch mode:

1 Very Easy	2	3	4	5 Very Difficult

4. The multi-touch platform accurately detected my intended key presses.

1 Strongly Disagree	2	3	4	5 Strongly Agree

5. Rate the relative speed of the two touch-input techniques for text entry on the multi-touch platform:

1 Multi-Touch	2	3	4 No Difference	5	6	7 Single-Touch

6. Rate the relative accuracy of the two touch-input techniques for text entry on the multi-touch platform:

1 Multi-Touch	2	3	4 No Difference	5	6	7 Single-Touch

7. Rate your preference of the two touch-input techniques for text entry on the multi-touch platform.

1	2	3	4	5	6	7
Multi-Touch			No Preference			Single-Touch

8. From the text entry methods/devices used in this study, rate your preferred text entry method (as 1, 2 and 3; 1 highest):

Physical Keyboard	________
Multi-Touch Keyboard in single-touch mode	________
Multi-Touch Keyboard in multi-touch mode	________

9. Would you use the multi-touch device to enter text for everyday computer activities?

   Yes______ No______ Unsure____

   Why or why not?

____

____

____

____

____

____

10. Do you have any comments?

____

____

____

____

____

____

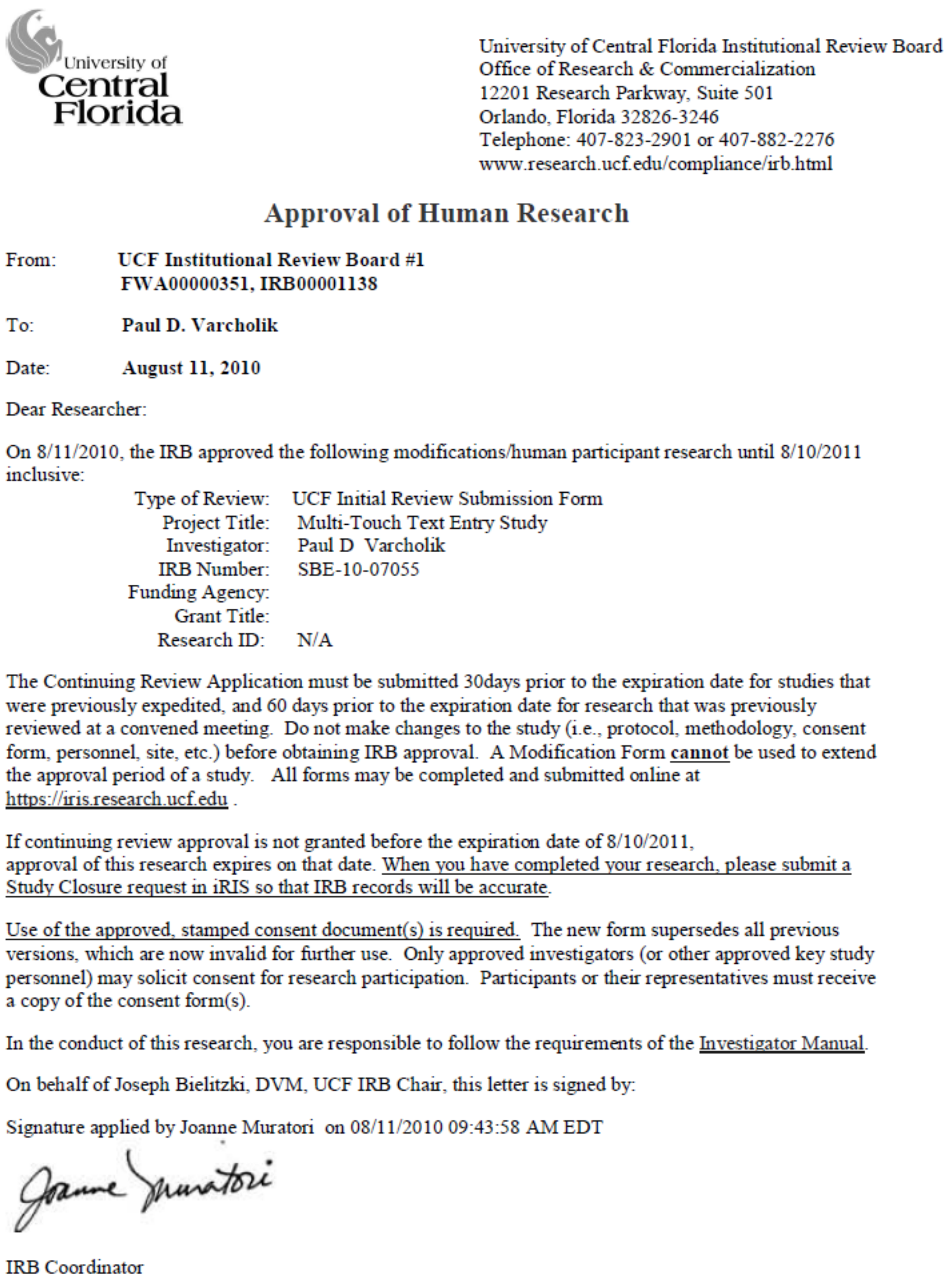
____

____

____

C.3 Text Entry Improvement Study

C.3.1 Approval of Human Research



University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Paul D. Varcholik**

Date: **August 11, 2010**

Dear Researcher:

On 8/11/2010, the IRB approved the following modifications/human participant research until 8/10/2011 inclusive:

Type of Review:	UCF Initial Review Submission Form
Project Title:	Multi-Touch Text Entry Study
Investigator:	Paul D Varcholik
IRB Number:	SBE-10-07055
Funding Agency:	
Grant Title:	
Research ID:	N/A

The Continuing Review Application must be submitted 30days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu .

If continuing review approval is not granted before the expiration date of 8/10/2011, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Joseph Bielitzki, DVM, UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori on 08/11/2010 09:43:58 AM EDT

Joanne Muratori

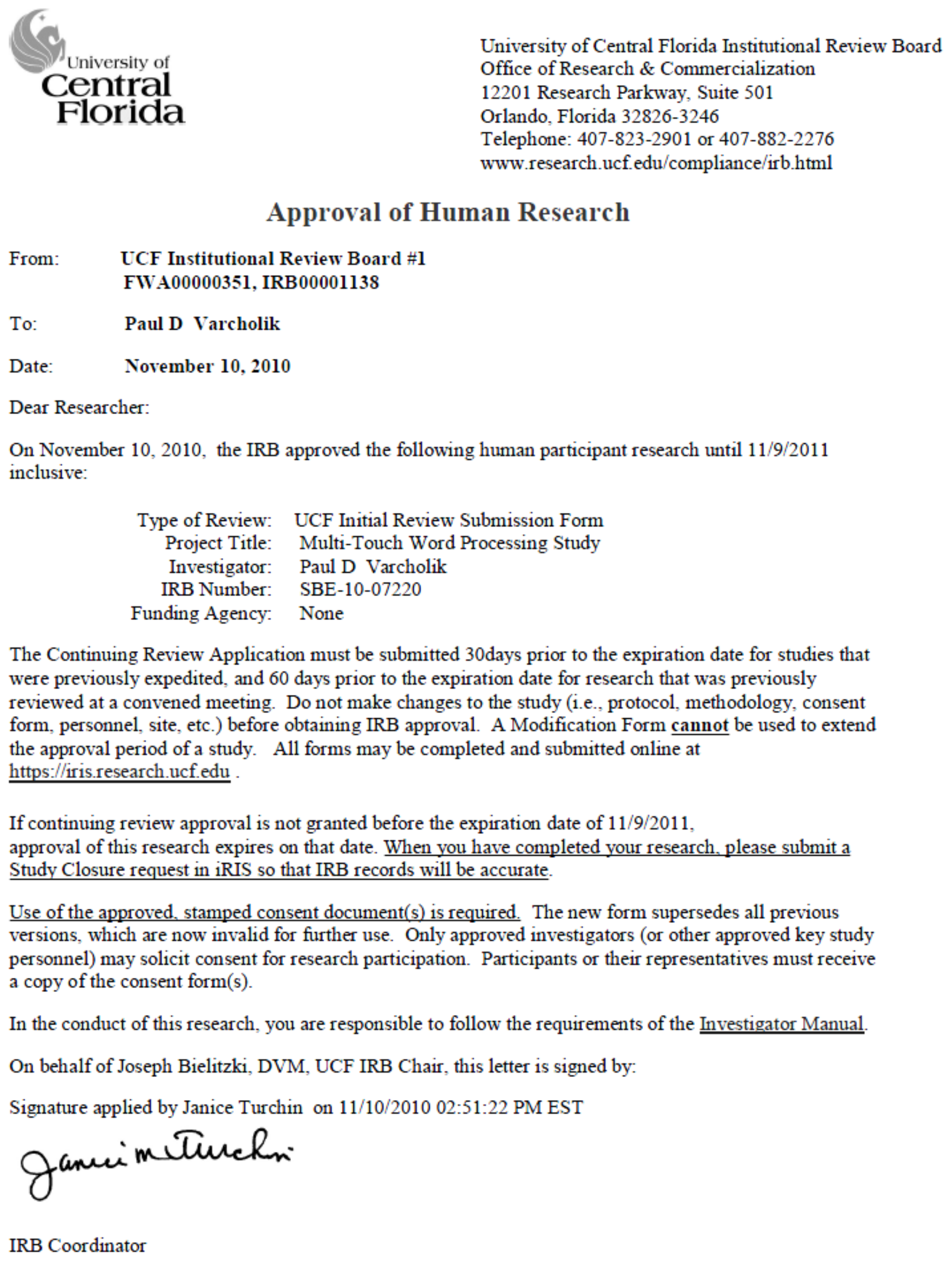
IRB Coordinator

C.3.2 Post-Study Questionnaire

1. Which was your preferred text entry technique? Key-Down, Key-Up, No-Preference

C.4 Word Processing Technique Study

C.4.1 Approval of Human Research



University of Central Florida

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Paul D Varcholik**

Date: **November 10, 2010**

Dear Researcher:

On November 10, 2010, the IRB approved the following human participant research until 11/9/2011 inclusive:

Type of Review:	UCF Initial Review Submission Form
Project Title:	Multi-Touch Word Processing Study
Investigator:	Paul D Varcholik
IRB Number:	SBE-10-07220
Funding Agency:	None

The Continuing Review Application must be submitted 30days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu .

If continuing review approval is not granted before the expiration date of 11/9/2011, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Joseph Bielitzki, DVM, UCF IRB Chair, this letter is signed by:

Signature applied by Janice Turchin on 11/10/2010 02:51:22 PM EST

[illegible]

IRB Coordinator

C.4.2 Post-Study Questionnaire

1. The multi-touch platform accurately detected my intended key presses.

1 Strongly Disagree	2	3	4	5 Strongly Agree

2. The multi-touch platform accurately detected my intended mouse interactions.

1 Strongly Disagree	2	3	4	5 Strongly Agree

3. From the word processing methods used in this study, rate your preferred method (as 1, 2 and 3; 1 highest):

Direct Mouse Interaction – Key Stroke on DOWN	_________
Direct Mouse Interaction – Key Stroke on UP	_________
Indirect Mouse Interaction (Virtual Mouse Pad)	_________

4. Would you use the multi-touch device for everyday word processing activities?
   Yes______ No______ Unsure____

   Why or why not?

____

____

____

____

____

5. Do you have any comments?

____

____

____

____

____

____

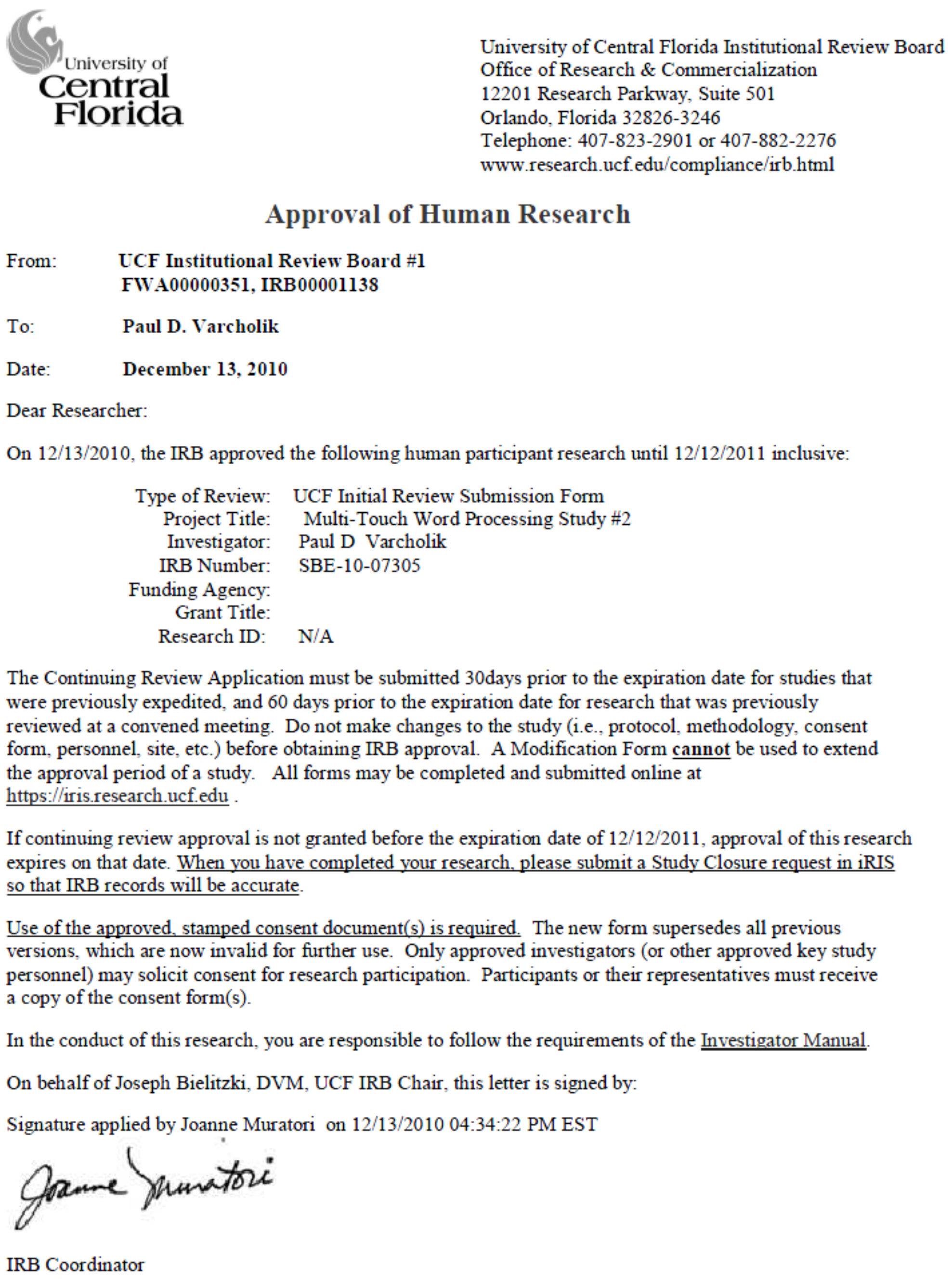
____

____

____

C.5 Word Processing Platform Study

C.5.1 Approval of Human Research



University of Central Florida

University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Human Research

From: **UCF Institutional Review Board #1**
FWA00000351, IRB00001138

To: **Paul D. Varcholik**

Date: **December 13, 2010**

Dear Researcher:

On 12/13/2010, the IRB approved the following human participant research until 12/12/2011 inclusive:

Type of Review:	UCF Initial Review Submission Form
Project Title:	Multi-Touch Word Processing Study #2
Investigator:	Paul D Varcholik
IRB Number:	SBE-10-07305
Funding Agency:	
Grant Title:	
Research ID:	N/A

The Continuing Review Application must be submitted 30days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form **cannot** be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu .

If continuing review approval is not granted before the expiration date of 12/12/2011, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Joseph Bielitzki, DVM, UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori on 12/13/2010 04:34:22 PM EST

[illegible]

IRB Coordinator

C.5.2 Post-Study Questionnaire

1. The multi-touch platform accurately detected my intended key presses.

1	2	3	4	5
Strongly Disagree				Strongly Agree

2. The multi-touch platform accurately detected my intended mouse interactions.

1	2	3	4	5
Strongly Disagree				Strongly Agree

3. From the platforms used in this study, rate your preferred method (as 1, 2 and 3; 1 highest):

Desktop keyboard and mouse	_________
Laptop keyboard and integrated track pad	_________
Multi-touch	_________

4. Would you use the multi-touch device for everyday word processing activities?
   Yes_______ No_______ Unsure____

   Why or why not?

____

____

____

____

____

5. Do you have any comments?

____

____

____

____

____

____

____

____

____

APPENDIX D: STUDY COMMENTS

D.1 Text Entry Study

Table 32 Text entry study would-use comments

Would Use?	**Would Use Comments**
No	It seems very sensitive and I had to constantly look at the keyboard on screen in order to know what I was typing. On a physical keyboard, I am able to know when I make mistakes even if I am not looking at the text I need to type or the keyboard itself. I didn't like that I couldn't rest my hands on the keyboard because it would detect key presses when I didn't mean for them to be. This was probably my most consistent problem.
No	too many accidental inputs. no recoil when touching, which will cause wear on hands.
No	there isnt any haotuc response, no physical feedback. Given that a physical keyvoard is present I would prefer to use that. If the multi touch was the only option i wouldnt mind using it....i coukd see myself getting used to it. Only in two hand input mode though. The sibgle finger mode is very frustrating and forces you to chicken peck.
Unsure	lack of physical feedback on a touch screen that one gets on a normal keyboard
Yes	Ease of use and beautiful ui
No	only for basic task like texting
Unsure	The physical keyboard isnt as sensitive and so doesnt allow for as many mistakes.
No	A physical keyboard fels better because of the feedback from the pressure of a physical key. I make many mistakes dure to stray touches on the multi-touch.
No	I'm not used to the typing on a texturless surface. Theres less feedback as to where my hands/fingers are at any given time.
Yes	I'm sure I'd adapt to it, and can see advantages of auto-spell kinds of features. But typing is so discretely unit-by-unit, that I don't think multi-anything helps that often.
No	It is very tiring, and the keyboard structure is different than what I'm used to. I like having physical keys.
Yes	I think I could get used to it with practice.
Unsure	I think that I'm just not ussed to it for everyday computer use. However, I see no reason why, eventually, I couldn't get used to it.
No	cannot type as fast as needed.
No	Would have to get use to it first dealing with the speed issue and the factthat youbcant lean on the screen

Would Use?	Would Use Comments
Yes	mutitouch interfaces are pretty space efficent, and allow for some pretty cool features/grestures other than typing; although if there was a better way to provide more tactile feedback when using a multitouch keyboard/interdace, it would be even better. In my opinion, multitouch controls still don't feel as precise as physical tactile input such as a keyboard, or buttons on a game pad.
No	i have tangible reference of where the keys are on a physical keyboard
Yes	I enjoy the technology of it all but i feel that it is isnt ergonomic enough as a regular keyboard
No	the software i use requires frequent use of hotkeys and constantly held keys for view manipulation (such as Alt key for Maya).
Unsure	If I had the option to use a physical keyboard I definitely would, but i'm plenty happy using my iphone keyboard for emails and other everyday use if that's what's available.
Unsure	There is more fatigue that occurs with a touch screen
Yes	i use an android phone with multitouch enabled
Unsure	It did not seem to have the feedback to let me know when a key had been pressed, or the incorrect key had been pressed.
Yes	It could be more convenient to have around. The large keyboard was very natural to use compared to a normal, physical keyboard, but a smaller touch keyboard would not be as comfortable. I suppose a full sized touch keyboard may not be as convient to carry around, etc.
Yes	Its necessary with the new cell phones.
No	the lack of a physical 'key' on my fingertip makes it harder to type confidently. my typing slowed because I had to continue to correct mis-spelled words, all due to my fingers not being in the right place.
Yes	I woulld but it would take some getting used to. The biggest probl2m ia no4 being able to rest fingers on the keys, which may contribute to stress in the hand.
Yes	I use my iPhone everyday. The problem is that every is familiar with physical keyboards, whereas there is no standard for the construction of a digital keyboard. With the physical I was looking at the screen, with the digital i was looking at my hands (and thus missing mistakes until the end when I looked up). With familiarity I think a touch screen could work. This was too sensitive, if my fingertips brushed the screen it counted as pressing the keys
Yes	If a proper angle and sizing were used i would find this very acceptable.
Yes	I use the iphone on a daily basis This seems better
Yes	this kind of typing is becomming more common, especially with texting devices, so with proper sound and touch feedback - and practice- this typing could become just as "easy" as keyboards

Would Use?	**Would Use Comments**
No	arms became more tired felt like i made more mistakes
No	too much slower and more painful for the hand positioning
No	There is no tactile feedback. I'd need to really get used to it. It would also help if it was bigger; it felt too small to type with, even though it was the same size as the laptop keyboard I am now typing on
Yes	easy interface, would just take some time to get used to
No	i kept misspell8ng alot of things. ut would be frustrating to correct them all
Yes	once i became more familiar with the device i think i would be able to use it just as easily as the physical keyboard.
Unsure	i would use it for drawing and art stuff, but like the tactile s;ensation of a real keyboard
No	benefits don't outweigh the difficulty
No	I prefer the tactile feedback provided by a physical keyboard.
No	The multi-touch device requires screenspace that could get in the way if you are multitasking
Unsure	Would have to get used to it
Yes	I use my touchscreen phone for everday but not that much!

Table 33 Text entry study general comments

	Comments
1	much easier to type on the single touch than multitouch...
2	Overall the physical keyboard was better at giving feedback as well as for speed in my opinion.
3	I think that most of the incorrec letters I typed on the touchpad were when I tried to rest my pinky fingers on the screen.
4	the difference between multi- and single- touch was arbitrary to me, since it really only came to play in 2 situations: 1) when capitalizing one letter or word (not a big deal), and 2) when my palm was accidentally resting on the touchscreen. I wouldn't really have a preference except for if my hand the did touch then in the single-touch setting, it wouldn't accept my finger touch. Maybe down the road, after you hit 'spacebar' and the word before it is a real word, there can be an audio program that says word just typed.
5	the arrow keys would be nice. i think having to remember the phrase takes away from the actual typing exercise.
6	Typing gets exhausting after a short anount of time on the touchpad. The touchpad becomes much easier to use after getting used to it, which only takes a few minnutes.
7	i like to rest my palms on something this is why i prefered the two touch style
8	except for capitalization, there is little to no difference between multi and single touch as far as typing complete and accurate phrases. having to hit Caps Lock makes me lose a comfortable position for typing. the size of the keyboard can also hinder typing

D.2 Word Processing Technique Study

Table 34 Word processing technique study would-use comments

Would Use?	Would Use Comments
No	too slow for me, have to think too much
No	Far slower! And kills my hands.
No	I can feel my way around a real keyboard better, so if I make a mistake I can very quickly get back into typing.
No	It does not feel as efficient as an actual keyboard.
No	The ergonomics make longterm typing tasks difficult.
No	Lack of tactile feedfack slows my typing down by 50% or more.
Yes	as long as sound was included as feedback for each hit
Yes	like to see what i am pressing, with visial and audio effects.

Table 35 Word processing technique study general comments

	Comments
1	i would like audio feedback for the bold/italic keys. the spacing of the keys seems fine in general but i always hit the > not ?, and m not n.

D.3 Word Processing Platform Study

Table 36 Word processing platform study would-use comments

Would Use?	**Would Use Comments**
No	Does not provide the feel of the keys that I am used to for one, and knowing how certain key feel, like the back space bar is important to me. Additionally, the multi touch device tends to be much more sensitive than a standard keyboard. For instance, a light touch of the spacebar, or any key for that matter shows up on the multi-touch, whereas on a regular key board it does not.
No	The feedback is not there. It is hard to differentiate between which keys you are hitting. Especially the shift/control keys. Also the mouse interactions were not as precise as a trackpad or mouse.
No	The familiarity with the desktop keyboard and overall comfortability with these input devices is far more accurate and less time consuming. If given the option I will always choose the desktop keyboard and mouse.
No	I'm so accustomed to typing on a regular keyboard and the way it feels that it just doesn't feel natural when i'm typing on a multi-touch. When typing on the keyboard, I can constantly look at the text that I'm copying from because I can trust that my typing on the keyboard will be accurate. With the multi-touch I found myself looking at the keys instead of the text I was copying.
No	There are better alternatives.
No	My hands could not comfortably rest upong the multi-touch device as I would on a keyboard as doing so would cause it to do unwanted actions. When using the normal keyboard and mouse and the laptop keyboard and trackpad, I was able to rest my hands on the table. Having my hands suspended above the multi-touch device was somewhat tiring for my hands and if I had to use it for my everyday word processing activities I imagine that it would cause my hands to be extremely uncomfortable.
No	No tactile feed back.
Unsure	If I had more exposure to the format, I see it as usable. Once I got acquainted to the layout and sensitivity, I found myself able to type quite well. The major drawback for me was the angle at which I was typing. The other descrepancy was that I'm used to "finger or thumb typing" on touch input devices, so I was unsure whether to approach the exercise from a "standard keyboard" approach or a "smart phone" approach.
No	Slower and not completely responsive.

Would Use?	Would Use Comments
Yes	I prefer touch screen over regular keyboards. There's a bit more of a learning curve, but after I become used to the key placement on a touchscreen I find I type faster.
Unsure	I use my iphone for emails constantly because that's what's most readily available. Likewise, I would use the multitouch if I didn't have a computer handy, but I would always prefer a desktop or laptop for regular use.
Unsure	I found it easiier to "double tap" keys and make more letters than intended. Also, selecting text without a dedicated mouse may be more difficult.
Unsure	It fun to use and it makes certain processes such as highlighting and correcting mistakes easier, but it requires me to look at the keyboard while i type because im not sure where my fingers are. the fact that i cant really "rest" my fingers on top of the keys without really pressing them is also a little uncomfortable.
Yes	My current cellphone has a multitouch screen that I need to use in order to type (no QWERTY external keyboard).
No	No I would not because the sensitivity was a big deal. I was receiving key presses from my palm resting on the screen. I like the clicking when I actually have something that is being pressed.
No	I couldn't rest my fingers on "home row' so I kept having to look down to see where my fingers were
Unsure	Unsure about the replacement capabilities due to familiarity with mouse control. Have to retrain myself with useing and open screen and my whole arm instead of the mouse.

Table 37 Word processing platform study general comments

	Comments
1	Back got tired with the multi-touch. I feel like I have to hold my hands in the air vs. resting them on the keyboard or laptop.
2	Multi-touch input devices have gotten better, but the speed and accuracy at which they can read your input is still a bit off. I am used to typing quite fast on both a desktop keyboard and mouse, or even a laptop keyboard that when I get to use the multi-touch I have this instinct to just blow through it, and this usually comes out all bad on the multi-touch.
3	Multitouch was better than the other methods for selecting Bold and Italics because it was much easier to click the activation buttons, but multitouch is much slower for typing because you have keep checking your hand positioning.
4	The laptop was my favorite because the trackpad was so close to the typing surface. On the desktop I lost a lot of time going to the mouse and back. If I could use ctrl I and ctrl B for formatting, the desktop would be my preferred platform.
5	Feedback on the "B"old and "I"talic buttons might be helpful for the testing
6	The keyboard on the laptop is much easier to use because it is of higher quality than the desktop keyboard, which feels of lower quality and made it slightly more difficult to type with.

LIST OF REFERENCES

1. I.F.S.R. Multitouch Allows for Unlimited Touch Inputs. http://news.softpedia.com/news/I-F-S-R-Multitouch-Allows-for-Unlimited-Touch-Inputs-130953.shtml.

2. 3M MP180 Pocket Projector. http://solutions.3m.com/wps/portal/3M/en_US/Pocket/Projector/Main/Products/MP180/.

3. 3M Touch Systems. http://www.3m.com/touch.

4. Andrea, L., Shumin, Z. and William, B. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Trans. Comput.-Hum. Interact.*, *5* (4). 326-359.1998.

5. Appple iPad. http://www.apple.com/ipad/.

6. iPhone. http://www.apple.com/iphone/.

7. Bi, X., Grossman, T., Matejka, J. and Fitzmaurice, G. Magic Desk: Bringing Multi-Touch Surfaces into Desktop Work *Proceedings of the SIGCHI conference on Human factors in computing systems*, Vancouver, BC, Canada, 2011.

8. Blickenstorfer, C.H. Graffiti: Wow! *Pen Computing Magazine*, 1995, 30-31.

9. Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. *3D User Interfaces: Theory and Practice*. Addison Wesley, 2004.

10. Multi-Touch Systems that I Have Known and Loved. http://www.billbuxton.com/multitouchOverview.html.

11. Buxton, W. and Myers, B. A study in two-handed input *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, Boston, Massachusetts, United States, 1986.

12. Clarkson, E., Clawson, J., Lyons, K. and Starner, T. An empirical study of typing rates on mini-QWERTY keyboards *CHI '05 extended abstracts on Human factors in computing systems*, ACM, Portland, OR, USA, 2005.

13. The inventor behind CNN's election 'Magic Wall'. http://www.cnn.com/2008/TECH/11/04/magic.wall/.

14. Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology *Proceedings of the 14th annual ACM symposium on User interface software and technology*, ACM, Orlando, Florida, 2001.

15. Echtler, F., Huber, M. and Klinker, G. Shadow tracking on multi-touch tables *Proceedings of the working conference on Advanced visual interfaces*, ACM, Napoli, Italy, 2008.

16. ACRYLITE EndLighten. http://www.acrylite-magic.com/AC_EN/Materialien/ACRYLITE_EndLighten_acrylic_sheet.php5.

17. Findlater, L., Wobbrock, J. and Wigdor, D. Typing on Flat Glass: Examining Ten-Finger Expert Typing Patterns on Touch Surfaces *Proceedings of the SIGCHI conference on Human factors in computing systems*, Vancouver, BC, Canada, 2011.

18. FingerWorks. http://www.fingerworks.com/.

19. Freund, Y. and Schapire, R. A desicion-theoretic generalization of on-line learning and an application to boosting. in *Computational Learning Theory*, Springer Berlin / Heidelberg, 1995, 23-37.

20. Fu, C.-W., Goh, W.-B. and Ng, J.A. Multi-touch techniques for exploring large-scale 3D astrophysical simulations *Proceedings of the 28th international conference on Human factors in computing systems*, ACM, Atlanta, Georgia, USA, 2010.

21. George, W.F., Hiroshi, I. and William, A.S.B. Bricks: laying the foundations for graspable user interfaces *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co., Denver, Colorado, United States, 1995.

22. Goldberg, D. and Richardson, C. Touch-typing with a stylus *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, ACM, Amsterdam, The Netherlands, 1993.

23. Gordon, K., George, F., Thomas, B. and Bill, B. The design of a GUI paradigm based on tablets, two-hands, and transparency *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, Atlanta, Georgia, United States, 1997.

24. Han, Y.J. Low-cost multi-touch sensing through frustrated total internal reflection *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM, Seattle, WA, USA, 2005.

25. Han, Y.J. Multi-touch interaction wall *ACM SIGGRAPH 2006 Emerging technologies*, ACM, Boston, Massachusetts, 2006.

26. Hinrichs, U., Carpendale, S., Scott, S. and Pattison, E., Interface Currents: Supporting Fluent Collaboration on Tabletop Displays. in *Symposium on Smart Graphics*, (2005).

27. Hinrichs, U., Hancock, M., Collins, C. and Carpendale, S., Examination of Text-Entry Methods for Tabletop Displays. in *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, (2007), 105-112.

28. Hinrichs, U., Schmidt, H., Isenberg, T., Hancock, M. and Carpendale, S. BubbleType: Enabling Text Entry within a Walk-Up Tabletop Installation, University of Calgary, 2008.

29. Hodges, S., Izadi, S., Butler, S., Rrustemi, A. and Buxton, B. ThinSight: versatile multi-touch sensing for thin form-factor displays *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM, Newport, Rhode Island, USA, 2007.

30. Holm, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6, 2*. 65-70.1979.

31. HP TouchSmart. http://www.hp.com/united-states/campaigns/touchsmart/.

32. IBM SPSS Statistics. http://www.ibm.com/spss.

33. Digital Scribe. http://www.iogear.com/product/GPEN100C/.

34. Jacucci, G., Morrison, A., Richard, G., Kleimola, J., Peltonen, P., Parisi, L. and Laitinen, T. Worlds of information: designing for engagement at a public multi-touch display *Proceedings of the 28th international conference on Human factors in computing systems*, ACM, Atlanta, Georgia, USA, 2010.

35. Jeremy, R.C., Sidney, S.F., William, B. and Kenneth, C.S. Reactive environments. *Commun. ACM*, *40* (9). 65-73.1997.

36. Johnson, R. and Fryberger, D. Touch Actuable Data Input Panel Assembly, U.S. Patent, 1972.

37. Jorda, S., Geiger, G., Alonso, M. and Kaltenbrunner, M. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces *Proceedings of the 1st international conference on Tangible and embedded interaction*, ACM, Baton Rouge, Louisiana, 2007.

38. Kaltenbrunner, M., Bovermann, T., Bencina, R. and Costanza, E., TUIO - A Protocol for Table Based Tangible User Interfaces. in *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, (2005).

39. Ken, H., Randy, P., Dennis, P. and Neal, F.K. Two-handed virtual manipulation. *ACM Trans. Comput.-Hum. Interact.*, *5* (3). 260-302.1998.

40. Lumsden, J. and Gammell, A., Mobile Note Taking: Investigating the Efficacy of Mobile Text Entry. in, (Glasgow, Scotland, United Kingdom, 2004).

41. MacKenzie, I.S. and Chang, L. A performance comparison of two handwriting recognizers. *Interacting with Computers*, *11* (3). 283-297.1999.

42. MacKenzie, I.S., Nonnecke, B., McQueen, C., Riddersma, S. and Meltz, M. Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies*, *41*. 775 - 792.1994.

43. MacKenzie, I.S. and Soukoreff, R.W. A character-level error analysis technique for evaluating text entry methods *Proceedings of the second Nordic conference on Human-computer interaction*, ACM, Aarhus, Denmark, 2002.

44. MacKenzie, I.S. and Soukoreff, R.W., A Model of Two-Thumb Text Entry. in *Graphics Interface*, (2002), 117-124.

45. MacKenzie, I.S. and Soukoreff, R.W. Phrase sets for evaluating text entry techniques *CHI '03 extended abstracts on Human factors in computing systems*, ACM, Ft. Lauderdale, Florida, USA, 2003.

46. MacKenzie, I.S. and Soukoreff, R.W. Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction*, *17 (2 & 3)*. 147-198.2002.

47. MacKenzie, I.S. and Zhang, S.X. The design and evaluation of a high-performance soft keyboard *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, ACM, Pittsburgh, Pennsylvania, United States, 1999.

48. MacKenzie, I.S. and Zhang, S.X. An empirical investigation of the novice experience with soft keyboards. *Behaviour & Information Technology*, *20* (6). 411 - 418.2001.

49. MacKenzie, I.S. and Zhang, S.X., The immediate usability of Graffiti. in *Graphics Interface*, (1997), 129-137.

50. MacKenzie, I.S., Zhang, S.X. and Soukoreff, R.W. Text entry using soft keyboards. *Behaviour & Information Technology*, *18*. 235–244.1999.

51. Micire, M., Desai, M., Courtemanche, A., Tsui, K. and Yanco, H. Analysis of natural gestures for controlling robot teams on multi-touch tabletop surfaces *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM, Banff, Alberta, Canada, 2009.

52. Microsoft Surface. http://www.microsoft.com/surface/.

53. Rich Text Format Specification, version 1.6. http://msdn.microsoft.com/en-us/library/aa140277(v=office.10).aspx.

54. Moscovich, T. Principles and Applications of Multi-touch Interaction, 2007.

55. Motorola Xoom. http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Tablets/ci.MOTOROLA-XOOM-US-EN.overview.

56. Nintendo DS. http://www.nintendo.com/ds.

57. Norman, D.A. and Fisher, D. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors*, *24* (5). 509-519.1982.

58. Diffused Illumination (DI). http://wiki.nuigroup.com/Diffused_Illumination.

59. Laser Light Plane Illumination (LLP). http://wiki.nuigroup.com/Laser_Light_Plane_Illumination_(LLP).

60. The Open Sound Control 1.0 Specification. http://opensoundcontrol.org/spec-1_0.

61. Perceptive Pixel. http://www.perceptivepixel.com/.

62. Pierre, W. The DigitalDesk calculator: tangible manipulation on a desk top display *Proceedings of the 4th annual ACM symposium on User interface software and technology*, ACM, Hilton Head, South Carolina, United States, 1991.

63. Pierre, W. Interacting with paper on the DigitalDesk. *Commun. ACM*, *36* (7). 87-96.1993.

64. Rosenberg, I. and Perlin, K. The UnMousePad: an interpolating multi-touch force-sensing input pad *ACM SIGGRAPH 2009 papers*, ACM, New Orleans, Louisiana, 2009.

65. DSI - Diffused Surface Illumination. http://iad.projects.zhdk.ch/multitouch/.

66. Rubine, D., Specifying gestures by example. in *International Conference on Computer Graphics and Interactive Techniques*, (1991), 329-337.

67. Samsung Galaxy Tab. http://galaxytab.samsungmobile.com/.

68. Soukoreff, R.W. and MacKenzie, I.S. Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic *CHI '01 extended abstracts on Human factors in computing systems*, ACM, Seattle, Washington, 2001.

69. Soukoreff, R.W. and MacKenzie, I.S. Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, Ft. Lauderdale, Florida, USA, 2003.

70. Soukoreff, R.W. and MacKenzie, I.S. Recent developments in text-entry error rate measurement *CHI '04 extended abstracts on Human factors in computing systems*, ACM, Vienna, Austria, 2004.

71. Soukoreff, R.W. and MacKenzie, I.S. Theoretical upper and lower bounds on typing speeds using a stylus and keyboard. *Behaviour & Information Technology*, *14*. 370-379.1995.

72. Stanley, K.O. and Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, *10* (2). 99-127.2002.

73. T. Sato, H. Mamiya, T. Tokui, H. Koike and Fukuchi, K., PhotoelasticTouch: transparent rubbery interface using a LCD and photoelasticity. in *ACM SIGGRAPH 2009 Emerging Technologies*, (2009).

74. Mersive Technologies. http://www.mersive.com/.

75. Talks: Jeff Han: Unveiling the genius of multi-touch interface design. http://www.ted.com/index.php/talks/jeff_han_demos_his_breakthrough_touchscreen.html.

76. OGRE - Open Source 3D Graphics Engine. http://www.ogre3d.org/.

77. How Projected Capacitive Touch Technology Works. http://elotouch.com/Technologies/ProjectedCapacitive/howitworks.asp.

78. How Surface Capacitive Touch Technology Works. http://elotouch.com/Technologies/SurfaceCapacitive/howitworks.asp.

79. Van Dam, A. Post-WIMP user interfaces. *Communications of the ACM*, *40* (2). 63-67.1997.

80. The Bespoke 3DUI XNA Framework. http://www.bespokesoftware.org/3dui/.

81. The Bespoke Open Sound Control Library. http://www.bespokesoftware.org/osc/.

82. Varcholik, P., LaViola, J., Nicholson, D. TACTUS: A Hardware and Software Testbed for Research in Multi-Touch Interaction *Human-Computer Interaction International (HCI International)*, San Diego, CA, 2009.

83. Cintiq. http://www.wacom.com/cintiq/cintiq-21ux.php.

84. Wacom. http://www.wacom.com/.

85. Ward, D., Blackwell, A. and MacKay, D. A data entry interface using continuous gestures and language models *Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM, San Diego, California, United States, 2000.

86. Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-Touch Surface. http://www.ece.udel.edu/~westerma/main.pdf.

87. Wigdor, D., Perm, G., Ryall, K., Esenther, A. and Chia, S., Living with a Tabletop: Analysis and Observations of Long Term Office Use of a Multi-Touch Table. in *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, (2007), 60-67.

88. AdaBoost. http://en.wikipedia.org/wiki/AdaBoost.

89. IBM Simon. http://en.wikipedia.org/wiki/IBM_Simon.

90. Newton. http://en.wikipedia.org/wiki/Apple_Newton.

91. Tablet PC. http://en.wikipedia.org/wiki/Tablet_pc.

92. Wilson, A. TouchLight: an imaging touch screen and display for gesture-based interaction *Proceedings of the 6th international conference on Multimodal interfaces*, ACM, State College, PA, USA, 2004.

93. Wobbrock, J. and Myers, B. Analyzing the input stream for character- level errors in unconstrained text entry evaluations. *ACM Trans. Comput.-Hum. Interact.*, *13* (4). 458-489.2006.

94. Yamada, H. A historical study of typewriters and typing methods: From the position of planning Japanese parallels. *Journal of Information Processing*, *2*. 175-202.1980.

95. Yuksel, B., Donnerer, M., Tompkin, J. and Steed, A. A novel brain-computer interface using a multi-touch surface *Proceedings of the 28th international conference on Human factors in computing systems*, ACM, Atlanta, Georgia, USA, 2010.