ABSTRACT

Title of Document: INTERFACE AND INTERACTION DESIGN FOR ONE-HANDED MOBILE COMPUTING.

Amy Kathleen Karlson, Ph.D., 2007

Directed By: Associate Professor Benjamin B. Bederson, Department of Computer Science

Mobile phones are not only a ubiquitous social accessory, but rapid technology advances have transformed them into feature-rich, Internet-enabled mobile PCs—a role once reserved for touchscreen-based personal digital assistants (PDAs). Although the most widespread phone styles in circulation feature the classic combination of numeric keypad and non-touchscreen display, larger touchscreen devices are gaining ground, as indicated by the fervor surrounding new devices such as Apple's iPhone and LG's Prada phone. Yet as devices evolve, users will remain constrained by the limits of their own visual, physical, and mental resources. My research has focused on the specific limitation that mobile users often have only one hand available to operate a device, which can be especially problematic for touchscreen-based devices, since they are frequently designed for two-handed stylus operation. Considering the growing volumes of data that small devices can now store and connect to, as well as the expanding cultural role of mobile phones, improving usability in mobile computing has potentially enormous implications for user productivity, satisfaction and even safety.

My own exploratory surveys have suggested that one-handed use of mobile devices is very common but that today's hardware and software designs do not support users in performing many tasks with only one hand. Motivated by these findings, the research goal of this dissertation is to contribute substantial knowledge in the form of empirically backed design guidelines and interaction techniques for improving one-handed usability and operation of mobile devices, with particular emphasis on those with touch-sensitive displays. The guidelines for one-handed mobile device design are the product of a series of studies conducted in pursuit of foundational knowledge in user behavior, preference, thumb capabilities and touchscreen-thumb interaction characteristics for single-handed device use. I also demonstrate the application of these guidelines through the development and evaluation of four applications. Two involve designs for navigating among programs, one provides an interface for searching large data sets, and the last offers a generalized mechanism for controlling arbitrary touchscreen interfaces with a thumb. Each of these applications explores a different one-handed interaction technique and offers perspective on its viability for one-handed device use.

INTERFACE AND INTERACTION DESIGN FOR ONE-HANDED MOBILE COMPUTING.

By

Amy Kathleen Karlson

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Associate Professor Benjamin B. Bederson, Chair
Assistant Professor François Guimbretière
Associate Professor José Contreras-Vidal
Professor Ashok Agrawala
Professor Adam Porter

Dedication

To My Mother and Father

Acknowledgements

I would like to extend my utmost thanks to my advisor, Dr. Ben Bederson, without whose guidance, patience, frankness and encouragement this thesis would not have been possible. I am also very grateful to my committee members, Dr. François Guimbretière, Dr. Ashok Agrawala, Dr. Adam Porter, and Dr. José Contreras-Vidal, for donating their time and expertise in making this document the best possible representation of my work.

I would also like to thank the entire staff and members of the HCIL Lab, who have been a joy to work with over the years. In particular, Aaron Clamage, Anne Rose, and Kiki Schneider were always friendly, responsive, and expert in their technical and administrative support. A special thanks to Ben Shneiderman and Catherine Plaisant for their leadership and Bongshin Lee, Adam Perer, Jerry Fails and Georg Apitz for their friendship.

I would also like to thank Microsoft Research, which funded the entirety of this thesis research, as well as their Visualization and Interaction Research (VIBE) group, who hosted me for two exceptionally educational and resourceful summer internships. George Robertson and Mary Czerwinski provided unparalleled support and guidance, and together with Greg Smith, Bryan Meyers, Dan Robbins, and Desney Tan, made enormous intellectual, technical, and social contributions to my summer research experience. In association, Patrick Baudisch and Dan Vogel's work on finger-based touchscreen interaction has had substantial inspirational and motivational impact on my own research.

Finally, a host of family and friends have had unfailing faith in me over the years, and to whom I am deeply indebted: David Karlson, Sarah Karlson, Patience Landry,

Richard Landry, David Levin, Edward Manuel, James Pasmantier, Emily Porter, David Richards, Kathleen Richards, Lisa Robey, Karen Salvini, Megan Spitz, Pamela Spitz, Chris Stone, Catherine Thompson, Jacqueline Wahl, Kendra Walther, and Jordan Wilkerson.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 The Rise of the Mobile Phone as Personal Computer

It may come as no surprise to hear that personal computing today is far different than it was just 10 years ago. We are, after all, quite accustomed to the relentless forward march of technology. But staring at our desktop computers, we might reflect, "*Really*? How different *is* this?" Sure, we are more likely to have flat-screen monitors than space-hogging CRTs, our operating systems have gotten a bit of a makeover, and our hard-drives are zippier, but are *we* more efficient? More productive? More empowered? Less annoyed? Given that so many people regularly complain about the distractions, annoyances and general information overload associated with computers, it is not clear how many people would feel this way.

To better understand where the changes have occurred and how they have impacted users, we must reconsider the outdated association of "personal computing" with a "desktop PC". In fact, the most dramatic personalization of computing has occurred *beyond* the desktop, most notably as a result of users' widespread access to the Internet. Let's take a look at a few examples. Web search engines began as powerful tools for finding and revisiting information among the tens of billions of web pages in the networked world, yet the same technology has transformed how we find and (no longer need to) manage our personal data sets as well. Web portals such as Facebook (www.facebook.com), YouTube (www.youtube.com), and Flickr (www.flickr.com), among many others, have flattened the entry cost for crafting a personal web presence

and participating in online communities, requiring only a web browser and an Internet connection. The web-based, community-built, editable encyclopedia Wikipedia (www.wikipedia.org) provides a free, convenient alternative to published encyclopedic works, which must be purchased as DVDs or digital subscriptions, or otherwise involves a trip to the local library. Web-based map utilities generate flexible, high-quality driving directions from a personalized start point, rather than from generic, well-known landmarks. The list is endless. And though most users will perform tasks like these at their desks, we owe the breadth of services and achievements in improved user control, efficiency, effectiveness, and satisfaction to the lifeline we call the Internet.

The significance of the Internet's ubiquity is not only about the data and services it offers, but also about the utility of continual connectivity—to work, to friends, to information, to entertainment, and so on. No place is the worldwide obsession with connectivity better illustrated than the explosion of the mobile phone market. With mobile phone sales growing by more than 20% per year since 2002 [99, 101, 156, 157], industry analysts are anticipating over 1 billion units will be sold in this year (2007) alone [100]. Significantly, the prediction for mobile phones sales is over four times the volume expected for personal computers [134]. After taking into account multiple subscriptions per user, 2.3 billion people, over 30% of the world's population, has a mobile phone today [69], which is three times the number that use PCs worldwide [133].

Once used solely for placing calls, today's mobile phones are increasingly equipped for text messaging, emailing, browsing the Internet, playing music and videos, taking pictures, viewing and even editing documents. Usage statistics confirm that phones are being used for tasks once reserved for desktop computing: more than twice as

many people are active users of the short message service (SMS) to send mobile text messages as are active users of desktop email, and 750 million people access the internet via mobile phone today—just under the 850 million who do so via PC [1]. Furthermore, mobile devices are more personal than ever—they are small enough to be carried or within reach at all times, they are typically not shared among friends, family or spouses, they are replaced on average every 18 months which allows users to keep them in sync with evolving technologies, personal needs and style, and their vast software offerings allow users to customize ring tones, background themes, and applications. Connected, capable, and personalized, mobile phones are ushering in a new era of personal computing, and with it, enormous opportunities and challenges in human-centered design.

1.2 Challenges in Mobile Computing

The diminutive size of the mobile phone has been central to its success: being small enough to fit in a pocket or unobtrusively worn increases the probability that users will be collocated with their phones, and, in turn, perpetually connected. This benefit has driven the steady trend toward smaller, slimmer forms. At the same time, technology advances have allowed devices to offer more hardware capabilities (e.g., larger screens, music play, camera, Bluetooth, and WiFi) and improved (e.g., faster, graphically richer, and more powerful) software experiences.

Unfortunately, the more our phones do, and the more data they store and access, the more we feel the pinch that their input and output constraints have on usability. Smaller screens mean less information can be displayed at once, so designers are faced with challenges in presenting and structuring information; users in turn bear the burden of

performing more navigation steps, which can increase overall interaction time, as well as mental and visual demands. Meanwhile, reducing the space designated for buttons leads to several possible design tradeoffs: 1) buttons are removed and the ones that remain take on multiple roles, which increases users' mental or visual demands; 2) buttons are shrunken and/or packed tighter, making them harder to hit reliably with thumbs and fingers; 3) buttons are hidden under a screen or clamshell design, so extra time must be taken to reveal them before use; or 4) physical buttons are replaced by software-based touchscreen buttons, which increases visual demand due to the absence of tactile feedback.

In addition to the design issues that arise from the physical limitations of small screens and keyboards, external factors that can further degrade usability abound in mobile computing. Not only do mobile scenarios vary widely with respect to lighting, noise levels, stability, privacy and social context, but users "on the move' must divvy their mental, visual, and physical resources between device tasks and the surrounding environment. The negative impact that mobility can have on attention [111, 142], task performance [10, 95, 141, 146], visual performance [107], mental demand [24, 146], motor skills [37, 105], and hand availability [79] during device use have been studied and verified widely. The significant implications these results have on device usability, enjoyment and user safety have perpetuated a de facto guideline in mobile interface design—to strive for "minimal attention" user interfaces [113]. My own research has focused on users' finite *physical* resources and, in particular, scenarios in which users have only a single hand available to operate a touchscreen-based device.

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

Figure 1. Mobile use scenarios. Mobile scenarios can leave only a single hand available for device use (a,b); mobile phones can often be operated successfully with one hand (c); stylus oriented touchscreen devices often require two hands to operate (d).

1.3 One-Handed Use of Mobile Touchscreen Devices

One characteristic of mobile device use is that people on the move often need a hand to carry additional belongings, open doors, hold handrails, or otherwise manage a dynamic environment (Figure 1a,b). Interfaces that accommodate single-handed device operation can thus offer an important benefit to users by freeing a hand for the variety of physically demanding and/or high-attention tasks common to mobile scenarios [79]. Traditional-style mobile phones that have read-only displays and numeric keypads are generally lightweight and easy to manage with one hand (Figure 1c) but can of course be unfriendly to thumbs if buttons are small or keypads are crowded. Devices with touch-sensitive screens present more serious and consistent challenges for one-handed operation. First, touchscreen displays tend to be larger than read-only displays and so touchscreen devices themselves tend to be larger, heavier, and more difficult to manage with one hand than non-touchscreen devices. Furthermore, because most touchscreen software is designed to be used with a pen-like stylus (Figure 1d), interaction objects can be too small to hit reliably with a thumb, or distributed in such a way as to be too award to hit with a given hand grip (e.g., too far).

The relatively large forms and stylus-based software of today's touchscreen devices make it clear that one-handed operation has not traditionally been considered a high-priority use scenario for this class of computer. Whether this is the result of oversight or deliberate design choices, my exploratory investigation into the relevance of one-handed mobile device use in Chapter 2 concludes that touchscreen devices, which typically serve to support rich, high-throughput, focused interaction, are not considered sufficiently enabled for one-handed mobile operation. This finding has formed the basis of my research position that improving support for one-handed touchscreen operation will extend the overall utility and usability of a significant family of mobile devices.

But what is the practical impact of this work? The popularity of small, sleek, affordable devices has kept the larger, pricier touchscreen devices languishing at the back of the pack with respect to general consumer sales. Yet because touchscreen devices support PC-like applications and interactions, they have enjoyed high penetration in professional communities. The expectation that touchscreen-based devices will continue to offer advantages over non-touchscreen devices in professional markets means there is indeed an important user population that will be well served by improving one-handed operation of mobile touchscreen devices. But a recent confluence of events may prove this thesis research to be more broadly relevant than anticipated. The explosion of web-accessible visual media such as photos, television, videos, and movies, has elevated the importance of pixels for the general consumer. At one end of the design spectrum, pixels can be favored over even hardware buttons, whose functions can be migrated to pixels in the form of touchscreen soft keys. This is the design approach that has given birth to Apple's iPhone sensation—the most successful touchscreen device to be marketed to the

non-professional consumer. Based on the substantial line-up of iPhone competitors, it appears the tide may have indeed turned for mobile touchscreen technology, which analysts at IMS Research (www.imsresearch.com) predict could be included in nearly 30% of the devices on the market by 2011. Even if these predictions are overly optimistic, they certainly bode of growth in the touchscreen device sector, and thus the potential for this research to have farther-reaching impact than would be possible today.

1.4 Research Strategy

To date there has been relatively little integrated study of the tasks, technologies, interaction techniques and interface designs that together contribute to the experience of mobile device mastery with one hand. Instead efforts have generally addressed only a single aspect of one-handed device operation. Technology-oriented research has investigated new input channels, such as the spatial position or orientation of the device [45, 64, 120], or by interpreting simple touchscreen-based finger gestures [25, 76], but both tend to limit users to coarse-grained navigation tasks or finite command sets. Other research has focused on supporting a specific task, such as media control [5, 115], text entry [151], or data entry for fieldwork [113]. But in the varied landscape of mobile devices and applications, one-handed solutions must extend to a wide variety of forms, features, functions, and of course, users.

This thesis does not pursue a single innovation that achieves this broad ideal, but I have instead used its goals as a guide. For example, I have investigated how different device *forms* affect thumb mobility, how different hardware *features* compare for manipulating an input cursor, and how the full range of standard touchscreen software *functions* might be performed with the help of a *user*-customized software widget. But

more formally, my research has approached the problem of one-handed device use from two perspectives: 1) *foundations* - motivating the relevance of one-handed mobile device use and investigating the fundamental characteristics and limitations of single-thumb interaction with handheld devices; and 2) *applications* - using this knowledge toward the development of touchscreen-based software interfaces and interaction techniques to support mobile input, access and management of data with a single hand.

To establish *foundations*, I have conducted a series of studies to develop an understanding of basic human factors involved in single-handed device operation: 1) a field study to understand the extent to which single-handed use is currently showing up "in the wild"; 2) a survey to record personal accounts of current and preferred device usage patterns; 3) an empirical evaluation of thumb tap speed to understand how device size, target location, and movement direction influence task performance; 4) a quantitative evaluation of the learnabilty and executability of touchscreen-based gestures; and 5) an empirical study to determine the required sizes for thumb-oriented touchscreen targets. This thesis is motivated by the results of the first two *foundation* studies, which offer evidence that one-handed use of mobile devices is common, but that one-handed interaction would be more prevalent if hardware and software designs better supported thumb interaction, especially for touchscreen-based devices. The remainder of the *foundation* studies provided the data to back design guidelines for thumb-oriented touchscreen interfaces.

Together the results of these first studies offer foundational knowledge in user behavior, preference, thumb motor capabilities, and touchscreen-finger interaction characteristics for one-handed mobile device use. But with mobile devices storing and

connecting to increasing volumes of data, techniques for effectively managing, navigating and searching local and remote data sets are essential to their continued utility. So to understand how these *foundations* apply in practice to real interfaces, I developed and studied four *applications* that explore design trade-offs in supporting a range of typical mobile information tasks with one hand.

The first two applications, AppLens and LaunchTile, embody competing approaches for using one thumb to navigate among a set of programs on a touchscreen-based device; the AppLens design uses an indirect gesture language to allow users to access small or distant objects with an object cursor, whereas the LaunchTile interface ensures interaction targets are always large enough and close enough to be hit or moved directly with the thumb. A formative study of the two interfaces investigated usability issues with the alternative design strategies as well as captured users' comparative preferences.

For the next application, I focused on the well-known issue that text entry is slow on mobile devices, and explored how interfaces might be designed to increase user efficiency for text-oriented activities. As an intern at Microsoft Research, I designed and built the FaThumb interface to understand the potential advantages of navigating and selecting data attributes as an alternative to keyword-based text entry for searching large data sets from a mobile device. A formal study of FaThumb confirmed that facet-based attribute navigation can outperform ad-hoc keyword search for exploratory browsing activities, and that users strongly favored dynamic results filtering via facet navigation over text entry for these tasks.

While the first three applications address one-handed design approaches for supporting high-level information navigation and search tasks, a great many device activities are instead accomplished through a series of low-level object selections. For example, browsing the Internet can be considered a sequence of link selections, and setting an alarm involves a series of selections and edits to hour, minute, and am/pm. It is unsurprising then that respondents to my web survey indicated they want better single-handed support across *all* touchscreen tasks, which are invariably include low-level object selection sub-tasks best performed with a stylus. The question then becomes whether we redesign all applications for thumb accessibility, or whether we somehow enable thumb use of stylus oriented interfaces. Both personal experience and user observation suggest that many factors determine the number of hands people use for device interaction, including hand availability, preference, and desired throughput. Because these factors may change dynamically in mobile environments, effective touchscreen interfaces will need to balance effective presentation and interaction efficiency independent of hand availability. Not only may some of these choices be unfavorable to thumb input, but there already exists a vast body of legacy touchscreen applications that favor stylus input.

To address the challenge of ensuring touchscreen devices are operable with the thumb without requiring designers to sacrifice the expressiveness of the information presentation, or the efficiency of input when two hands are available, I designed and developed a fourth and final application called ThumbSpace, a generalized method for supporting one-handed use of arbitrary touchscreen interface layouts. ThumbSpace is a personalized interaction widget that users launch on-demand to aid in selecting a screen

object that is too small or too far to hit accurately or comfortably with the thumb. Through design refinements and formal comparative studies, I have demonstrated that ThumbSpace is a well-liked means for improving user accuracy in hitting small, out of reach targets on touchscreen devices, and that the benefit ThumbSpace provides outweighs the time or decision costs associated with on-demand use.

1.5 Thesis Contributions

1.5.1 Foundations: Guidelines for One-Handed Mobile Device Interaction

Contribution (C1): Motivated the need for single-handed interface design based on the reporting of current usage patterns and preferences for one- and two-handed mobile device interaction.

Results from my field observations of travelers using mobile devices indicate that one-handed use of cell phones is common. User activity seems to correlate with the number of hands used for device operation, but based on observation alone, the direction of influence remains unknown.

Results from my survey of device use, behavior and preference also indicate that single-handed phone use is common. The different usage patterns for different types of devices indicate that device input capabilities influence the number of hands used to perform an activity. Overall, the majority of users stated they would prefer to use one hand to perform all activities, suggesting attention should be paid to improving one-handed device use across all activities for all devices.

Contribution (C2): Provided researchers and practitioners with practical, empirically-backed guidelines for one-handed mobile interface designs.

Results from my study of thumb movement across different sized devices offer researchers and practitioners guidelines for developing mobile interfaces that support one-handed interaction: 1) support localized interaction (e.g., by placing targets close to one another) to minimize grip adjustment; 2) allow users to configure tasks for either left or right-handed operation especially when the tasks involve diagonal movements; 3) strive to place interaction targets toward the vertical center of the device to support a stable, balanced grip, as well as toward the middle of the device (horizontal center) to make targets easy to reach for both left and right handed users; and 4) favor small devices in order to support overall device control and satisfaction.

Results from my study of thumb-appropriate touchscreen target sizes indicate that a target size of about 10 mm^2 should be sufficiently large to support fast, low-error target selection.

Finally, user performance with applications that integrate thumb gestures suggest that gestures are best used to complement rather than replace tapping interaction.

Together with the supporting quantitative data, these guidelines can provide designers the resources necessary to make informed decisions about interface target placement and the resulting impact on one-handed performance. Interfaces created with these guidelines in mind have the potential to produce comfortable, usable, and ergonomic one-handed interaction experiences.

1.5.2 Applications: Interaction Techniques for One-Handed Data Access

Contribution (C3): Inspired novel application management methods for small screens based on fluid context switching, rich adaptive application representations, one-handed access, and device independence.

The AppLens and LaunchTile designs offer unique solutions for navigating among programs on a touchscreen-based device with a single hand. User feedback for both designs was very positive, especially the feature for displaying of high-value summary detail for multiple applications at once. This work also introduced a novel approach to small device software design—the ability to dynamically adapt interfaces to devices with varying screen sizes and aspect ratios.

Contribution (C4): Offered a one-handed approach to searching large volumes of data from a mobile phone without the need for text entry.

The FaThumb interface presents a unique solution for enabling rapid, fluid access to large data sets from a mobile phone, and holds great potential for adaptation to touchscreen-based devices. Despite anticipated advances in text entry methods for small devices, single handed designs are not likely to rival the two-handed touch typing speeds of desktop computing. Thus designs for handheld devices which deemphasize text entry while still providing powerful, rapid access to remote and local data stores, will support the continued efficacy of personal mobile computers.

Contribution (C5): *Offered a generalized one-handed approach for interacting with touchscreen applications designed for rich presentation and stylus-based interaction.*

ThumbSpace is a personalized interaction technique that adapts rich, stylus-oriented touchscreen interfaces to the limited motion range and noisy input of the thumb. By offering ThumbSpace as an input option, touchscreen interface designers will be free to focus on layouts that provide the most effective presentation or efficient navigation of information irrespective of the number of hands users have available at the time of use. In addition, the ThumbSpace input approach has the potential to make the large body of stylus-based touchscreen designs in circulation today forward-compatible with one-handed thumb use.

1.6 Thesis Organization

Chapter 2 of this thesis serves as the detailed motivation for my research on interface designs and interaction techniques to accommodate one-handed use of mobile touchscreen devices by reviewing relevant prior work and presenting two exploratory studies: 1) a field study conducted to understand the extent to which single-handed use is currently showing up "in the wild"; and 2) a survey that captures personal accounts of actual and preferred device usage patterns. Chapter 3 completes the *foundations* of my work in presenting three studies that explore various human factors of high relevance to one-handed use of touchscreen-based mobile devices: 1) an empirical evaluation of thumb movement capabilities to understand how device size, target location, and movement direction influence task performance; 2) an empirical study to determine the required size for thumb-oriented touchscreen targets; and 3) a quantitative evaluation of

the learnabilty and executability of touchscreen-based gestures. Chapters 4 through 6 present *applications* of my *foundational* findings for supporting users in performing real-world mobile data access tasks with one hand. Chapter 4 introduces AppLens and LaunchTile, which embody two alternative design strategies for developing one-handed touchscreen interfaces. In Chapter 5 I present the FaThumb interface, and explore whether searching and browsing large data sets using attribute navigation can be a competitive alternative to traditional text-based keyword entry. Chapter 6 presents ThumbSpace, a personalized widget for interacting with stylus-based touchscreen interfaces with the thumb. Finally, this dissertation concludes with lessons learned and future work in Chapter 7.

Due to the characteristic that this thesis approaches the problem domain of one-handed thumb-based touchscreen interaction from the complementary, but differing perspectives of *foundations* and *applications*, a broad range of prior research has proven relevant to my work. Thus instead of devoting a single chapter to a literature review, I have chosen to contextualize the prior work by including a section within each chapter to discuss the associated related research.

Chapter 2
Foundations: Why Design for One-Handed Mobile Devices?

Motivated by the rationale Pascoe et al. offered for developing minimal attention user interfaces (MAUIs) [113] for field scientists, it seemed probable that mobile device users in the general population would benefit from one-handed designs. Furthermore, my review of past and ongoing research concluded that little attention had yet been paid to touchscreen-based devices. In order to establish that a research agenda based on improving one-handed support for touchscreen-based mobile devices had high impact potential, I performed two exploratory studies. I first ran a field study to capture the extent to which single-handed device use is currently showing up "in the wild". Second, I polled users directly to record personal accounts of current and preferred usage patterns for a variety of mobile tasks across a range of device types.

This chapter begins by presenting the research related specifically to the motivations for supporting one-handed mobile device use, and the solution spaces that others have explored to date. Following the review, I present the details of the preliminary studies that serve as the groundwork for the remainder of this thesis.

2.1 Related Work

2.1.1 Effects of Device Size on Design

The input and output constraints of mobile devices, discuss previously in Section 1.2, have necessitated innovations in information and interaction design for providing users effective means of viewing, navigating, and inputting data on small devices. One driver

has been the increasing adoption of web-enabled handheld devices; although over 40% of Americans who have a mobile device access the Web through their device [3], the majority of web content targets standard desktop-sized displays. Restructuring [32], reformatting [124, 148], summarizing [29] and zooming [14] web content are just some of the techniques that have been used to adapt web pages for small-screen browsers like the iPhone's Safari (www.apple.com/safari/) and Microsoft's DeepFish (www.labs.live.com/deepfish/). Many of these techniques are also appropriate for the broader body of data that is synchronized and accessed across stationary and mobile platforms. Of course, remedies such as zooming [27, 56, 81], focus+context displays [16, 56] and visual cues [13] that have long been used for managing large data sets within the limits of desktop displays have proven useful for small screens as well.

With respect to input constraints, one of the strongest motivators of change has been the contagious popularity of SMS text messaging. Largely lead by the teen market [84], 39% of Americans now use text messaging to communicate using their phones [104]. Yet unlike sending email from a desktop's Qwerty keyboard, the majority of text messages are tapped out on the 12 keys of a traditional phone keypad (Figure 2a). Several approaches have been developed over the years to reduce the multiple key presses traditionally required to disambiguate among letters appearing on the same key, including word prediction [91, 144], new letter-to-key mappings [52, 109], chording techniques [150] and joystick gestures [153]. Despite the significant performance improvements that many of these techniques have achieved over multi-tap for 12-key layouts, the fact that their input speeds still lag far behind those achieved at the desktop has allowed device

designs that include a full Qwerty thumbboard (e.g., Figure 2b,c) to remain competitive despite being larger and heavier than the 12-key models.

(a) (b) (c)

Figure 2. Examples of mobile device keyboards. The assignment of letters to buttons for a standard phone keypad (a). A cell phone model with a slide-out Qwerty thumbboard (b). A phone model with an integrated thumbboard on the front of the device (c).

2.1.2 Attention and Mobility

Developing effective ways to view and interact with information given the small form factors of handheld devices is only one of the challenges in designing for mobile computing. Another challenge is that as mobile devices become increasingly integrated into our daily lives, they are now commonly used in highly dynamic environments, such as while walking in crowds, shopping, riding public transportation, driving, and so on. In such scenarios, users have no choice but to divide their mental, visual, and physical attention between the device task and the surrounding environment. Indeed, many researchers have studied and verified the negative impact that mobility has attention [111, 142], mental performance [10, 95, 141, 146], mental demand [24, 146], visual performance [107], and motor skills [37, 105]. Perhaps the magnitude of the drain that mobile device use can have on users' faculties is most succinctly captured by the study which demonstrated that talking on the phone, even hands free, while driving is akin to driving while legally intoxicated [141]. The resounding message to the public is that multitasking with a phone can be dangerous. The message to interaction designers is that

enormous benefits in usability and safety may be reaped by minimizing the mental, visual, and physical demands of mobile computing.

Pascoe, Ryan and Morse capture the spirit of this approach with their Minimal Attention User Interface (MAUI) framework [113], developed in response to the physical and visual challenges field scientists face while gathering data using PDAs. Their framework considers both activity and interaction-mode characteristics to guide appropriate design choices for the visual, audio and tactile interaction channels. For example, a MAUI to support an animal counting activity would provide hardware buttons for incrementing and decrementing the tally, which allows the scientist to dedicate her visual attention to the subject matter. Kristoffersen and Ljungberg [79] observed that similar challenges are faced in other mobile professions which then cause users to carve out dedicated time and/or space, or "make place", to complete work tasks with a handheld computer. Like Pascoe et al., they emphasize eyes free input and output (audio) when viable.

A limitation shared by these early investigations, however, is that they focused on specialized users and tasks. Because of these assumptions, the interfaces could be highly scripted and simplified so that only a small number of actions were available to users at any time. More commonly, devices and interfaces need to support a wide variety of users, tasks and scenarios. So while the principles of minimal attention user interfaces provide useful guidance for mobile design, how they are applied in practice for consumer devices remains an active area of research.

2.1.3 Impact of Form on Physical Resource Demands

Given the variety of tasks for which mobile computers are used and the variability of use environments, it is inevitable that no single MAUI design or philosophy will be a panacea for reducing the concurrent mental, physical, and visual demands of computing while mobile. In my own work, I have focused on the problem that mobile users often have only one hand available to use a device—a limitation of their *physical* resources. Furthermore, I have paid particular attention to touch-sensitive devices, since they have traditionally been designed for two-handed stylus interaction. I will position my work within the field by reviewing other research efforts that have aimed to reduce the physical demands of mobile computing.

It would be remiss to discuss solutions to the physical demands of mobile device interaction without a discussion of the varying affordances that different hardware features offer. Other than the minimum functional guarantee of placing and receiving calls, little else is standardized among the hardware and software features of mobile devices. In fact, consumer demand for new, feature-rich, technically sophisticated, and stylish devices is a powerful diversifying force in the mobile market, and the software-hardware design tradeoffs represented by each device model conjure a unique point in usability space. However, to focus the discussion, I will simplify the design space into three device styles based on the main input channel (12-key numeric keypad, Qwerty keyboard, and touchscreen), which I will review briefly, along with the physical demands necessitated by their forms.

12-Key Numeric Keypad Devices

The original mobile phones were rectangular "candy-bar" shaped devices, which had a display-only screen for output and a standard 12-key numeric keypad for input. The majority of cell phones today retain this 12-key design, and with good reason. The limited number of raised physical keys allows expert users to dial and SMS text message eyes-free, which is helpful for allowing users to keep their visual focus on a dynamically changing environment, as well as for allowing discreet (e.g., under the table or in the pocket) communication. Because the 12 keys are localized, they can all be reached with the thumb using a single grip, which typically allows users to operate the phone easily with one hand. However, keys that are too small or too close to one another can threaten user precision when using the thumb, which can then cause users to resort to two-handed index-finger operation. Alternately, tasks requiring high throughput, such as text entry, may encourage users to increase input parallelism with two-handed thumb operation.

Qwerty Keyboard Devices

Devices that give high priority to text entry tasks like email or SMS text messaging offer users a Qwerty keyboard, which is what distinguished the RIM Blackberry device (Figure 3a, www.rim.com) from the field when it entered the market in 1999. Today a wide variety of other models exist, including those both with and without touchscreens, as well as convertible models whose keyboards flip or slide into view (e.g., Figure 3b). Mobile Qwerty keyboards are typically used with two hands for several reasons. First, most users are accustomed to using two hands to type on full-sized desktop keyboards, so it is natural for users to transfer this interaction style to miniature keyboards, even though the speed advantage of ten-finger touch typing is largely lost when users move to only two

thumbs. In fact, some Blackberry models angle buttons differently on the left versus right halves of the device to better accommodate the different access angles of the left and right thumbs (Figure 3a), further reinforcing the two-thumb entry style. Second, the device keyboard must be wide enough to fit at least 10 buttons, which can lead to devices that are simply too wide for users to reach all buttons with the thumb when held in one hand (Figure 3b). Ultimately, even those Qwerty devices that are narrow enough to hold in one hand (Figure 3c) may often be used with two because of the speed advantage that parallel thumb entry offers to users.

Dedicated hand resources are not the only physical demand of Qwerty text entry. Significant visual attention is also required to ensure users hit the correct keys because there are simply too many buttons for most users to rely solely on tactile cues. And finally, message composition is mentally engaging. Thus, overall, text entry is a demanding mobile activity in terms of physical, visual, and mental resources, especially while using a mini Qwerty keyboard.

(a) **(b)** **(c)**

Figure 3. Examples of phone models with Qwerty keyboards. A Blackberry being used in the traditional manner with two hands (a). A slide-out device model shown in one hand (b). A Palm Treo shown in one hand (c).

Touchscreen Devices

It is not uncommon for various phone models to be described as "converged" devices, which refers to phones adopting capabilities that have traditionally been offered by a

dedicated device, such as a camera, a music player, a GPS, etc. Yet converged devices also encompass those that merge the capabilities of a personal digital assistant (PDA) and a phone. PDAs were the first generation of handheld, mobile computer which synchronized data with a desktop PC, and aimed to offer PC-like capabilities on-the-go, such as a calendar, calculator, address book, and notepad. Because of these design goals, screen space and input flexibility were a priority, and so PDAs tended to feature large, touch-sensitive displays, and few physical buttons. Mobile phones, on the other hand, have traditionally favored non-touchscreen displays and 12-button input. Interestingly, as the capabilities of phones and PDAs converge, the phone industry is experiencing a jump in the popularity of touchscreen-based devices. This trend has primarily been attributed to the June 2007 release of Apple's iPhone, which features only a single physical button and a large, high-tech, multi-touch display.

To optimize the use of screen space for information display, most touchscreen software designs feature small interaction targets. These targets are easily hit using the included pen-like stylus, but can be too small for reliable finger actuation; capacitive touchscreens, which are in broadest use today, accept only one input point generated from the average contact area of the finger, and this point can vary dynamically and unpredictably with changes in finger pressure and input angle. Because using a stylus requires two hands, most touchscreen designs are not well suited to one-handed interaction. The two-handed use model for touchscreens is reinforced by the fact that touchscreen displays are often larger than a user's thumb range given a particular grip, resulting in targets that can be too *far* to hit when the device is held in only one hand. Interestingly, Apple's iPhone is designed for finger-only use, but not necessarily one-

handed use, since interaction objects can be found across all regions of its relatively large touchscreen display. In practice, however, the slim, lightweight form of the iPhone makes it quite manageable with one hand.

2.1.4 Strategies for Reducing Hand Requirements

Since hands are an important user resource, a task that requires the dedicated use of two hands typically consumes significant attention resources due to associated visual (e.g, looking at a thumbboard or touchscreen) or mental (e.g., text composition) demands. Thus, minimizing the number of hands required to do a task is consistent with MAUI guidelines, which aim to minimize overall user distraction [113]. Approaches for reducing hand requirements for tasks differ according to the device hardware. At one extreme, head mounted displays require no hands, but suffer potential drawbacks of visual [82] and social interference [94], not to mention that such equipment is not yet found in common use. Because of these types of competing issues that arise when attempting to eliminate all hand use from mobile computing, efforts have more commonly focused on reducing hand requirements from two hands to one. Since 12-key phones are already quite easy to use with one hand, touchscreen-based devices have received the majority of attention in adapting input for one hand.

Specialized Hardware

Of course, many touchscreen devices include some type of indirect input hardware such as a directional pad or thumbwheel that accommodates one-handed thumb use, but the degree to which the device is fully operable with the hardware alone depends on whether the operating system and applications have been designed with indirect input in mind. The RIM Blackberry devices (www.blackberry.com), for example, are famous first for

their thumb wheels, and more recently trackballs, which accommodate powerful navigation, selection, and smart context menu capabilities throughout the entire interface experience. But the Blackberry's elegant, system-wide thumbwheel navigation capability came about because the models have never had touch-sensitive displays. In contrast, when the touchscreen-based Windows Mobile Pocket PC was retrofitted with a scroll wheel in the Cingular 8525 (a.k.a. HTC TyTN), the result was less successful because a subset of interface elements remained actionable only through direct touch, causing interruptions in the flow of interaction.

Accelerometer-equipped devices and the use of tilt orientation for input has been well-studied as a means for accommodating one-handed use of devices, such as for text entry [127, 151], scrolling [11, 45, 64, 120], zooming [49], menu navigation [120] and photo browsing [11, 33]. Yet because tilt is achieved through gross motor movement in the hand and wrist, tilt may be inappropriate for fine-grained input control or parameter manipulation [119]. Although most mobile devices on the market do not yet incorporate accelerometer technology, the iPhone uses a rotation sensor to automatically adapt the orientation of the screen contents to the user's view perspective.

Others have developed specialized hardware solutions for one-handed control of specific mobile tasks, such as the Twiddler for one-handed chording text entry [87], the iPod for music play [5], and the EarPod for menu navigation [161]. My work has focused on standard touchscreen-based mobile devices, and resolving problems associated with using touchscreens directly with the thumb for performing a wide variety of mobile tasks.

Specialized Software

Another strategy for enabling one-handed use of touchscreen devices is to focus on the software design. Beyond my own work, few others have developed touchscreen-based applications that specifically support one-handed thumb operation. A notable exception is Huot and Lecolinet's SpiraList [67], which offers users a focus+context view of a text-based list in a spiral layout (Figure 4); the SpiraList supports serial navigation via a thumb-operated widget that modulates the speed and direction of list rotation (Figure 4a), or users can aim for visible items using an offset-cursor [116] (Figure 4b). Unfortunately, the technique was not studied formally to understand whether the properties of thumb operability and the increased visibility of items in long lists benefits user tasks over standard, scrollable 1D list layouts. Although there are few other examples of *research* into specific applications, commercial trends for supporting thumb operation of touchscreen-based devices include several thumb-based virtual keypads available from third party vendors, such as the Phraze-It® keypad from Prevalent Devices (www.prevalentdevices.com).

(a) **(b)**

Figure 4. SpiraList, a one-handed thumb-based design for viewing and navigating text-based lists on a touchscreen device. Browsing via an indirect list control widget (a), and direct item selection using an offset cursor (b).

Researchers have also suggested general software design guidelines for serving the interests of one-handed thumb-based touchscreen interaction. For example, Pascoe et al. [113] and Kristoffersen and Ljungberg [79] both concluded that if software is structured so that only a small number of interaction objects occupy each screen, and each object is assigned to a conceptually distinct region of the display, such as "top, middle, bottom", "top-left, top-right, bottom-left, bottom-right" [113], or "North, South, East and West" [79], then assuming the interaction objects are large enough, users stand a good chance being able to hold and operate the device with just one hand and minimal visual attention, especially when coupled with audio feedback. An example of an interface by Pascoe et al. that follows these guidelines is shown in Figure 5. Unfortunately such simplified software designs for touchscreens are not practical for most modern applications, which must balance input requirements with those for effective information presentation and efficient navigation.

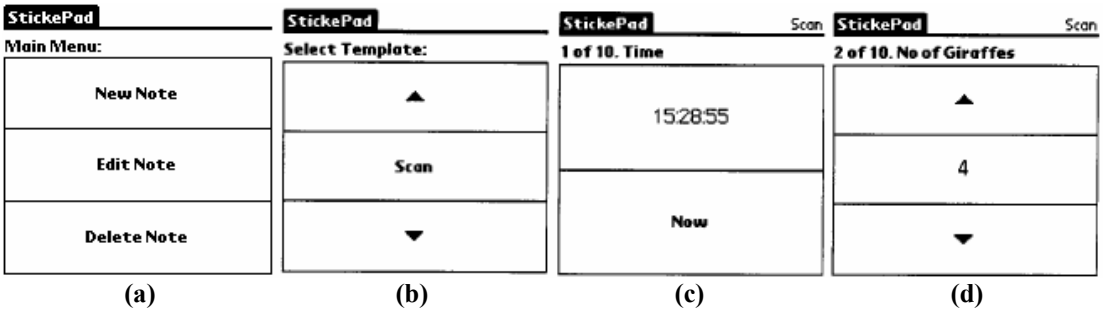


Figure 5. An interface that embodies the design guidelines of Pascoe et al. for supporting one-handed touchscreen interaction. (a-d) The first four steps of entering a new field note.

One of the few handheld *systems* dedicated to single-handed operation is the touchscreen-based N2 phone by Neonode (www.neonode.com), which supports application navigation and interaction using only thumb taps and sweeps. However, the N2's primary use is as a phone, camera and media player, rather than a personal data manager. As such, the N2 is not designed to support rich graphical interfaces or data

interactions. The Apple iPhone (www.apple.com) and LG Prada phone (www.pradaphonebylg.com) are two touchscreen-based mobile phones dedicated to finger-only interaction, primarily by offering users large interaction targets. It is unclear whether these systems were designed specifically for one-handed thumb interaction since both use the entire display space for object placement, potentially placing some objects out of thumb reach. Even so, a review of web-based articles and message boards about the two phones suggests that many users can indeed operate both the iPhone and Prada phone with one hand.

Another option for reducing the number of hands needed to operate a touchscreen-based device is to wear the device instead of hold it, which frees both hands when no interaction is required, and demands the use of only a single finger otherwise. In [25, 115] Brewster et al. conducted a series of studies to explore interaction techniques for operating a hip mounted touchscreen device. Because the low position and vertical orientation of the device reduced the visibility of the display, their system de-emphasized the need to see and hit specific targets by supporting audio-enhanced command gestures. Although the system did support one-handed operation, the investigators only pursued a single media player application since the primary focus of the work was on understanding the benefits of audio feedback during eyes-free device operation. For example, in [115] Pirhonen, Brewster and Holguin demonstrated that users experienced significantly lower workload, faster input time, and faster walking speed when using an audio-enhanced gesture language to operate the hip-mounted touchscreen music player than when using the traditional interface only.

2.1.5 The Role of Audio in Mobile Interaction

Audio input and output are generally used so that a user does not have to look at a device in order to interact with it, thereby freeing the user's visual resources for monitoring the environment. Although audio has been applied much more widely as a means for output than input in mobile applications, it is the use of audio for input that has had the most impact on reducing the hand requirements of mobile computing, since it allows a user's hands to be free for carrying items, opening doors, driving—whatever the physical requirements are for the task.

Audio Input

Using speech for device input has received more public attention than non-speech methods, perhaps because of the naturalistic quality of the interaction and its anthropomorphic appeal: how nice it would be if we could speak to a computer as if with a personal assistant and it would do as we say! Speech recognition technology can be very accurate for basic speech-to-text translation activities such as dictation, especially when trained to an individual. However, the greater challenges involve the human-computer dialog when speech is used to actually *interact* with a device. Systems that support operating a device via voice typically support only a limited vocabulary, which users must either memorize, thereby increasing mental demand, or be prompted for, which slows down interaction [140].

While speech input is very flexible and can be applied to a wide range of applications, it can be slow for rapid direct input since recognizers typically wait until the user has completed an utterance before classifying the input [138]. Non-speech utterances such as humming, on the other hand, can be processed dynamically, and so can offer the

possibility of faster system response [138]. In addition, non-speech may be better suited than speech for setting certain types of parameters and fine-grained input control, since continuous voice characteristics like pitch and volume can be interpreted directly, but performing the same tasks with speech-based commands might require tedious repetition [140].

Many audio input systems have been developed to reduce the physical requirements of desktop computing for users with disabilities. However, because these systems assume a standard desktop computer setting, they do require a high degree of visual attention for performing tasks such as cursor manipulation [40, 59, 73, 98, 139], game play [138], and web-browsing [34].

Because audio is typically used expressly to reduce visual demand in mobile scenarios, some speech-based systems that have targeted the mobile domain such as VoiceNotes [140] and NomadicRadio [126] do not even have a display; each system does, however, allow for button-based input when speech input would be inappropriate. In addition, these systems were developed for differing aspects of taking, receiving, and navigating messages, and so have more specialized uses than do today's mobile phones. High-end mobile phones often support limited speech input for dialing (e.g., "call *contact* at [*work* | *home*]"), however, general purpose speech-based navigation and control to accomplish common mobile tasks (e.g., "search for pizza places close to home") is not yet available.

Audio Output

Audio output is usually provided as non-speech feedback in response to button or touchscreen input to reduce or eliminate the amount of visual attention required to

interact with a device while mobile. The use of audio, therefore, does not inherently reduce the hand requirements of mobile computing, but audio feedback has indeed been used to support one-handed interaction in cases where a user's capacity to visually monitor the interaction outcome has been reduced. Continuing to focus on hip-mounted touchscreen interaction, Brewster extended his research of audio-enhanced gestures for media play [115] by dividing the touchscreen into a 3x3 audio landscape of short composite audio cues, or "earcons", and demonstrating that users were more accurate in drawing symbols on the screen when the gestures were accompanied by audio feedback than when they were not [25]. Brewster et al. [24] also demonstrated that interface objects could be shrunken without reducing input performance if simple audio cues were provided to convey hitting (success) or slipping off (error) a target when using a stylus. As a complement to these experiments McGookin and Brewster [96] explored the design space and usability limits of earcons for conveying multiple system states simultaneously.

Despite the potential benefits of audio input and output, they face the situational challenges of 1) interfering with ambient noise and 2) being overheard. Noisy environmental conditions can make it difficult for users to hear sounds emanating from a device, and can also degrade the accuracy of command recognition when sound is used for input. In social situations, the use of audio I/O may be disruptive to others and can compromise the privacy of the owner's information. For these reasons, audio is not typically provided as an exclusive interaction method; rather, audio is offered to enhance usability for highly mobile situations, while visual and physical mechanisms can be relied upon for operating the device silently when more attention can be devoted to the task.

2.2 Exploratory Study 1: Field Study

One motivation for my research into supporting one-handed use of mobile devices was my assumption that people already use devices in this manner, and the implication that current use behavior has predictive power over future use behavior. For example, if mobile users operate a device with one hand because of habit, preference, or the constraints imposed by mobility, the same reason will likely apply to different tasks and devices as well. So to first establish the current extent of one-handed behavior, I conducted an *in situ* study of user interaction with mobile devices. The study targeted an airport environment for the high potential of finding mobile device users and its ease of access for unobtrusive observation.

I was the sole investigator of this study, which was conducted in November 2003, and supervised by my advisor. The results of this study have been published as a component of a chapter in the *Handbook of Research on User Interface Design and Evaluation for Mobile Technology* [75].

2.2.1 Method

I observed 50 travelers (27 male, 23 female) at Baltimore Washington International Airport's main ticketing terminal during a six hour period during peak holiday travel in November 2003. Because the observation was limited to areas accessible to non-ticketed passengers, seating options were scarce. I expected to observe the use of both PDAs and cell phones since common traveler activities include coordinating transportation, catching up on work, and using mobile devices for entertainment purposes. Since most users talk on the phone with one hand, I recorded only the cell phone interactions that included

keypad interaction as well. All observations were performed anonymously without any interaction with the observed device users.

Note that while any subject observation without consent presents a legitimate question for ethical debate, for this research we used the federal policy on the protection of human research subjects [42] as a guideline. The policy states that the observation of public behavior is not regulated if the anonymity of the subjects is maintained and that disclosure of the observations would not put the subjects at risk in terms of civil liability, financial standing, employability, or reputation. Since we were interested in capturing natural behavior, did not record identifying characteristics, and consider phone use while standing, walking and sitting to be relatively safe activities, we did not obtain subject consent.

2.2.2 Measures

For each user observed, I recorded sex, approximate age, and device type used: candy bar phone, flip phone, Blackberry, or PDA. A "candy bar" phone is the industry term for a traditional-style cellular phone with a rigid rectangular form, typically about 3 times longer than wide. For phone use, I recorded the hand(s) used to dial (left, right or both) and the hand(s) used to speak (left, right or both). I also noted whether users were carrying additional items, and their current activity (selected from the mutually exclusive categories: walking, standing, or sitting).

2.2.3 Results

Only two users were observed operating devices other than mobile phones—one used a PDA and the other used a Blackberry. Both were seated and using two hands. The

remainder of the discussion focuses on the 48 phone users, 62.5% of whom used flip phones, and the other 37.5% used candy bar phones. Overall, 74% used one hand for keypad interaction. Sixty five percent of the one-handed users had a hand occupied, and by activity, 54% were walking, 35% were standing, and 11% were sitting. Figure 6 presents the distribution of subjects who used one vs. two hands for keypad interaction, categorized by the activity they were engaged in (walking, standing, or sitting). The distribution of users engaged in the three activities reflects the airport scenario where many more people were walking or standing than sitting.

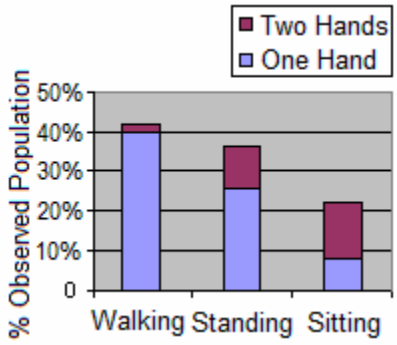


Figure 6. Airport Field Study: The percentage of observed travelers using one vs. two hands, by activity.

It is clear from Figure 6 that the relative proportion of one-handed to two-handed phone users varied by activity; the vast majority of walkers used one hand, about two-thirds of standers used one hand, but seated participants tended to favor two hands to operate their devices. However, I also recorded whether one hand was occupied during the activity, and found that walkers were more likely to have one hand occupied (60%), followed by standers (50%), and then sitters (25%), which may be the true reason walkers were more likely than standers to use one hand for device operation, and why standers were more likely than sitters to use one hand. Regardless of activity, when both hands

were available for use, the percentage of one vs. two-handed phone users was equal (26%).

2.2.4 Discussion

Although Figure 6 suggests that a relationship exists between user activity and keypad interaction behavior, it is unclear whether activity type influences the number of hands used, or vice versa. Furthermore, since the percentage of users with one hand occupied correlates with the distribution of one-handed use across activities, it is possible that hand availability, rather than activity, is the more influential factor in determining whether one or two hands are used for interacting with the keypad. While use scenario certainly impacts usage patterns, the fact that users were as likely to use one hand as two hands when both hands were available suggests that preference, habit and personal comfort also play a role. Given that almost three-quarters of the observed travelers operated their phones with one hand, we can safely conclude that one-handed phone use is quite common, and thus is an essential consideration in mobile phone design. More importantly, the data provide evidence that the more mobile a user is, the more likely she is to use one hand to interact with her mobile device; whatever the reason for this trend—be it increased user mobility or decreased hand availability—it is likely that the same factor will influence user choice when operating a touchscreen device as well.

Generalizability

Note that my choice of observation location may have biased my results from behaviors found in the general population since travelers may be more likely to be: 1) carrying additional items; 2) standing or walking; and 3) using a phone vs. a PDA. Different

environments, tasks, populations, and scenarios will yield unique usage patterns. My goal was not to catalogue each possible combination, but to learn what I could from a typical in-transit scenario.

2.3 Exploratory Study 2:Web Survey

While the airport study was sufficiently informative for a preliminary exploration, shortcomings of the results include 1) the lack of knowledge or understanding of users' motivations for usage style; 2) the limited types of devices observed (phones); and 3) the limited tasks types observed (assumed dialing). To broaden my understanding of device use across these dimensions, I conducted a follow up survey to capture user perceptions of, preferences for, and motivations surrounding their own device usage patterns.

I was the sole administrator of this survey, which was conducted in January 2004, and supervised by my advisor. Both the survey questions and raw results can be accessed on the web at http://www.cs.umd.edu/hcil/mobile/survey/. The results have also been published as part of a chapter in the *Handbook of Research on User Interface Design and Evaluation for Mobile Technology* [75].

2.3.1 Method

The survey consisted of 18 questions presented on a single web page which was accessed via an encrypted connection (SSL) and hosted on a server of the Department of Computer Science, at the University of Maryland, College Park. An introductory message informed potential participants of the goals of the survey and assured anonymity. Notification that results would be posted for public access after the survey period was over provided the only incentive for participation. Participants were solicited from a voluntary subscription

mailing list that publishes activities associated with the university's Human-Computer Interaction Lab (HCIL). In addition, the solicitation was propagated to one recipient's personal mailing list on the topic of medical informatics, and a link to the survey was posted on two undergraduate Computer Science course web pages.

2.3.2 Measures

For each participant, we collected age, sex and occupation demographics. Users recorded all styles of phones and/or PDAs owned, but were asked to complete the remainder of the survey with only one device in mind—the one used for the majority of their information management tasks. We collected general information about the participant's primary device, including usage frequency, input hardware, and text entry method. We then asked a variety of questions to understand when and why participants typically use one vs. two hands to operate a device. We asked participants to record the number of hands they use (one and/or two) for eighteen typical mobile tasks, and then asked them to specify the number of hands (one *or* two) they would *prefer* to use for each task. For three applications (email, calendar, and contacts) we divided tasks in to "reading" (email reading, calendar lookup, and contact lookup) vs. "writing" (email writing, calendar entry, and contact entry) activities to help us understand how hand usage patterns change for different types of tasks within the same application. Participants then recorded the number of hands used for the majority of their device interaction and under what circumstances they chose one-handed use over two-handed use, and vice versa. Finally, participants were asked how many hands they would prefer to use for the majority of device interactions (including no preference).

2.3.3 Results

Two hundred twenty-nine participants (135 male) responded to the survey solicitation. One male participant was eliminated from the remaining analysis because his handheld device was specialized for audio play only, leaving 228. Median participant age was 38.5 years. Participant occupations reflected the channels for solicitation, with 25% in CS, IT or engineering, 23% students of unstated discipline, 20% in the medical field, 10% in education, and the remainder (21%) from other professional disciplines. We obtained institutional review board (IRB) approval to conduct this study.

Devices Owned

The three most common devices owned were flip phones (52%), small candy bar phones (23%) and Palm devices without a Qwerty keyboard (20%). Palm devices with an integrated Qwerty keyboard were as common as Pocket PCs without a keyboard (14%). Since interaction behavior is likely to depend on device input capabilities, we reclassified each user's primary device into one of four general categories based on the device's input channels: (i) *keypad-only (51%)* are devices with a 12-key numeric keypad but no touchscreen, (ii) *TS-no-qwerty (23%)* are devices with a touchscreen but no Qwerty keyboard, (iii) *TS-with-qwerty (21%)* are devices with a touchscreen as well as an integrated Qwerty keyboard, and finally (iv) q*werty-only (5%)* are devices with an integrated Qwerty keyboard but no touchscreen. Note that due to the severe imbalance among the device types owned in the participant population, I was unable to use device type as a between-subjects factor in the analysis; I do, however, report the device type data graphically. For users with multiple devices, I derived their primary device type from the text entry method reported.

Current Usage Patterns

A 2 (*number-of-hands*: one vs. two) x 18 (*task-type*) Repeated Measures Analysis of Variance (RM-ANOVA) was run on the current usage data. Significant main effects for both *number-of-hands*, $F(1, 255)=14.3$, $p<.001$, and *task-type*, $F(8.5, 2163.3)=87.3$, $p<.001$, were observed. A significant interaction between *number-of-hands* and *task-type* was also found, $F(9.9, 2529.9)=43.5$, $p<.001$. Post hoc analyses of main effects were conducted using Bonferroni correction.

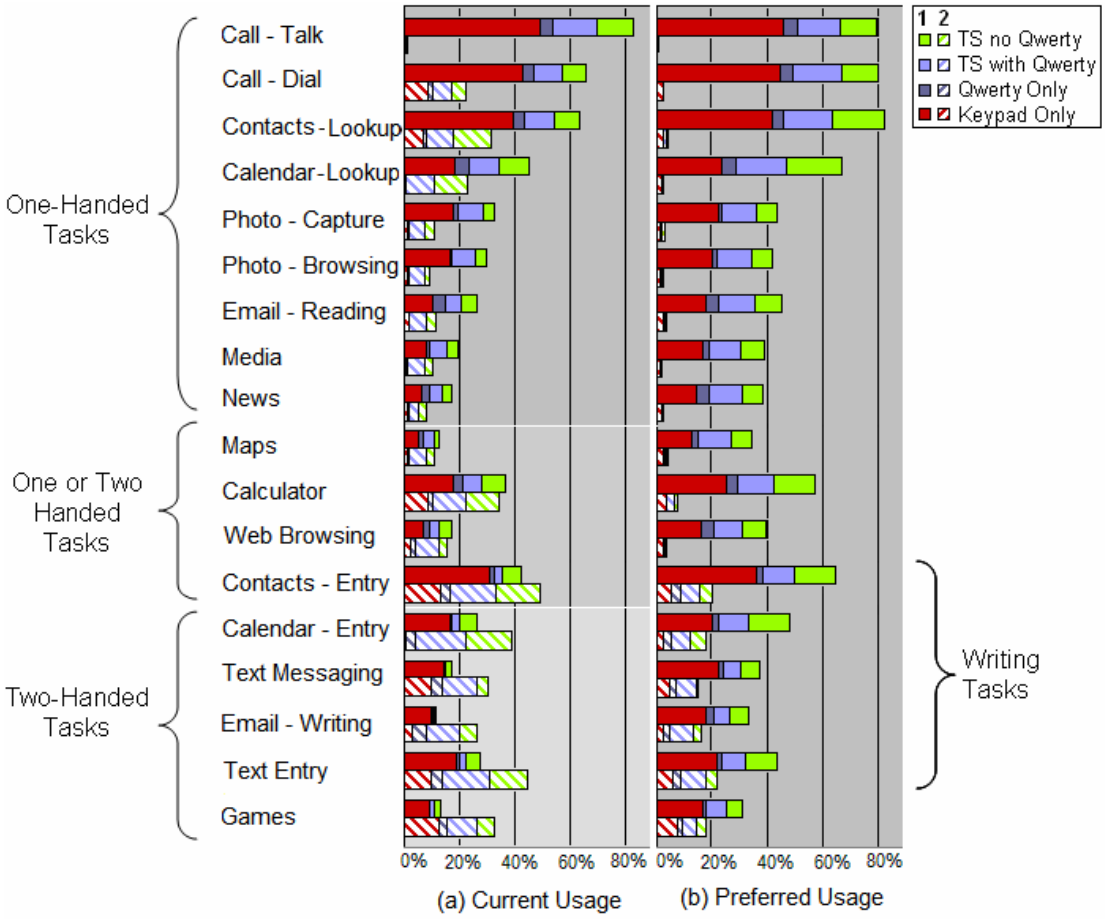


Figure 7. Web Survey: The number of hands currently used and preferred to be used for 18 common mobile tasks, as a percentage of the observed population. One-handed use is shown as solid bars, two-handed use is shown as striped bars. The different colored backgrounds indicated tasks for which one-handed use was appreciably more common than two-handed use (darkest), tasks for which hand choice was roughly equal (medium), and finally tasks for which two-handed use was more common than one-handed use (lightest). Hand usage for each task is broken down by device type. Note that TS=touchscreen.

Overall, significantly more participants use one hand to perform common tasks than use two hands, (29% vs. 20%). The main effect of *task-type* can be explained by the fact that users did not record answers for tasks they never perform (e.g., many users cannot browse the web from their phones). Thus regardless of the number of hands used, the user participation level varied considerably for different tasks.

Trends shown in Figure 7 help explain the interaction result between the *number-of-hands* used and *task-type*. Of the 18 tasks included in the survey, more participants used one hand than used two hands for 9 of the tasks (the top 9 tasks in Figure 7a, shown with the dark background), more participants used two hands than one hand for 5 of the tasks (the bottom 5 tasks in Figure 7a, shown with the light background), and roughly equal numbers of participants used one hand as two hands for 4 of the tasks (shown with the medium gray background in Figure 7a). Upon inspection, all of the "reading" activities were performed more often with one hand (darkest background) and nearly all "writing" activities with two hands (lightest background). Considering users' device types, we notice that with the exception of gaming, owners of *keypad-only* (e.g., 12-key) devices tended to use one hand rather than two hands for tasks; owners of *TS-no-qwerty* devices were more likely to use two-hands for most activities, and those owning Qwerty based devices tended to use two hands when performing writing tasks, but one hand for reading tasks.

Overall, 45% of participants stated they use one hand for nearly all device interactions, as opposed to only 19% who responded similarly for two hands (Figure 8a). Considering device ownership, however, users of touchscreen-based devices were more likely to use two hands than one hand "always". When participants use one hand, the

majority (61%) report they do so whenever the interface supports it, which was the reason cited by only 10% of those who use two hands (Figure 8b). Device form factors influenced usage behavior when the device was too small for two hands, too large for one hand, or when large devices could be supported by a surface and used with one hand. Participants cited task type as a reason for hand choice, primarily as a trade off between interaction efficiency and hand resource usage: 14% of users selected one hand only for simple tasks (conserving resources), while 5% selected two hands for entering text, gaming, or for otherwise improving the speed of interaction (favoring efficiency). Finally, according to respondents, the majority of two-handed use occurs when it is the only way to accomplish the task given the interface (63%), shown in the last column of Figure 8b.

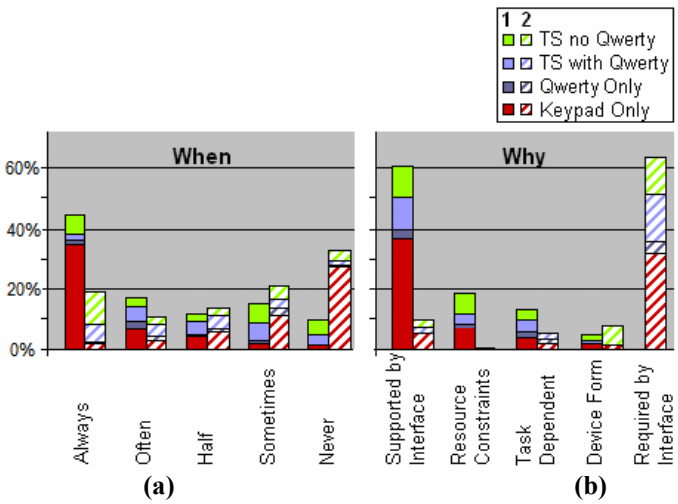


Figure 8. Web Survey: The average frequency with which participants use one vs. two hands to operate their devices (a), and participants' reasons for their hand choices (b). One-handed use is shown as solid bars, two-handed use is shown as striped bars, segmented by device type.

Hand Preferences

When participants were asked how many hands they would *prefer* to use to perform each of the 18 tasks, one hand was preferred overwhelmingly to two hands for all tasks (Figure 8b). The activities with the closest margin between the number of participants who

preferred one vs. two hands were playing games (13%) and composing email (16%). With one exception (gaming), the activities for which more than 14% of users stated a preference for two hands were "writing" tasks (e.g., those that required text entry): text entry, contact entry, calendar entry, email writing, and text messaging, in decreasing order. Even so, except for users of *TS-with-qwerty* devices, the majority of users stated a preference for using one hand, regardless of task or device owned. Users of *TS-with-qwerty* devices preferred two hands for text messaging, email composition, and text entry. Based on these data, it is consistent that 66% of participants stated they would prefer to use one hand for the majority of device interaction, versus 9% who would prefer two hands for all interaction. Twenty-three percent did not have a preference and 6 users did not respond.

2.3.4 Discussion

Considering current usage patterns only, there is no obvious winner between one-handed and two-handed use. Excluding phone calls, the number of activities for which appreciably more respondents use one (7) vs. two hands (5) is nearly balanced. However, device type certainly seems to influence user behavior; owners of keypad-only devices nearly always used one hand, while owners of touchscreen devices more often favored two hands, especially for tasks involving text entry. But participant justifications for hand choice indicate that the hardware/software interface is to blame for much of the two-handed device use reported. Most participants use one hand if at all possible and only use two hands when the interface makes a task impossible to perform otherwise. Other than gaming, tasks involving text entry are the only ones for which users may be willing to use two hands, especially when the device used provides an integrated Qwerty keyboard. It

seems, therefore, that the efficiency gained by using two hands for such tasks is often worth the dedication of physical resources, which is also true of the immersive gaming experience. It is also worth noting that while many participants who reported using two hands for writing tasks claimed they would prefer to use one hand, enabling single-handed text input may not be sufficient for those users to switch to one-handed text entry in *practice*, since throughput is an important satisfaction measure that is almost certain to drop in the transition. Even so, participants' responses overall send a consistent message that interface designers for all device types should make one-handed usability a priority, and strive to bridge the gap between current and desired usage patterns.

2.4 Conclusion

Inspired by the goals of minimal attention user interfaces (MAUIs), and evidence that few researchers have yet to focus their efforts on developing interfaces or interaction methods to support one-handed operation of stylus-oriented touchscreen devices, I performed two exploratory studies to assess the potential benefits of a research agenda aimed at improving one-handed interaction for touchscreen-based mobile devices. Results from my *in-situ* airport study and survey of actual and preferred mobile device use confirmed that one-handed device operation is widespread—not limited to a niche user segment. Furthermore, users reported they would prefer to use one hand more often than the hardware/software designs of their devices allow. The greatest disparity between the number of people currently using one hand to perform tasks and the number that would prefer to use one hand was found for owners of devices with touchscreens; nearly all respondents who use two hands to perform reading based tasks with touchscreen

devices would prefer to use one hand, as would roughly half of those currently performing text entry tasks with two hands.

Together, the results from my preliminary studies justify the research in the remainder of this thesis, as they suggest significant benefits might be achieved in device usability and user satisfaction by improving mobile device support for one-handed touchscreen operation.

Chapter 3
Foundations: Human Factors in One-Handed Mobile Device Interaction

Outcomes from the exploratory studies I undertook to project the potential value of my research provided ample evidence that mobile users could benefit greatly from new interfaces and interaction techniques for one-handed device operation targeting all device types. Following these studies, my first goal was to understand some of the basic human factors involved in single-handed device operation from which I might derive generalized design guidelines. I pursued three studies to generate the empirical data for backing such guidelines, having the following goals: 1) to understand the limits of thumb mobility when holding a device in one hand, especially how device size, target location, and thumb movement direction influence task performance; 2) to determine the required sizes for thumb-oriented touchscreen targets, and 3) to assess the learnabilty and executability of touchscreen-based thumb gestures. These studies, together with those from the previous chapter, constitute the *foundations* of my research.

3.1 Related Work

3.1.1 Ergonomics

Ergonomics, or human factors, is "an applied science concerned with designing and arranging things people use so that the people and things interact most efficiently and safely", according to the Merriam-Webster Dictionary (www.m-w.com). The discipline of ergonomics is applied to the development of a vast range objects we all encounter in

everyday activities, as well as those that professionals use in specialized fields. It is no surprise, therefore, that ergonomics plays an important role in the design and use of mobile devices. Unfortunately, little knowledge has escaped the corporate research labs for public consumption, or in a format that software designers might use to guide the development of more effective mobile interfaces. However, several high-level consumer reports have been issued in recent years warning of potential injury to the thumb resulting from overuse in text messaging [8, 38].

Scientists in the medical community have studied the biomechanics of the thumb extensively for the purposes of both reconstruction and rehabilitation. The static structure of the thumb has been understood for the last decade [9], but only recently have scientists begun to reliably quantify the functional capabilities of the thumb. In the medical domain, strength is the primary parameter used to assess mechanical ability, and the influence of movement direction upon thumb strength has been established both at the anatomical [114] and functional [23, 85] levels. However, these strength-direction, or *force vector* measurements have only been recorded for standard anatomical planes of movement. This excludes the plane parallel to the palm, which is the one that is most relevant to operating a mobile device with the thumb when held in the same hand.

As a complement to the data on thumb force capabilities, others have looked at the *extent* of thumb movement. Kuo [80] has developed a model for the maximal 3D workspace of the thumb, and in the HCI field, Hirotaka [65] has quantified an average for thumb rotation angle. In addition, average static hand measurements for both men and women are published in [145]. Unfortunately, the conditions under which these data have been gathered do not account for the additional external constraints that would be

imposed by holding objects of varying size, as would be the case when using a handheld device. In [65] Hirotaka also outlined how the cell phone might be redesigned for improved ergonomics and interaction based on the device's center of mass and the static range of thumb motion. Although qualitative user feedback on Hirotaka's design was positive, the design was not evaluated formally. My approach differs in that I have focused on understanding the dynamic capabilities of the thumb based on user performance data while holding various sized devices with one hand.

The government of the United Kingdom has published extensive ergonomics guidelines for the design of military equipment. The two most common types of grip, the precision grip and the power grip, are covered in [103]. A precision grip, which is logically the most appropriate for manipulating a mobile device, is characterized by using the pad of the thumb in opposition to the pads of the other fingers. However, this does not describe the movements of the thumb when operating a mobile device in one hand, where the thumb is used in opposition to the plane of the palm. Neither could these movements be described as a power grip, in which the thumb is *not* rotated at the base to oppose the fingers, and the remaining fingers are flexed such as when opening the lid of a jar. Thus it seems that the role of the thumb in operating a mobile device one-handed defies the classic definitions of hand grip, suggesting that attention is warranted in understanding the limitations of the thumb when used in this manner.

Considering the more aesthetic aspects of human factors in mobile device design, Yun et al. [159] investigated which cell phone features contribute most to user satisfaction by asking 76 users (38 male, 38 female) across 3 age groups (teens, twenty-somethings, and thirty-somethings) to rate 50 cell phones along 10 dimensions:

luxuriousness, simplicity, attractiveness, colorfulness, texture delicacy, harmoniousness, salience, ruggedness, and overall satisfaction. Together with a characterization of each phone based on 56 design elements, the ratings were used to develop a model for the relationship between age, gender, and desired phone design properties. While the outcomes of this work lend insight into the physical characteristics valued by consumers, my research has the potential to offer complementary guidelines for improved input efficiency, such as button placement and crowdedness—neither of which were parameters featured in this work.

3.1.2 Target Sizes for Touchscreens

An open question in developing software interfaces for thumb-based touchscreen interaction is how large targets should be to minimize input error. Several past studies have indeed explored appropriate target sizes for touchscreen-based mobile devices [24, 92, 105, 106, 132] and desktop-sized touch-sensitive displays [36, 130]. Unfortunately, due to the specific parameters of each study, the resulting recommendations are not strictly applicable to thumb-based interaction for mobile device touchscreens. Even though previous PDA studies have targeted the same platform I am interested in, they have focused on two-handed stylus input rather than single-handed thumb input. Studies that address desktop-sized displays, on the other hand, have considered finger-based interaction, but recommendations cannot be directly applied since 1) the tip of an index finger is typically smaller than that of a thumb, and 2) users of desktop displays have different motor constraints than users of mobile devices, who must contend with holding a device and interacting with its surface with the same hand.

Target Sizes for PDAs

Research efforts to determine appropriate target sizes for stylus-based mobile touchscreen tasks have drawn different conclusions about whether target size affects performance. MacKenzie and Zhang [92] found no difference in text entry rates between two Qwerty-based virtual keypads, one with 6.4 mm wide keys and the other with 10 mm wide keys. While these targets are fairly large for stylus entry, Sears and Zha [132] confirmed and extended this finding for keys from 2.6 - 4.4 mm wide. However, in studying single-target selection tasks for targets between 2 - 5 mm, Mizobuchi et al. [106] generally found speed and error rate *improved* with increases in key size, and though Brewster [24] was specifically interested in the influence of audio feedback on user performance as target sizes changed, he too found a significant improvement in throughput when targets increased from 2.5 to 5 mm.

While these results seem contradictory, they are both consistent with Fitts' model for motor movement [50], which defines movement time (MT) with respect to the distance to (or amplitude, A) and size (W) of the target as:

$$MT = a + b\,(\log_2(A/W + 1)) \qquad (1)$$

The constants *a* and *b* have been described as representing the efficiency of the pointing device in question (here, the stylus on a touchscreen). The log term, or index of difficulty (ID), is defined here as in [88], and embodies the intuition that targets are harder to hit the farther they are, but easier to hit the larger they are. Thus the lower a target's ID, the easier (e.g., faster) it will be to hit. In the text entry studies of MacKenzie and Sears, the keypads scaled uniformly, which maintained constant IDs across changes in key sizes; since IDs were equal in each condition, it makes sense that performance rates were also

the same. However the task designs of Mizobuchi and Brewster varied only target size, not distance, so IDs were not the same across conditions. Thus here, too, the results are consistent with Fitts' Law, which would have predicted the smaller targets would be more difficult, and thus slower to hit.

In a study by Himberg et al. [62], subjects used the thumb of their primary hand for interacting with a soft keypad located at the edge of a touchscreen-enabled laptop PC. The laptop had a flip phone cover attached to the back of the display in order to make the interaction more similar to one-handed use of a handheld device. However, the goal of the study was to explore the viability of keypad adaptation, rather than to develop size recommendations for thumb targets.

Finally, Microsoft has published a recommendation that buttons should be 9 mm large for finger interaction on a touchscreen-based mobile device [152]. Unfortunately, the text is unclear about whether this recommendation assumes index fingers or thumbs. However, since one-handed use of touchscreen-based devices has only recently gained attention, it is quite likely the 2002 guideline referred to index fingers. Under this supposition, we would expect that the recommendations for thumb targets would need to be larger, and so my own research is interested in how *much* larger.

Target Sizes for Desktop Sized Displays

When desktop-sized touchscreen displays entered into public use, early studies were designed to better understand the new technology, such as comparing finger input to traditional devices [129, 131] and quantifying the impact of display parallax [58, 129]. Although technological advances date some of these results, others are still relevant. Potter at al. [116] demonstrated that users could select items with a finger more reliably if

selection was delayed until a user removed the finger over the desired target (lift-off) rather than immediately upon contact with the screen (land-on). In fact, by combining the lift-off strategy with an offset cursor and input stabilization, Sears and Shneiderman [131] showed that touchscreens could support target access speed and accuracy for the finger that rivaled the mouse for targets as small as 1.7x2.2mm. Even so, selection times were fastest and error rates lowest for the largest targets tested (13.8x17.9mm).

In an early study of touchscreen-based keyboards, Sears and Shneiderman [130] investigated the interaction between key size and typing speed for novice and experienced users of a desktop-sized (640x480) touch screen display. Keys were sized between 5.7 mm and 22.7 mm, arranged in a Qwerty layout, selected using any finger(s) from either hand, and supported the lift-off selection strategy. They found text entry rates increased with key size for both novice and experienced users, and that novices made significantly fewer errors on the largest keyboard (3.6%) vs. the smallest (6.7%). Because the keyboards in the study scaled uniformly, these results contradict those of MacKenzie and Zhang [92] and Sears and Zha [132], who all found that keyboard size alone had no impact on speed and errors when studying stylus-based text entry on touchscreens. Reasons for this difference include the fact that two of the targets sizes in Sears and Shneiderman's study (7.6 mm and 5.7mm) were smaller than that of the average index finger, and so occlusion and the characteristics of resistive touchscreen technology may have interfered with the predictions of Fitts' Law (all the targets in the stylus-based studies were larger than the tip of the stylus). Also, since users were entering text with both index fingers in parallel in Sears and Shneiderman's study, it is possibly that the

different keyboard and label sizes affected search and planning unequally, making users faster the larger and more visually salient the letters were.

More recently Colle and Hiszem [36] studied the effects of manipulating the size and spacing of targets on a touch-sensitive kiosk display, using a similar design to [24]. In their experiment the participants used their right hand index finger to enter 1, 4 or 10 digits using a virtual numeric keypad. In accordance with [131], they found that targets should be between 20mm and 25mm. However, these sizes, as well as the 22.7 mm key size of the largest keyboard in the Sears and Shneiderman [130] study, are all impractically large for general use on a handheld device, allowing for only 9 targets on a 3.5" screen.

Motivated by the requirement for efficient text and numeric entry, the majority of previous investigations into optimal target sizes have preferred experimental designs modeled after data entry tasks. However, Colle and Hiszem [36] presented interesting results that suggest that there is a difference between tasks that require selection of a single target (e.g., selecting a button, checkbox, or menu alternative), and those comprising a rapid sequence of selections (e.g., text or numeric entry). They found error rate decreased when target sizes increased from 10 mm to ≥15 mm for numeric sequences of 4 and 10 digits, but that error rate remained constant for single-digit entry. One possible explanation for the differences observed might be that for multi-digit entry, users could plan for the next item *during* the execution of the previous item, making serial entry more efficient. This is supported by the fact that for all target sizes, users spent more time per character for single-character tasks.

3.1.3 Gestures for Input

Gestures have been explored in many nontraditional computing settings when standard input methods or interaction styles impose unique challenges or limitations for effective system use. For example, strokes, as opposed to taps, are a natural way to interact with pen based systems, and so have been used frequently as the building blocks in gesture languages, including those for menu item selection (e.g., [55, 63]) and text entry [22, 78, 93, 155], and also more specialized tasks such as video editing [117], distant object selection [12], and setting drawing parameters [4]. The appeal of gestures is that they efficiently combine command and operand into a single motion, and can reduce the need for software buttons and menus, especially useful under the space constraints of handheld interface design. Furthermore, gestures are an attractive solution for one-handed mobile touchscreen interaction because they don't require users to hit specific targets; designers then do not need to be concerned with making sure targets are large enough or close enough to be hit with the thumb, and so frees them to focus on the challenges associated with effective information presentation.

Stylus-based gestures have been explored for a variety of mobile touchscreen applications. With Collapse-to-zoom [14], users issue diagonal strokes over uninteresting elements of a web page to remove them from view, which frees screen space to display the remaining elements in more detail. The Power Browser [30] supported simple left-to-right and right-to-left pen strokes to navigate web pages and up/down strokes for scrolling. Nicholson and Vickers also explored more complex multi-stroke gestures for browsing movie listings from a mobile device [110]. Because most of these systems use simple gestures they may be adaptable for thumb use, but they remain two-handed

techniques in their original stylus-based incarnations. An exception is Apple's iPhone, which *is* designed for finger use, and uses simple gestures in a few key places: a left-to-right finger sweep unlocks the devices, and the same gesture is used to enter 'delete' mode for list items.

The potential that gestures hold for supporting single-handed device interaction has intrigued many researchers. Media play has been a popular application for single-handed gestures, exemplified by Apple's thumb-controlled circular Click Wheel for navigating music on the iPod [5]. Pirhonen et al. [115] explored the use of a 5-gesture language coupled with audio feedback for supporting eyes-free, index finger control of a music application on a hip-mounted touchscreen PDA, and found users performed gestures significantly more accurately with audio feedback than without. Brewster et al. [25], extended this work to include spatial sounds and head-nodding for navigating and interacting with a broader range of applications, and enhanced the gesture feedback to include both pitch and tone for communicating finger position within a 3x3 gesture input grid. An evaluation of the enhanced feedback for an expanded set of 12 gestures again showed that users performed gestures more accurately with audio feedback.

The iPod Click Wheel has inspired other variations on list navigation with circular thumb gestures. The earPod by Zhao et al. [161] has no visual interface, but is a small handheld control that supports eyes-free hierarchical menu navigation by linking thumb movements on a circular touchpad to highly responsive spoken audio menu items. Unlike the iPod's Click Wheel, which interprets relative finger movement rather than absolute finger position for traversing a list, the earPod is divided into sectors that are each assigned to a different menu item—a design which limits the structure of the menu

system to a relatively small number of items (e.g., 8) at each level. The SpiralList [67] combines some of the properties of the Click Wheel and earPod for navigating lists on touchscreen-based mobile devices. The SpiraList displays menu items alphabetically in a circular coil, providing a focus+context view of a long list by showing the details of items on the outermost ring and index cues for items within the inner rings. As with the earPod, a user can select an item directly with the thumb, but the user can also browse the list serially by dragging her thumb along an arc-shaped widget at the right of the spiral to rotate it. Although the spiral rotation is controlled using relative thumb motion, the SpiraList differs from the Click Wheel in that the speed of the list rotation is based on the amount of finger displacement from the point of touchdown rather than the speed of finger movement.

Although gesture-based interfaces can obviate the need for command buttons or widgets and thus free screen real estate for displaying informational content, the invisible nature of gestures can make them hard to remember. Under the assumptions that 1) gestures are readily learned if similar tasks are assigned to similar gestures, but that 2) learning is hindered if disparate tasks are assigned to similar gestures, Long et al. [86] developed a model for assessing the similarity of gestures with the goal of helping designers choose gestures that are both easy to learn and remember. Unfortunately the model was never verified with respect to the stated goals of learnability and memorability. My own goals are to understand how well users can learn and execute a small thumb-based gesture language for generalized one-handed use of a touchscreen-based mobile device.

3.2 Thumb Movement Study

Although the forms and input technologies of mobile devices steadily evolve, we as users of these devices remain much the same. We are quite adaptable *intellectually*, eager to learn how our devices can make us more efficient, more productive, more socially connected, and more entertained. Yet we humans have relatively fixed physical and mental capabilities, which place practical limits on the extent to which we can safely integrate mobile device use into our lives. These human constraints are made apparent in such diverse situations as using the buttons of a device, which can become dangerous with overuse because of the risk of repetitive strain injury [6], and talking on the phone while driving [142], which is dangerous because too many of our mental resources are reallocated from the task of driving to the task of talking.

In recognition of these types of human limitations, it seemed appropriate to try to understand how one-handed operation of a mobile device might be impacted by the bio-mechanic capabilities (or otherwise constraints) of the hand and thumb. Though the thumb is highly versatile in its range of motion, it is most adapted for grasping tasks, playing opposite the other four fingers [23]. Hence thumb interaction on the surface of a mobile device when held in the same hand imposes non-traditional movement and exertion requirements for the thumb—repetitive pressing tasks issued on a plane parallel to the palm. Given the possibility that one-handed use of a device requires the thumb to be used in a non-optimal manner, we presumed that variations in device size, interaction location, or thumb movement direction might each affect ease of use unequally. Thus, by developing an understanding of the impact these factors have on user task performance,

we hoped to suggest design guidelines for the placement of software (or hardware) targets for one-handed mobile interfaces.

Because no research had yet been conducted with these goals in mind, I designed and conducted a user study to generate the data needed to understand how device form and interaction characteristics influence thumb mobility. We focused on tapping tasks because tapping (or button pressing) is the predominant interaction method for keypad-based devices, and tapping is used almost exclusively for target selection, albeit with a stylus, on touchscreen-based devices. We hypothesized that the difficulty of a tapping task would depend on device size, movement direction, and surface interaction location. We captured the impact of these factors on user performance by measuring task completion speed, under the assumption that harder tasks would be performed more slowly than easier tasks.

I was the sole investigator on this study, which was conducted in April and May of 2005, and supervised by my advisor. The equipment used to gather data was generously provided by the Department of Kinesiology and coordinated by Dr. José Contreras-Vidal. The results of this work have been published as part of a chapter in the *Handbook of Research on User Interface Design and Evaluation for Mobile Technology* [75].

3.2.1 Equipment

Device Models

For real devices, design elements such as buttons and screens communicate to the user the "valid" input areas of the device. We instead wanted outcomes of task performance to *suggest* appropriate surface areas for thumb interaction. We identified four common

handheld devices to represent the range of sizes and shapes found in the marketplace: 1) a Siemens S56 candy bar phone measuring 4.0 x 1.7 x 0.6 in (10.2 x 4.3 x 1.5 cm); 2) a Samsung SCH-i600 flip phone measuring 3.5 x 2.1 x 0.9 in (9 x 5.4 x 2.3 cm); 3) an iMate Smartphone measuring 4 x 2.0 x 0.9 in (10.2 x 5.1 x 2.3 cm); and 4) an HP iPAQ h4155 Pocket PC measuring 4.5 x 2.8 x 0.5 in (11.4 x 7.1 x 1.3 cm). These devices are shown in the top row of Figure 9, which we refer to hereafter as SMALL, FLIP, LARGE, and PDA. To remove the potential for hardware characteristics like button and screen placement to bias user performance, we created a 3D "blank-slate" model of each device, which captured the device form, but eliminated all superficial design features.

Figure 9. Thumb Movement Study: Devices used for the thumb movement study, chosen to represent a range of sizes and forms (top row), together with their study-ready models (bottom row).

The phone models were developed using Z Corp.'s (http://www.zcorp.com/) ZPrinter 310 3D rapid prototyping system. Although the device models were hollow, we

reintroduced weight to provide a realistic feel in the hand. Once "printed" and cured, the models were sanded and sealed to achieve a smooth finish (bottom row of Figure 9).

A grid of circular targets 1.5 cm in diameter was affixed to the surface of each device. Circles were used for targets so that the sizes would not vary with direction of movement [89]. The target size was selected to be large enough for the average-sized thumb, while also providing adequate surface coverage for each device. The grid dimensions for each device were: SMALL (2x5), FLIP (3x4), LARGE (3x7) and PDA (4x6), as shown in the bottom row of Figure 9.

Measurement

A typical measurement strategy for tapping tasks would involve a surface-based sensor to detect finger contact. Unfortunately, due to the number and variety of device sizes investigated, no technical solution was found to be as versatile, accurate[1] or affordable as required. Instead we used Northern Digital Inc.'s OPTOTRAK 3020 motion analysis system designed for fine-grained tracking of motor movement (shown as part of the study setup in Figure 10). The OPTOTRAK uses 3 cameras to determine the precise 3D coordinates of infrared emitting diodes (IREDs). Three planar IREDs attached to the surface of each device defined its local coordinate system, and a fourth IRED was included on each surface for redundancy (see Figure 9, bottom row). To capture thumb movement trajectories, two IREDs were affixed to the participant's thumb—one on the top and one on the side facing the cameras; the data recorded were the relative movement

[1] Touch sensors by Phidgets Inc. (www.phidgets.com) were initially wired to copper tabs associated with each target to capture surface touches. Unfortunately, we experienced an unacceptably high rate of data loss, potentially due to electrical interference, simultaneous touches caused by the side of the hand, and/or the low sampling rate of the Phidgets hardware. Thus the data generated from the touch sensors had to be disregarded in the final analysis.

trajectories of the thumb IREDs translated with respect to the local coordinate system of the device used. Diode positions were sampled at 100Hz, and the data were post-processed to derive taps from thumb minima.

Software

Data collection and experiment software were run on a Gateway 2000 Pentium II with 256 MB of RAM running Windows 98.

3.2.2 Participants

Twenty participants were recruited via fliers posted in the Department of Computer Science at the University of Maryland, College Park. The only criterion for acceptance into the study was that the participant be right-handed. We obtained institutional review board (IRB) approval to conduct the study, and all participants granted us their informed consent. Participants (15 male, 5 female) ranged in age from 18 to 35 years with a median age of 25 years. Participants received $20 for their time.

3.2.3 Tasks

Users performed reciprocal tapping tasks in blocks as follows. For SMALL and FLIP, trials were divided equally into two blocks. For the LARGE and PDA devices, trials were divided equally into four blocks. Trials were assigned to blocks to achieve roughly equal numbers of *distance* x *direction* trials, distributed evenly over the device surface. Trials were announced by audio recording so that participants could focus their attention fully on the device. Participants were presented with the name of two targets by number. For example, a voice recording would say “one and three”. After one second, a voice-recorded “start” was played. Participants then tapped as quickly as possible between the

two targets, and after five seconds, a "stop" was played, which signaled the completion of the trial. After a 1.5 second delay the next trial began. Trials continued in succession to the end of the block, at which point the user was allowed to rest as desired, with no user resting more than two minutes. Device and block orders were assigned to subjects using a Latin Square, but the presentation of within-block trials was randomized for each user.

3.2.4 Design

For each target on each device (SMALL, FLIP, LARGE, and PDA), users performed tasks of every *distance* (one or two circles) and *direction* (↕,↔,⤢,⤡) that could be supported by the geometry of the device. For example, SMALL could not accommodate trials of distance-2 circles in the directions (↔,⤢,⤡). Note that the grid layout results in *actual* distances that differ between *orthogonal* trials (↕,↔) and *diagonal* trials (⤢,⤡), which I consider separately in the analysis. For LARGE and PDA, trials of distance-4 circles were included as the device geometry permitted. Finally each device included a ⤢ and ⤡ trial to opposite corners of the target grid. For each device, a small number of trials (1 for SMALL, LARGE and PDA, 3 for FLIP), selected at random, were repeated so as to make the total trial count divisible by four. The resulting number of trials for each device were: SMALL (32), FLIP (48), LARGE (108), and PDA (128). Larger devices required more trials because they had more surface targets to test.

3.2.5 Procedure

Each session began with a brief description of the tasks to be performed and the equipment involved. Two IRED markers were then attached to the participant's right thumb with two-sided tape. One diode was placed on the leftmost edge of the thumb nail,

and a second on the left side of the thumb. The orthogonal placement was intended to maximize visibility of at least one of the diodes to the cameras at all times. The two marker wires were tethered loosely to the participant's right wrist with medical tape.

The participant was seated in an armless chair, with the device held in the right hand, and the OPTOTRAK cameras positioned over the left shoulder. A mockup of the study setup is shown in Figure 10. The participant was then given more detailed instruction about the tasks to be performed, and was informed of the error conditions that might occur during the study: if at any point fewer than three of the device-affixed IREDs or none of the thumb IREDs were visible to the cameras, an out-of-sight error sound would be emitted. The participant was instructed to continue the trial as naturally as possible upon hearing the error sound, but should attempt to make postural adjustments to improve the diode visibility to the cameras. Next, the participant performed a practice session of 24 trials using the first device; the trials were selected to represent a variety of distances, directions, and surface locations. During the practice trials, the administrator intentionally occluded the diodes to give the participant familiarity with the out-of-sight error sound and proper remedies. After completion of the practice trials and indication that the participant was ready, the study proper began.

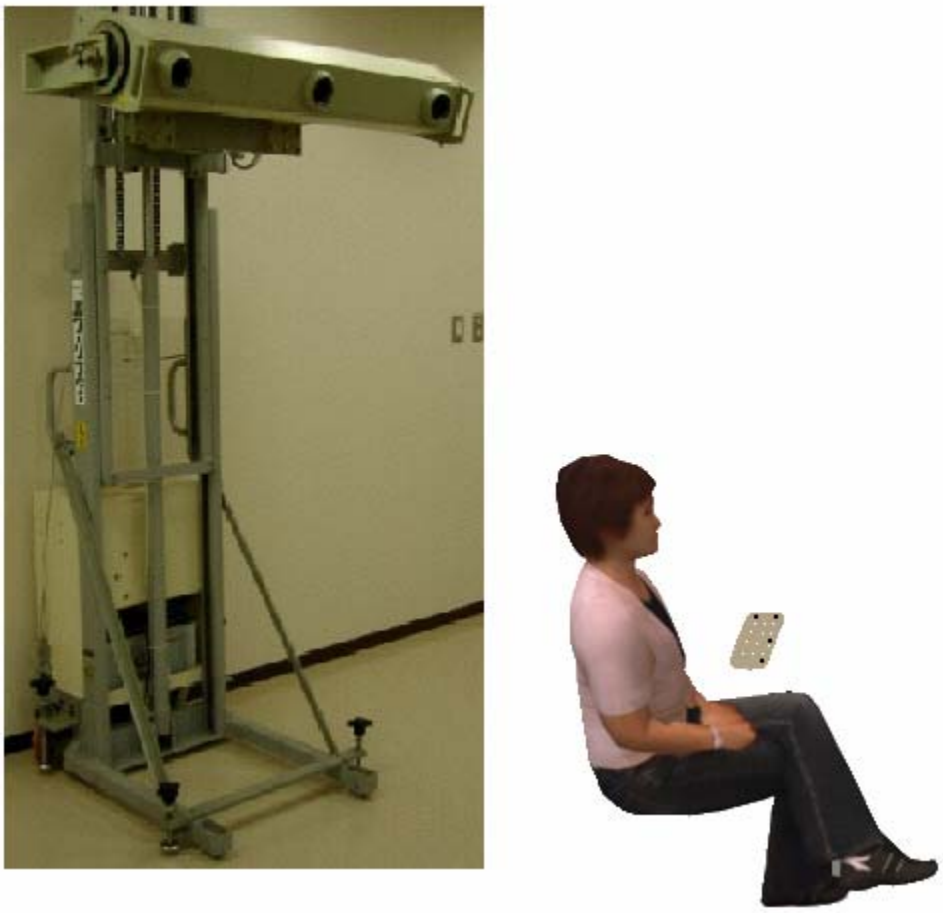


Figure 10. Thumb Movement Study: A mockup of the study equipment and user setup.

During the study, participants were allowed to hold the devices in whatever manner supported their best performance. Since the instructions were presented audibly and with a short pause before each trial began, users could prepare the position of their grips if desired. We chose not to control for grip in our study under the reasoning that it more accurately resembles real world settings, in which users have the freedom to adjust their grips to best suit the environment and task.

After all trials for a device were completed, users were allowed to rest while the next device was readied, typically on the order of three to five minutes. We hoped that the combination of the rest periods and the counterbalanced device order would minimize

as much as possible the impact of fatigue on our results. After completing all trials for the last device, the participant completed a questionnaire, where she recorded demographics and subjective ratings. Total session time was two hours, approximately an hour of which was devoted to data collection.

3.2.6 Measures

The raw 3D thumb movement data for each five-second trial was truncated to the middle three seconds to eliminate artifacts resulting from initiation lag and anticipated trial completion, phenomena routinely observed by the administrator. In a post processing phase, taps were identified within the remaining three-second interval and a single average tap time was computed from the difference in time between the onset of the first tap to the onset of the last tap, divided by one fewer than the total number of taps detected. In the post-experiment questionnaire, participants assigned an overall rating of difficulty to each device (1-7, where 1 = easy, 7 = difficult), and indicated the device regions that were both easiest and hardest to interact with.

Data Post Processing

Since the 3D thumb position (*x,y,z*) was recorded relative to the device surface, the *z*-value represented the thumb height above the device. While one might assume that taps were those thumb positions for which the *z*-distance was 0, the IREDs were mounted on participants' thumbnails, and so never actually reached the surface of the device. Taps were instead defined as points when both the *z*-value and change in *z*-value (velocity) were minimal. For example, Figure 11 shows a plot of one trial's *z*-values over time; the valleys of the resulting wave-like pattern indicate the times when the participant tapped the surface of the device.

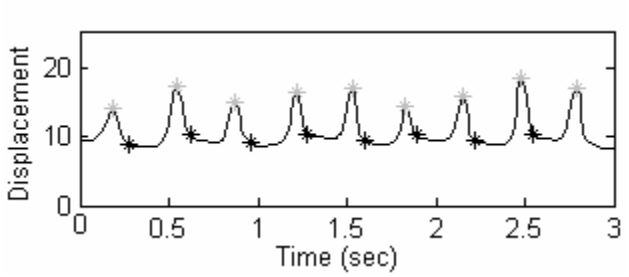


Figure 11. Thumb Movement Study: An example plot of thumb distance from the surface of a device over the course of a trial. Grey and black stars depict MATLAB's auto-detected peaks and valleys respectively.

The raw data was first preprocessed to extract the middle three seconds of each trial as well as to select whichever of the two thumb diode streams had the most complete data set (e.g., the fewest number of missing frames, or if equal, the one with the smallest windows of missing frames). Linear interpolation was performed on missing frames if the gap was less than 100 ms. Missing frames included those lost due to out-of-sight errors, as well as occasional frames dropped by the collection hardware.

The data were then analyzed by the PICKEXTR MATLAB function to identify extrema in a signal. This function is provided with the RelPhase.Box MATLAB toolbox for relative phase analysis of oscillatory systems [43]. The accuracy of the tap classifier was verified by inspecting a visual representation of each trial (Figure 11). When required, manual corrections were made using the following rules: 1) valid endpoints were always preserved; 2) if intermediate taps were missing, they were added; 3) if intermediate taps were incorrect, they were recoded by hand; and 4) if endpoints were invalid, the entire signal was coded by hand. Since average tap time was calculated as the number, not placement, of intervening taps, this method minimized as much as possible the bias of human annotation. Of the trials included for statistical analysis, 1.3% were discarded because they could not be encoded by machine or human, or had less than 1.5 seconds of encodable signal.

3.2.7 Results

The goal of our analysis was to understand whether user performance was influenced by device size, task region, and movement direction. To enable comparison among the devices, we limited the analysis to trials with distances of one or two circles since the geometries of all but the smallest device (SMALL) supported trials of these distances in all four movement directions. To address the fact that actual movement distance differed between orthogonal and diagonal trials, we analyzed these groups separately. For all analyses, Huynh-Felt corrections were used when the sphericity assumption was violated, and Bonferroni corrections were used for post hoc comparisons.

Movement Direction

A 2 (*distance*) x 2 (*direction*) repeated measures analysis of variance (RM-ANOVA) was performed on mean task time data for both orthogonal trials (distances: 1, 2; directions: ↕,↔) and diagonal trials (distances: 1.4, 2.8; directions: ⤢,⤡) for the three largest devices. Since SMALL did not support distance-2 trials in all four directions, a one-way RM-ANOVA was performed on mean task time for trials of distance-1 and distance-1.4.

SMALL: A main effect of direction was observed for diagonal trials ($F(1,19) = 65.1$, $p < .001$). Post hoc analyses showed that trials in the ⤢ direction were performed significantly faster than those in the ⤡ direction (0.26 v. 0.28 ms, $p < .001$).

FLIP, LARGE, and PDA: Results were similar across the analyses of the three largest devices. Unsurprisingly, a main effect of distance was observed for both orthogonal and diagonal trials, with shorter trials performed significantly faster than longer trials. There were no further effects of direction or interaction between direction and distance for orthogonal trials (↕,↔). However, for diagonal trials, a main effect of

direction was observed, with trials in the ⤢ direction performed significantly faster than those in the in ⤡ direction for all devices. In addition, a *distance* x *direction* interaction showed performance differences between the diagonal trials were more pronounced for longer trials than for shorter trials (Table 1).

Table 1. Thumb Movement Study: Mean movement time for *direction* and *distance×direction* for the FLIP, LARGE, and PDA devices.

	Direction (⤢ v. ⤡)	$F_{1,19}$	p	Distance x Direction (⤢ v. ⤡)	$F_{1,19}$	p
FLIP	.31 v. .35 ms	50.5	<.001	(1.4) .27 v. .30 ms (2.8) .34 v. .41 ms	14.6 34.5	< .001 < .001
LARGE	.31 v. .36 ms	46.1	<.001	(1.4) .27 v. .28 ms (2.8) .35 v. .42 ms	28.0 44.2	< .001 < .001
PDA	.32 v. .36 ms	46.5	<.001	(1.4) .28 v. .30 ms (2.8) .36 v. .43 ms	23.0 38.0	< .001 < .001

Device Size

To determine if device size impacted comparable tasks across devices, we analyzed all trials performed in the lower right 3x4 region of the three largest devices using a 3 (*devices*) x 43 (*trials*) RM-ANOVA. While a main effect of trial was observed, this was expected, as trials of every distance and direction were included for analysis. Yet neither a main effect of *device* nor an interaction between *device* and *trial* was found.

Target Location

To determine if target location affected performance, we analyzed task times for the shortest (distance-1) tasks for each device. We chose short tasks because they provide high granularity for discriminating among device locations. Since direction was shown to affect task time for diagonal trials, only orthogonal tasks could be considered. For each device, a one-way RM-ANOVA was performed on mean trial time. The number of trials

analyzed varied for each device because they each supported different numbers of distance-1 trials.

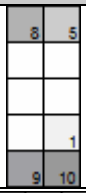
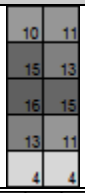
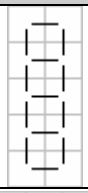
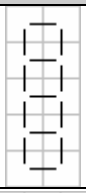
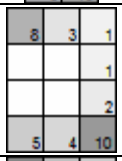
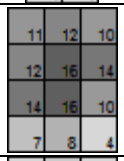
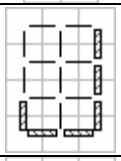
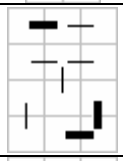
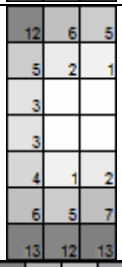
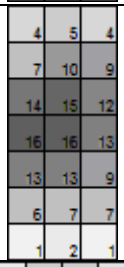
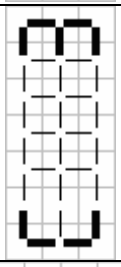
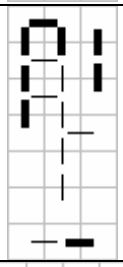
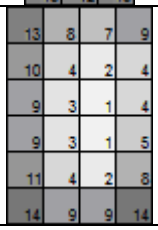
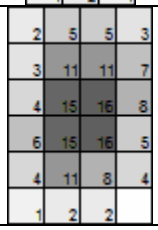
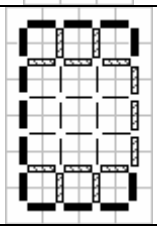
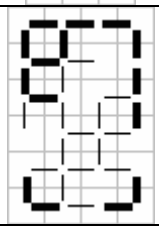
A main effect of target location was observed for SMALL ($F(8.6, 163.3) = 2.1$), $p = .032$), FLIP ($F(11.5, 218.4) = 3.5$, $p < .001$) and PDA ($F(9.8, 188.1) = 3.9$, $p < .001$), but not for LARGE. Since it is difficult to derive helpful high-level conclusions by comparing large numbers of trials pair-wise, we explored two aggregation techniques.

Subject-derived regions. Based on subjective opinion of which regions were easiest to interact with for each device, we divided tasks into three groups (E)asy, (M)edium, and (H)ard. Tasks for SMALL and FLIP were assigned to only E and M groups. A one-way RM-ANOVA on mean group task time was performed for each device. A main effect of group was found for FLIP ($F(5.5, 105.1) = 11.3$, $p < .001$), LARGE ($F(3.1, 58.7) = 8.4$, $p < .001$), and PDA ($F(4.8, 91.0) = 22.0$, $p < .001$). Post hoc analyses using Bonferroni correction revealed that all groups differed significantly from each other for FLIP and PDA, where group E was faster than M, which was in turn faster than H. For LARGE, E and M were significantly faster than H, but were indistinguishable otherwise, so we collapsed them to E (Table 2, third column).

Data-derived regions. For each device we ordered tasks by mean tap time, and then segmented them into seven groups. If the number of trials was not divisible by 7, the remainder trials were included in the middle group. A one-way RM-ANOVA on mean group task time was performed for each device. A main effect of group was found for FLIP ($F(5.5, 105.1) = 11.3$, $p < .001$), LARGE ($F(3.1, 58.7) = 8.4$, $p < .001$), and PDA ($F(4.8, 91.0) = 22.0$, $p < .001$). From these results, groups were labeled *fastest* and *slowest* such that all groups in *fastest* were significantly faster than all groups in *slowest*,

according to post hoc analyses using Bonferroni correction. Trials in these groups are shown visually in the rightmost column of Table 2. Mean task time for *fastest* v. *slowest* trials for each device were FLIP (0.26 v. 0.28 ms), LARGE (0.25 v. 0.28 ms), and PDA (0.26 v. 0.29 ms).

Table 2. Thumb Movement Study: Preference and movement time maps for each device type studied. Depth of color in columns 1 and 2 indicate stronger user agreement.

	Subjective Difficult Regions	Subjective Easy Regions	Preference-Derived Regions Easy Medium Hard	Data-Derived Regions Fastest Slowest
SMALL				
FLIP				
LARGE				
PDA				

Subjective Preferences

After completing all trials, participants were presented with diagrams of each device similar to those shown in the first two columns of Table 2 and were asked to identify the targets they found most easy and most difficult to interact with. Aggregating results

across users yielded a preference "map" for the least and most accessible targets of each device (columns 1 and 2 of Table 2), with darker regions indicating more agreement among participants. We see that for each device (e.g., row in Table 2), the representations in the first two columns (difficult v. easy to reach location) are roughly inverses of one another.

In addition to region marking, we asked users to rate the overall difficulty of managing each device with one hand on a 7 point scale (7 = most comfortable). Average ratings from most to least comfortable were as follows: SMALL (6.4), FLIP (5.4), LARGE (4.1) and PDA (3.0).

3.2.8 Discussion

The findings from our analysis of thumb movement suggest the following guidelines. First, thumb movement in the ↘ direction is difficult for right-handed users regardless of device size. Presumably the difficulty arises from the considerable thumb flexion required to perform these types of tasks. Under this reasoning, the opposite movement ↙ would be difficult for left-handed users, so conservative designs should constrain repetitive movement to the ↕ and ↔ directions, especially for repetitive tasks such as text or data entry.

Second, device region affects both task performance and perceived difficulty. Not only did the slowest trials correspond to those regions users found most difficult to interact with, but fastest trials also matched those regions users found most easy to interact with. As an example, notice how the darkest (perceived easiest) regions in the second column of Table 2 correspond to the easiest/fastest subjective groups in column 3, and fastest groups in column 4. In general, regions within reach of the thumb were fastest

and most comfortable, favoring those toward the middle of the device. We conjecture these regions lay within a "sweet spot" that is both easy to reach and easy to interact with, requiring movement primarily from the base of the thumb. The lower right corners of the devices do not fit this characterization because they are biomechanically awkward to reach owing to the fact that they are "too close" rather than "too far".

Because the absolute time differences between the fastest and slowest regions of the devices were quite small (at most 30 ms), we do not think performance speed is the main concern in forming design recommendations from these data. Rather, it is the fact that the speed differences between the regions were statistically significant (e.g., 7%-12% slowdown between the fastest and slowest regions), which suggests that mechanical or physical limitations were to blame. The data, therefore, are concerning primarily from an ergonomics perspective. In fact, we believe that the slowdowns we found should be thought of as optimistic, since they capture only localized movement and required substantial changes in user grip between tasks; subjective opinion, user observations and practical experience indicate that designers should be cautioned against using the entire device surface for thumb interaction, especially for larger devices. We instead recommend placing interaction objects centrally to accommodate both left and right handed users, or offering configurable displays. Since hand size and thumb length will differ by individual, designs should strive to support a range of users.

More generally, designs should strive to support localized interaction by placing targets close to one another, with the goal of minimizing grip adjustments. This recommendation is drawn from three informal observations: 1) that participants adjusted their grips between trials to accommodate tasks at different surface locations; 2) that

participants had difficulty maintaining control of the device when tasks involved large movement distances; and 3) that these behaviors were especially noteworthy when participants were interacting with the larger devices (LARGE and PDA). Yet the trail time data showed that users performed relatively consistently across devices and surface locations when performing short-distance tasks. For example, on average, users took between 242 ms and 306 ms to move between adjacent targets, across all devices and all locations. This narrow range of time results suggests that, in general, users are able to control and interact with all devices reasonably well as long as the interaction locations are close to one another.

Finally, the finding that users performed trials in the lower right 3x4 sub-grid of the three largest devices equally well suggests that holding a large device does not inherently impede thumb movement. Rather, larger devices simply have more surface regions that are difficult to reach with the thumb, and so have more regions that are inappropriate for object placement in one-handed designs. Together with user opinion that larger devices were more difficult to manage suggests that the current trend toward smaller device forms benefits one-handed device operation and control.

3.3 Target Size Study

Current touchscreen interfaces are composed of widgets similar in size and function to those featured on the desktop. While this is acceptable for interaction with a stylus tip, which is typically smaller than 1 mm in diameter, it means that most widgets are much smaller than the average thumb pad in at least one dimension, making reliable finger access difficult or impossible. This problem is similar to the one observed by University of Maryland researchers when studying touchscreens designed for index-finger

interaction on desktop sized displays [116]. They noted that users became frustrated with interaction designs that executed a selection as soon as the finger landed on the screen, since parallax and calibration problems caused a high incidence of error. They found that offsetting a cursor from the finger and instituting a lift-off selection strategy allowed users to reliably select targets as small as a single character.

The problem with using an offset cursor as a general strategy for finger interaction on a PDA, however, is that precisely positioning a cursor is prohibitively time-consuming for tasks like text entry that require tens to hundreds of successive selections. Instead, providing targets large enough to be reliably selected would be more useful. But because touchscreen widgets compete with other information for limited screen space, it is desirable to keep the dimensions of interaction targets as small as possible without degrading user performance or satisfaction. Unfortunately, results from previous studies cannot offer guidance because none have looked at the specific question of one-handed touchscreen interaction with the thumb. Inspired by the task-based performance differences found by Colle and Hiszem [36] for index finger use on desktop sized touchscreens, we conducted a two-part study to investigate the interaction between target size and performance for thumb-based mobile touchscreen use, considering first single-target (*discrete*) and then multi-target (*serial*) tasks.

In the Fall semester of 2005, I managed the research of a Finnish student visitor, Pekka Parhi. Our goal was to develop recommendations for software target sizes on touchscreen-based mobile devices that supported one-handed thumb operation. I developed the detailed study design, and worked closely with Pekka during his implementation, administration, and analysis of the study. We co-authored a paper that

summarizes this work, and which has been published in the proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI) 2006 [112].

3.3.1 Procedure

The study was divided into two phases—one for single-target (*discrete*) tasks, and the other for multi-target (*serial*) tasks. After completing an initial questionnaire to collect demographics and prior device use, the participants performed the discrete target phase followed by the serial target phase. Each phase began with a practice session, which consisted of one block of trials, and which was followed by the five official test blocks. After each phase, participants recorded subjective ratings of the interaction experience. The total session time was approximately 45 minutes.

3.3.2 Participants

Twenty participants (17 male, 3 female) were recruited via e-mail announcement and fliers posted in the Department of Computer Science at the University of Maryland, College Park, with the only restriction that participants were right-handed. We obtained institutional review board (IRB) approval to conduct this study, and all participants granted us their informed consent. The age of the participants varied between 19 and 42 years, with a mean of 25.7 years. Hand width and thumb length were recorded for each participant. Thumb length varied between 99 and 125 mm (μ=115 mm, σ = 5.75), and hand width varied between 75 and 97 mm (μ=88 mm, σ = 6.08). Participants received $10 for their time.

3.3.3 Equipment

Both phases of the experiment were performed on an HP iPAQ h4155 Pocket PC measuring 4.5 x 2.8 x 0.5 in (11.4 x 7.1 x 1.3 cm) with a 3.5" screen, measured diagonally. The display resolution was 240x320 pixels with a 0.24 mm dot pitch. The study interface and control software was developed using the Piccolo.NET graphics toolkit [17].

3.3.4 Discrete Target Phase

The goal of the discrete target phase was to determine size recommendations for widgets used for single-target tasks, such as activating buttons, radio buttons and checkboxes. This discrete phase used a 5 (target sizes) x 9 (locations) x 5 (repetitions) within subjects design. Target sizes were 3.8, 5.8, 7.7, 9.6 and 11.5 mm on each side. We performed pilot studies to determine the appropriate target sizes for the study. Since standard widget sizes range from 2.64 mm (radio buttons) to 4.8 mm (buttons), 3.8 mm represents an average target size for existing devices. Pilot studies indicated that performance rates leveled off for target sizes greater than 11.5 mm and thus represented the largest practical recommended size for singular targets.

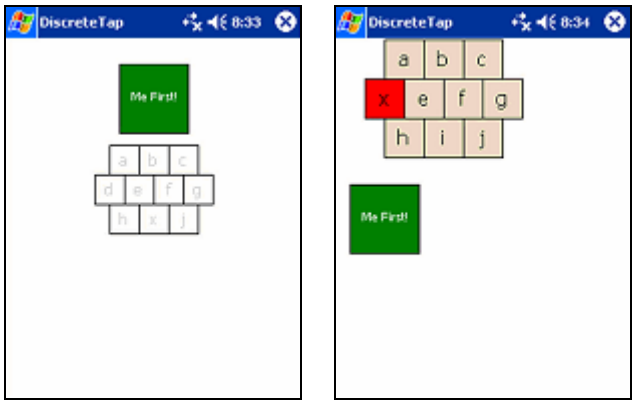
Nine target locations were defined by dividing the display into a 3x3 grid of equal-sized cells. For each trial the target was located in the center of one of the nine cells. Each target size (5) was tested 5 times in each of the 9 regions for a total of 225 trials. Trials were distributed across 5 blocks. With the first five participants, the sizes and locations of the targets were accidentally randomized across all blocks, but after minor modifications to the software for both phases, the sizes and locations of the targets

were randomized within each block to ensure that each size x location combination was tested once per block.

Tasks

The participant's task for each discrete target trial was to tap an initial start position and then the target to be selected. All tasks were performed standing and one-handed, using only the right hand thumb for interacting with the touchscreen. The participants were instructed to perform the tasks as naturally as they could, favoring accuracy to speed.

For each trial, the start position was indicated by a large green button designed to be easy to select, but from which movement distance could be measured (Figure 12). The distance between the green button and the target was constant for all tasks, while the relative location of the green button varied depending on the region in which the target was positioned. To standardize movement direction across trials, the green start button was located either directly North or South of the target, so chosen because North-South (↕) movement better matches the thumb's natural axis of rotation than East-West (↔) movement when holding a device in one hand. If the target was located in the first row of the grid, the green button was located in the cell below the target. Otherwise, the green button was located in the cell above the target.



(a) (b)

Figure 12. Target Size Study: The experimental interface for the discrete target phase. The startup view for a trial testing a 5.8 mm target in the center zone (a). The display for a trial in the upper left zone as the user selects the 7.7 mm target (b).

Two issues arose in the design of the tap target. First, our pilot studies indicated that lone targets were perceived to be easier to tap than those near other objects. To address this issue, we surrounded each intended target by 'distractor' targets. This meant participants were required not only to hit a target, but also avoid others. In addition, the design provided an interface closer to real world applications which often present multiple widgets close to each other instead of one single target on the screen. Our second concern was that the constant distance between each start location and the task target meant that users could conceivably adopt a routine or preprogrammed movement for task completion rather than via deliberate aiming. Here, too, the distractor targets were of value. Although the relative position of the target with respect to the start position never changed, the distractors were presented in randomized locations around the target, which promoted a sense that the participant was not moving the same exact distance and in the same direction for each trial.

In each trial, the intended target was designated with an 'x', while the distractors were labeled with other alphabetic characters. At the start of a trial, the target and all distractors were displayed with a white background and light-gray lettering, so as to

deemphasize the target, and to discourage users from locating the target preattentively before the start of the trial (Figure 12a). When the start button was tapped and released, labels turned black and keys turned pink to draw attention to all on-screen objects (Figure 12b).

Target selections were recorded at the time and location that a user's thumb was lifted from the screen. A successful target selection required that both the tap and release positions were located within the target area. Target taps could also be cancelled by dragging the thumb outside of the target area before the release, similar to the method that allows for canceling widget actions on the Pocket PC.

To ensure visual search was not impacted by the variability of white space surrounding labels as targets changed size, font sizes were scaled with target sizes. Because of limited screen space and evidence that performance is unaffected by key spacing (e.g., [36]), we used 0 mm edge-to-edge spacing between targets and distractors. Participants were provided with both auditory and visual feedback when touching targets. The 'x' target was highlighted in red upon thumb contact (Figure 12b), and both success and error sounds were played upon thumb release to indicate whether the target was hit successfully or not. If a tap was cancelled no auditory feedback was given.

Measures

Application logs recorded the time between the start (first) tap and target (second) tap, the absolute position of the second tap, and trial success or failure. After completing all trials, the participants were asked to rate how comfortable they felt tapping the target 'x' in each region of the screen using a 7-point scale (1 = uncomfortable, 7 = comfortable), as well as which target size was the smallest they felt comfortable using in each region.

3.3.5 Serial Target Phase

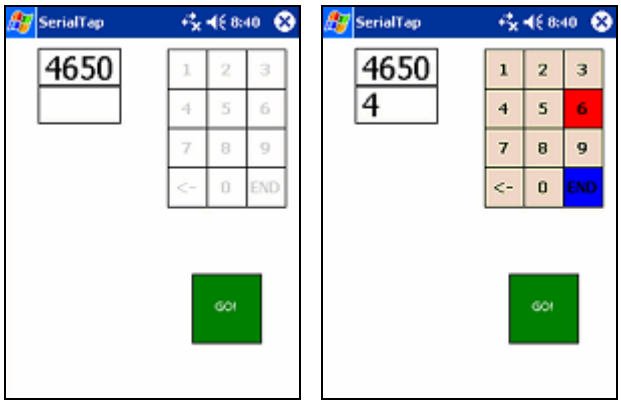
The goal of the serial target phase was to evaluate required key sizes for widgets used for text or numeric entry. Target sizes were 5.8, 7.7, 9.6, 11.5, and 13.4 mm with 0 mm edge-to-edge spacing. Target sizes were similar to those of the discrete target phase, except due to previous findings that error rates tended to increase for sequential selections [36], the smallest target (3.8 mm) was removed and an even larger target (13.4 mm) was added. To study the effect of location on task performance, four regions were defined by dividing the screen into a 2x2 grid.

Each of the target sizes (5) were presented 5 times in each of the 4 regions for a total of 100 trials. As in the discrete target phase, trials were divided into 5 blocks. Except for the first 5 subjects who received all trials randomized across all 5 blocks, each size x location combination was presented once per block, in randomized order.

Tasks

The serial target task design was based on tasks used for previous studies [24, 36]. Subjects were required to enter a series of four digit codes using a virtual numeric keypad. They performed the tasks with the thumb of the right hand while standing, as in the discrete target phase.

For each task, a green 'start' button, a numeric keypad and a randomly-generated 4-digit goal sequence were displayed. Backspace and 'END' keys were also presented in the bottom corners of the keypad (Figure 13). Since the keypad's location varied from trial to trial, the remaining interface elements were repositioned as follows: the green 'start' button was positioned in the cell above or below the keypad, and the 4-digit goal sequence appeared to the left or right of the keypad.



(a) (b)

Figure 13. Target Size Study: The experimental interface for the serial target phase. The startup view for a trial testing a keypad with 7.7 mm targets in the upper right zone (a). The display for the same trial as the user selects the second digit in the sequence, which is a 6 (b).

The participant's task was to tap the green button first, enter the target sequence with the keypad, and finally touch the 'END' key to confirm the entry and proceed to the next task. The input string was displayed directly below the target sequence. The backspace key could be used for corrections; however it was not necessary for users to input the correct number before moving on—only that they input 4 digits.

Several interaction features were retained from the discrete target phase. After tapping the green 'start' button, the background of the keypad changed from white to pink and the labels from light gray to black (Figure 13). Here, too, font sizes adapted to changes in target size. Finally, visual and audio feedback were provided upon target selection. The success sound was played for all key hits, except in the event that the 'END' key was selected before all numbers had been entered, or a numeric key was selected after all four digits had been entered; in these cases an error sound was played. Again a lift-off strategy was used for selection.

Measures

Application logs recorded total task time from the release of the start button to the release of the 'END' button, the first transition time from the release of the start button to the release of the first keypad button, and the first transition distance. Errors were recorded similarly to Sears et al. [130], where uncorrected errors were recorded by comparing the target and input sequence, and corrected errors by counting the number of backspace presses during the trial. After completing all trials, participants were asked to rate how comfortable they felt using the keypad in each region of the screen using a 7-point scale (1 = uncomfortable, 7 = comfortable), and which keypad size was the smallest they felt comfortable using in each region.

3.3.6 Discrete Target Phase Results

Task Times

Task time, defined from the release of the start button to the release of the target 'x', was analyzed using a 5 x 9 repeated measures analysis of variance (RM-ANOVA) with factors of target size (3.8, 5.8, 7.7, 9.6 and 11.5 mm) and location (9 regions derived from a 3x3 division of the screen). Erroneous trials were eliminated from the data set and the mean total time of the remaining trials was computed. A 5% level of confidence after Greenhouse-Geisser correction was used to determine statistical significance. A main effect of target size, ($F(1, 25) = 70.42$, $p < .001$) was observed. No other main effects or interactions were observed.

Not surprisingly, as targets grew in size, participants were able to tap them faster (Figure 14a). Post hoc comparisons using Bonferroni corrections revealed that time differences between all target sizes were significant, even between the two largest target

sizes (p = .04). These results are consistent with Fitts' Law [50]. Due to the small screen size and limited practical range of target sizes in this study, the values for the indices of difficulty (IDs) were small, and the range was narrow. While these conditions make our study inappropriate for offering official values for *a* and *b* in the Fitts model, Figure 14b shows that differences between the task indices of difficulty explain why tap times got faster as target sizes increased.

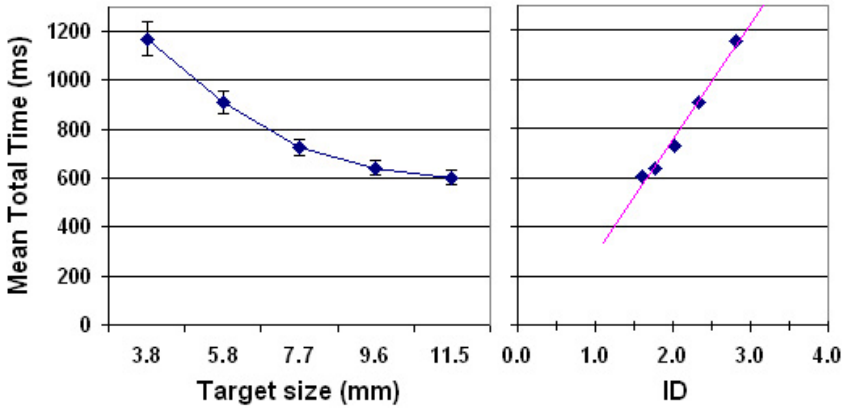


Figure 14. Target Size Study: The mean task time between the release of the start button and release of the target 'x' for each target size in the discrete target phase (left), and the relationship between movement time and task index of difficulty (right).

Error Rates

A 5 x 9 repeated measures analysis of variance (RM-ANOVA) was carried out on the percentage of trials that were performed in error, with factors of target size (3.8, 5.8, 7.7, 9.6 and 11.5 mm) and location (9 regions derived from a 3x3 division of the screen). Once again, a main effect of target size was observed ($F(1, 27) = 49.18$, $p < .001$), but no effects of target location nor interactions between target size and location were found.

As shown in Figure 15, errors declined as target size increased. Post hoc comparisons using Bonferroni corrections showed that error rates for the two smallest targets differed significantly from one another, and were significantly higher than for all other targets. Also, participants made significantly more errors when aiming for the mid-

sized target (7.8 mm) than the largest target (11.5 mm). However, there was no significant difference in error rate between the two largest targets (9.6 mm v. 11.5 mm). So while speed improves significantly as targets grow from 9.6 mm to 11.5 mm, error rate does not.

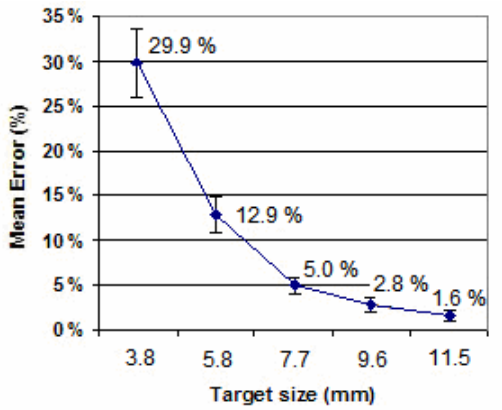


Figure 15. Target Size Study: The mean error rate for each target size in the discrete study phase.

Hit Distribution

Several investigations into target size requirements have used actual selection location to derive recommendation for on-screen targets. Since error rate was not distinguishable between the two largest targets, Figure 16 displays the on-screen hit distribution for the smallest four targets in all nine screen locations. The nine solid white boxes in each figure indicate the valid hit zones, with the center shown as a black crosshair. Taps that fell within valid bounds are shown as gray dots, and erroneous hits are shown in black. The dark gray outline near each zone center encloses all hits that fell within 2 standard deviations (2-SD) of the means in both the X and Y directions.

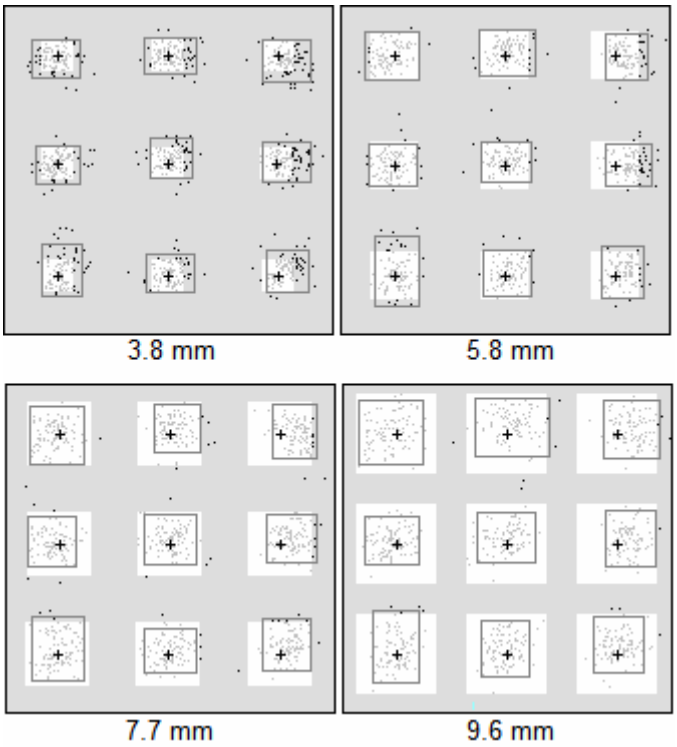


Figure 16. Target Size Study: The actual tap locations for targets sized 3.8, 5.8, 7.7, and 9.6 mm. The white squares indicate the true targets and the black crosshairs indicate their centers. Gray dots indicate successful hits, black dots indicate unsuccessful hits, and gray bounding boxes indicate hits that fall within 2 standard deviations of the tap means in the X and Y directions.

The maximum width and height across the 2-SD bounding boxes for each target was 6.5x6.5 mm, 7.0x8.6 mm, 6.7x7.9 mm, and 9.1x8.9 mm for targets sized 3.8 mm, 5.8 mm, 7.7 mm, and 9.6 mm respectively. These measures offer the minimum sized box that would be expected to enclose 95% of hits at any screen location. We see that in general, the total area of these boxes increases with target size, indicating users were indeed trading off speed for tap accuracy. If we consider the relative shape and position of the 2-SD bounding boxes with respect to the true target centers, we notice some trends along rows and columns. For example, the hits in the bottom row tend to fall above the target center. This trend does not seem to be due only to the direction of movement, since targets in the middle row were also approached from above, and yet hits for those targets tend to fall more centrally than for those in the bottom row. Considering trends across

columns, we see that hits along the rightmost column tended to fall to the right of the target center, even though movement direction was from either directly above or below.

Subjective Preferences

Participants were asked how comfortable they felt tapping targets in each of the 9 regions, regardless of target size (7 point scale; 1 = uncomfortable, 7 = comfortable). Mean ratings for comfort level are shown in the upper left corner of each region of Figure 17a, and the darker the region, the more comfortable users found it to be for target selection. The center region was considered the most comfortable (μ=5.7), while the NW and SW regions were rated as the least uncomfortable locations for discrete target interaction with the thumb (both with μ=3.7).

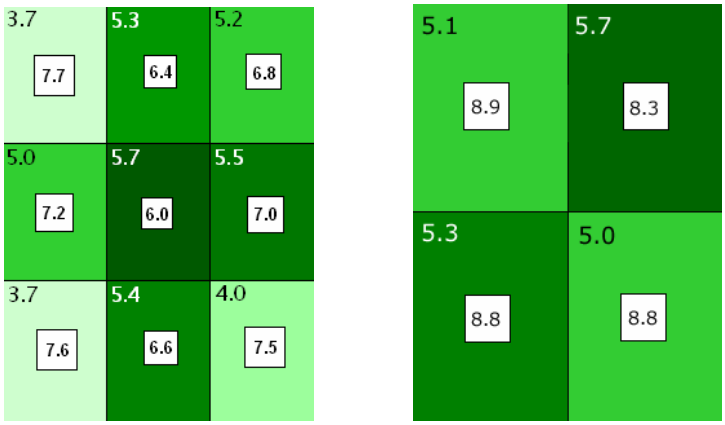


Figure 17. Target Size Study: Subjective ratings for interacting with discrete targets in 9 regions (a), and serial targets in 4 regions of the device (b). Mean comfort rating (1-7; 7=most comfortable) is shown in the upper left corner of each region, and depth of background color indicates more comfort. White blocks in each cell indicate the mean size, in mm, of the smallest comfortable target in the region.

Participants were also asked which was the smallest of the 5 target sizes they felt comfortable tapping in each region. Mean target sizes are shown as white blocks in each of the nine regions in Figure 17a. Overall, participants perceived they would be comfortable with smaller targets within the center column, and in the center region in

particular (μ=6.0 mm). Participants felt the largest targets would be required in the NW, SW, and SE corners of the display (μ=7.7, μ=7.6, and μ=7.5 mm respectively).

In general, the more comfortable participants were tapping targets in a region, the smaller they felt targets needed to be. Indeed, the subjective ratings correlate with performance results in Figure 16—across targets of varying size, corner regions tended to have larger 2-SD bounding boxes than the center regions. Even though user performance could not be discriminated statistically based on interaction region, the subjective preferences and hit locations indicate that users had the most difficulty interacting with objects along the left side and bottom right corner of the device and were at most ease interacting in the center of the device.

3.3.7 Serial Target Phase Results

Task Times

A 5 (target size: 5.8, 7.7, 9.6, 11.5, 13.4 mm) x 4 (locations: 4 regions derived from a 2x2 division of the screen) repeated measures analysis of variance (RM-ANOVA) was carried out on the task time data, defined from the release of the first digit in the sequence to the release of the 'END' button. Trials with either corrected or uncorrected errors were eliminated from the data set and the mean total time after the first transition of the remaining trials was computed. As with the discrete target results, a main effect of key size was observed, $F(1, 25)=60.02$, $p < .001$. Neither a main effect of keypad location nor an interaction between size and location were observed.

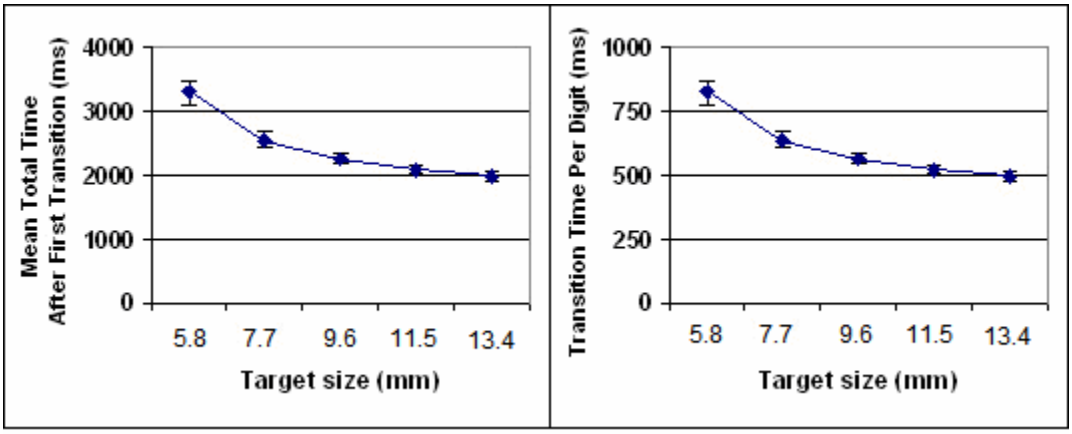


Figure 18. Target Size Study: The mean task time between the release of the first digit and the release of the 'END' key for each key size in the serial target phase (left), and the mean transition time between taps after the first transition for the same key sizes (right).

As shown in Figure 18, users were able to enter 4-digit sequences faster as the key sizes, and thus total keypad size, grew. Post hoc comparisons using Bonferroni corrections revealed that time differences between all key sizes were significant. However, in contrast to the results of the discrete phase, Fitts' Law does not explain this finding. Since the keypads used for each condition scaled uniformly, IDs remained equal across keypads of differing sizes. Under these circumstances, Fitts' Law would predict performance rates to be equal across conditions, yet we observed that performance improved as key sizes grew. One explanation for this finding is that finger size interacted with key size. Since all but the largest keys were sized smaller than the average thumb, users may have made intentional physical accommodations to increase accuracy such as reorienting the thumb, which would have slowed performance. Although our study was not specifically designed to understand this phenomenon, we hypothesize that the actions users take to accommodate touchscreen targets smaller than the thumb acts upon Fitts' model as if the target size is smaller than it actually is, thereby increasing total movement time.

Error Rates

A 5 (target size: 5.8, 7.7, 9.6, 11.5 and 13.4 mm) x 4 (locations: 4 regions derived from a 2x2 division of the screen) repeated measures analysis of variance (RM-ANOVA) was carried out on the percentage of trials that were performed in error. A trial was considered to be successful only if no errors, corrected or uncorrected, were made. A main effect of target size was observed ($F(2,43) = 11.83$, $p < .001$), but no main effect of keypad location was present. However, an interaction between key size and keypad location was observed ($F(12,228) = 1.87$, $p = .039$).

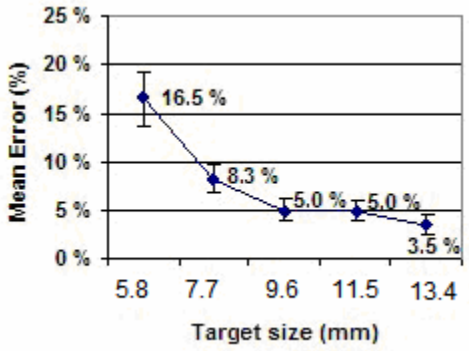


Figure 19. Target Size Study: The mean error rate for each key size in the serial target phase.

In general, errors declined as key size increased (Figure 19). Post hoc comparisons using Bonferroni corrections revealed the keypad with the smallest key sizes (5.8 mm) caused significantly more errors to be made than those with key sizes ≥ 9.6 mm. No differences between error rates for the other key sizes were significant.

Interactions between key size and location were somewhat anomalous, and therefore are hard to interpret. The most notable findings were that error rate for keys 7.7 mm wide were highest in the NW region, and error rates for the largest key size (13.4 mm) were highest in the SW region.

Subjective Preferences

Participants were asked to rate how comfortable they felt using the keypads in each of the 4 regions, regardless of target size (7 point scale; 1 = uncomfortable, 7 = comfortable). The NE region was considered the most comfortable (μ=5.7) and SE region the least comfortable location (μ=5.0) for direct thumb interaction in serial tasks (Figure 17b).

Participants were also asked which was the smallest of the 5 keypad sizes they felt comfortable using in each region. On average, participants thought they would be comfortable with smaller keys in NE region (8.3 mm) while larger keys would be required in NW, SW and SE regions (8.9, 8.8 and 8.8 mm respectively).

3.3.8 Discussion

Although speed continued to improve significantly with even the largest targets in both phases of our study, the error rates could not be discriminated statistically for target sizes ≥ 9.6 mm in discrete tasks and key sizes ≥ 7.7 mm in serial tasks. However, considering the hit distribution for 9.6 mm targets (Figure 16), the minimum sized box that would be expected to enclose 95% of hits at any screen location would be 9.1 x 8.9 mm. Together with the fact that, on average, users claimed they would comfortable using targets as small as 7.7 mm in all regions, it is probable that targets 9.1 mm square would strike an acceptable balance between speed, error rate, and user satisfaction for discrete targeting tasks. But since we did not test 9.1 mm targets, a conservative recommendation would be for designers to use targets 9.6 mm square for discrete targets such as buttons and checkboxes.

Although error rates in serial tasks did not decline significantly with key sizes ≥ 7.7 mm, the error rates for all target sizes were higher in serial tasks than in discrete

tasks. Because this relationship suggests that targets for serial tasks should be at least as large as those for discrete tasks, as well as the fact that participants stated they would not be comfortable hitting targets smaller than 8.9 mm in some regions, we again conclude that 9.6 mm is a safe recommendation for key sizes used in serial tapping tasks, such as data or numeric entry. Given that thumb pads are generally larger than those of index fingers, it makes sense that we recommend targets slightly larger than the 9 mm targets Microsoft recommends [152] for PDA interfaces designed for finger use.

Note that because our recommendation of 9.6 mm targets is an artifact of the exact target size presented to users in the study, a reasonable rule of thumb to carry forward to designers is to strive for 1 cm^2 targets for interfaces designed for thumb interaction.

It is notable that mean transition time between taps in the serial target phase differed by target size, in contrast to what Sears recently found for stylus interaction on a virtual PDA keyboard [132]. We hypothesize that this is because users took extra care when hitting targets that were smaller than the thumb, whereas in Sears' study, the stylus was always smaller than the targets involved. In addition, since the results of the hit distribution evaluation showed a surprising right-leaning trend for targets on the rightmost column, we recommend that for right handed users, targets on the right side of the screen should extend all the way to the edge. Presumably we would observe a mirrored phenomenon for left handed users, thus we also recommend that targets on the left side of the screen should extend to the left bezel to accommodate left-handed operation of the device.

3.4 Gesture Study

An important distinction between touchscreen-based devices and keypad-based devices is that touchscreens support "direct manipulation", an interaction style coined by Ben Shneiderman [135] to describe the initiation of actions on interface objects using a pointing device, such as mouse, finger, or in the case of a touchscreen-based mobile device, a stylus. Unfortunately, the basic assumption that touchscreen-equipped mobile devices will be used with a stylus has lead to software designs that favor two-handed use since input targets are often too small or too distant to be actuated with the thumb. Yet, one-handed interfaces that support thumb operation of the touchscreen are attractive because they favor a balanced, stable grip—as opposed to the use of directional navigation hardware which is often relegated to the periphery of the device and can be too low for stable one-handed operation.

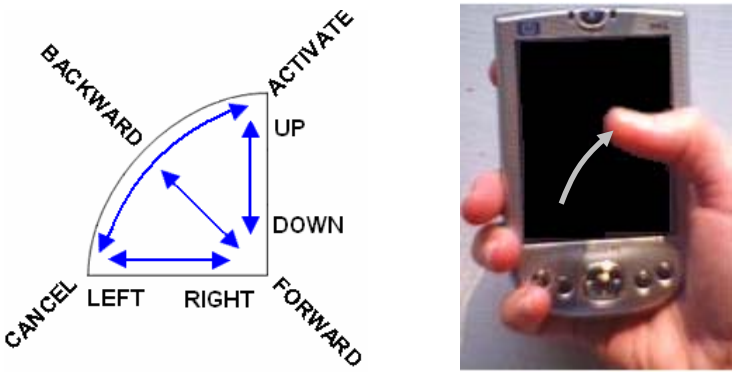
It seems clear that one option in designing for one-handed touchscreen interaction is to ensure that all targets are large enough for accurate thumb selection and are positioned close enough for users to reach comfortably with the thumb. But another approach might be to place no constraints on the size or location of interaction objects, but instead offer indirect object pointing and selection via an "object cursor". Users would be able to position the cursor over any object for selection, even objects that would otherwise be too far or too small to hit successfully with the thumb. It's easy to imagine that users could move the cursor with the directional input hardware available on most devices, but we are cautioned by Hirotaka's observation that cell phones typically require uses to interact with keypads using a low, unstable grip [65], which is also true of the directional input controls offered with touchscreen-based devices. Thus, to support a

more balanced, stable device grip, the cursor-based system might support a gesture language that could be issued anywhere on the surface of the device, and using whatever grip users found most stable. To explore the viability of interacting with touchscreen interfaces as described, we studied how well users could learn and execute a simple 8-gesture language for indirectly controlling an input cursor.

I was the sole investigator on this study, which was conducted in August of 2004, and supervised by my advisor. This work was published in the 2005 proceedings of the Conference on Human Factors in Computing Systems (CHI 2005) [76].

3.4.1 Gesture Language

We developed a gesture-based command language with the goal of accommodating the limited range of motion of the thumb when holding a device in one hand, (e.g., Figure 20b), and strived to make it simple enough that it that could be executed easily, recognized reliably, and learned with minimal training. After informally experimenting with a variety of gestures, we decided on the 8 gestures of Figure 20a. We assigned the directional commands UP, DOWN, LEFT and RIGHT to spatial gestures that map directly to their function. For example, sweeping the finger from left to right across the screen moved the input cursor to the right. We assigned ACTIVATE and CANCEL to the two gestures defined by pivoting the thumb diagonally from bottom to top and top to bottom respectively. This assignment was made both to reinforce their opposing relationship, as well as for ergonomic ease in issuing these common commands. Finally, we assigned the upper-left to lower-right diagonal to FORWARD due to its relative forward nature, and by similar reasoning, the reverse gesture to BACKWARD. These last two commands have similar functionality to TAB and SHIFT-TAB in a Windows operating system.



(a) (b)

Figure 20. Gesture Study: The indirect gesture command language for controlling the object selection cursor (a); and an example of issuing an ACTIVATE command (b).

All gestures are performed by sweeping the thumb across the surface of the touchscreen. A convenient property of this particular set of gestures is that each gesture is uniquely defined by a slope and movement direction. Since neither the length nor location of the gesture conveys information, users can issue the gestures anywhere on the screen surface, and can draw them as short or long as they wish (beyond an activation threshold of 20 pixels). This flexibility lets users interact with the device using the grasp that provides them the most stability and comfort in a mobile scenario.

3.4.2 Tasks

To study both the learnability and executability of our gesture set, we had study participants perform two types of tasks. Gesture tasks required users to perform a gesture when presented with the associated command name. Navigation tasks required participants to navigate to a particular cell within a hierarchy, and for some tasks, set a property related to that cell, described in more detail below.

3.4.3 Materials

We constructed a software test environment modeled after a hierarchical tabular information space. Each level in the hierarchy appeared as a 3x3 grid of equal-sized cells, numbered 1-9 in the same arrangement as a cell phone keypad (Figure 21). The experimental hierarchy was limited to 5 levels. Cell contents were labeled with a hierarchical address constructed by prepending the cell's local logical number (1-9) with its parent's label, separated by a dot (Figure 21a-c). This meant that the label of a cell provided a navigational reference for the user by communicating the cell's position in the hierarchy. We reserved an area at the top of the screen for task instructions, and disabled tapping to restrict the input to gestures alone.

The eight gestures were associated with the following actions within the software environment. Directional gestures LEFT, RIGHT, UP and DOWN controlled the movement of an orange rectangular cursor within a level of the hierarchy; ACTIVATE navigated to a lower level in the hierarchy and CANCEL navigated out to the previous level. Zooming animations were used to transition from one hierarchical level to another. Within the context of the test environment, FORWARD and BACKWARD were used for selection within a cell; users could "activate" the digits of a cell label (indicated by displaying the digit in **bold**) individually by moving a highlight (e.g., over the 5 in cell 6.5.4 of Figure 21c) to a specific digit and issuing the ACTIVATE command. Digits could also be un-bolded or "deactivated" using the CANCEL gesture. Participants were provided with a reference sheet that described the hierarchical information space and labeling scheme, as well as the eight gestures to be used for navigation and interaction.

The study software was run on an HP iPAQ h4155 Pocket PC measuring 4.5 x 2.8 x 0.5 in (11.4 x 7.1 x 1.3 cm) with a 3.5” screen.

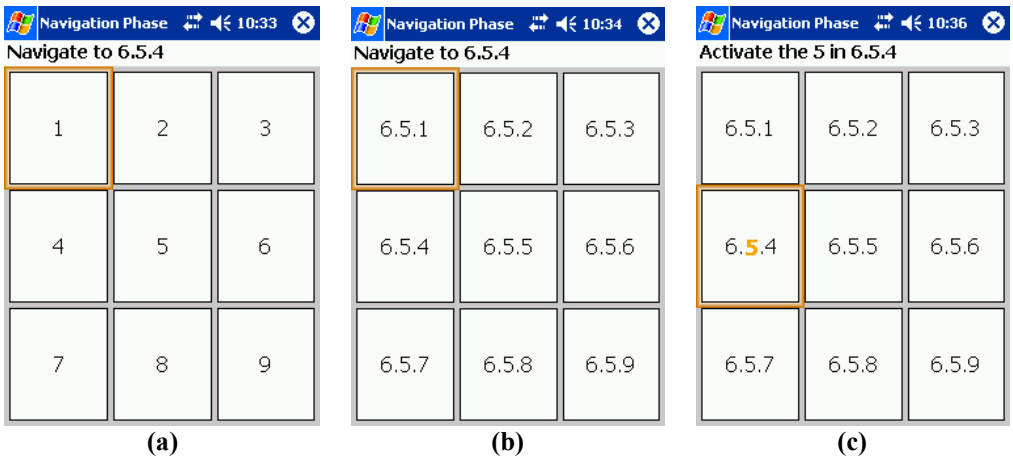


Figure 21. Gesture Study: Examples of three states in the experimental software environment. (a) The top level; (b) the third level after ACTIVATE was performed on 6.5; and (c) highlighting the 5 within cell 6.5.4.

3.4.4 Participants

Twenty participants (12 male, 8 female) were recruited from the general population with the only selection criterion that participants were right-handed. We obtained institutional review board (IRB) approval to conduct this study, and all participants granted us their informed consent. The median participant age was 30, and while 12 of the participants considered themselves advanced computer users, only 6 regularly used a handheld computer.

3.4.5 Measures

Application logs recorded the time and gesture executed for each task, and whether the task was completed successfully. Participants were instructed to press a hardware button between tasks, which served to record task time and advance the software script to the next task. Due to a bug in our logging software, task time was recorded at second rather than millisecond resolution. However, since our goal was to identify performance trends

rather than comparison to another input method, second-level resolution was sufficient. Participants rated their experience using five nine-point Likert scales: frustrating - satisfying, hard to learn - easy to learn, hard to use - easy to use, slow - fast, and boring - fun.

3.4.6 Procedure

After reading a description of the test environment and gestures, participants performed 16 practice tasks similar in nature to those in the navigation task phase. Participants practiced for 5-10 minutes.

After the practice phase, participants performed gesture tasks: presented with a command name, participants performed the associated gesture and pressed a hardware button to advance to the next task. Command names were presented to participants in random order, four times for each of the eight commands. The gesture reference sheet was placed face down so that the administrator could record the number of times it was referred to.

The second test phase required participants to perform goal-directed tasks within the information space. These included (**N**)avigation ("Navigate to 6.5.4"), (**A**)ctivation ("Activate the 5 in 6.5.4"), (**NA**)vigation+activation ("Activate the 2 in 4.3.2"), and (**C**)ancellation ("De-activate all digits in 4.3.2") tasks. Participants then recorded their subjective ratings of the interaction experience. Some participants completed the study in as little as 15 minutes, most within 30 minutes, and none required more than 40 minutes.

3.4.7 Results

In the gesture phase of the study, participants correctly performed gestures on average 87% of the time when presented with the name of the gesture. Looking at performance by gesture type, participants correctly performed directional gestures 93% of the time, but had more difficulty with the diagonal gestures ACTIVATE and CANCEL at 88% and 85% accuracy respectively. BACKWARD and FORWARD gestures had the worst rates of success, at 70% and 64% accuracy respectively. Time to perform gestures followed a similar trend. On average, participants required 2.4 seconds to perform each gesture: 1.5 – 1.7 seconds for directional gestures, 2.6 – 2.8 seconds for ACTIVATE and CANCEL gestures, and 3.6 – 3.7 seconds for BACKWARD and FORWARD gestures respectively. Although we tallied user peeks at a reference sheet, we assumed that the acts of page-flipping and answer-searching have been reflected in the task time, and thus we did not analyze this data further.

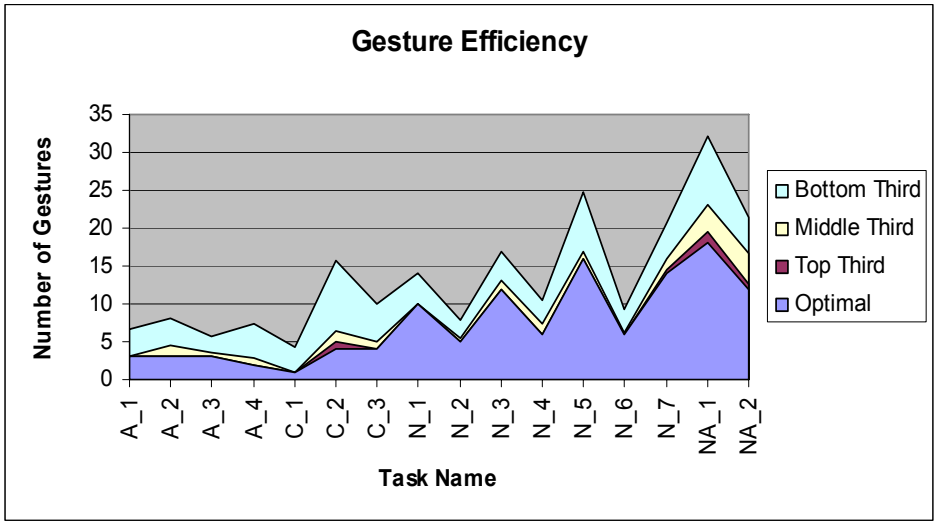


Figure 22. Gesture Study: The mean number of gestures performed for each task, as compared to the minimum, or optimal, number of gestures required.

The second test phase evaluated accuracy and efficiency for goal-directed navigation and activation tasks. The average task success rate was 95%. While both

navigation and navigation+activation tasks were performed with 98% accuracy, activation-only tasks were close behind at 96%. Cancellation tasks, however, were only completed correctly 83% of the time. On average, participants performed 2.4 additional steps than the optimal number to complete a task. By clustering participants into three performance groups for each task, we found that the Top third of participants performed nearly optimally on all but a few tasks (Figure 22). The Middle third of participants performed comparably to the Top third, but the gaps were widest between Top and Middle performers for those tasks that Top performers had most trouble with (e.g., C_2, NA_1, NA_2). That is, the tasks for which the Middle performers deviated farthest from Optimal, were those for which the Top performers also deviated. Thus, the most serious difficulties users had in performing the navigation phase tasks were experienced by only the Bottom third of the participants.

The average subjective rating for each of our 5 satisfaction measures, on a scale of 1-9 where 9 was positive, fell within a 1 point span of one another, between 5.9 (satisfying) and 6.75 (fun).

3.4.8 Discussion

Because the gesture phase of the study did not distinguish between errors of recall (remembering which gesture was associated with a command name) and execution (tried to issue the appropriate gesture, but was unrecognized), we could not classify the reasons that users were unsuccessful in performing some of the gestures. Analyzing the log files, it is safe to say that errors of both types occurred. The low error and speed measures for directional navigation support our hypothesis that the directional gestures have an intrinsic spatial mapping. Presumably, this mapping contributes to better learnability,

more reliable execution, and lower cognitive demand. The positive jump in execution speed for the other four gestures is unsurprising when we consider that the mappings between gesture and command are more abstract than the directional mappings, and likely to require more cognitive effort to perform the mental transformation. This alone does not explain the associated increase in error rate, but insights from Long's work on gesture similarity [86] suggest that users may perceive the diagonal gestures as similar, and because they have different functionality, the similarity in this case would make them more difficult to learn.

The better performance that the ACTIVATE and CANCEL diagonal gestures enjoyed over the BACKWARD and FORWARD diagonal gestures may be attributed to the fact that BACKWARD and FORWARD should have been more physically challenging to perform. However, the difference may also be due to the disproportionate practice time the two sets of diagonals received, since a single navigation task provided more opportunities to issue ACTIVATE and CANCEL gestures than the within-cell activation tasks provided for FORWARD and BACKWARD. These intuitions also help explain the efficiency results—users were more successful and efficient in pure navigation tasks which contained a proportionally large number of directional gestures compared to within-cell activation/cancellation tasks. It should be noted that the relative complexity of the gesture navigation environment may have confounded the results by inflating the inefficiency measures.

Overall, the performance and subjective rating data suggest that indirect gestures can be learned and performed with minimal (< 10 minutes) training, especially if there exists a direct spatial mapping between each gesture and its associated function.

3.5 Conclusion

Motivated by the results from my early field study and web survey, which suggested mobile devices should be updated to better support one-handed use scenarios, I ran three studies to develop the background needed to offer designers and developers high-level guidelines for building more effective one-handed mobile interfaces. Based on the results of my study of thumb movement, I have developed the following rules for the placement of interaction objects (including physical and touchscreen buttons) on mobile devices to support one-handed thumb operation: 1) support localized interaction (e.g., by placing targets close to one another) to minimize grip adjustment; 2) avoid diagonal movements to accommodate both left and right handed users, especially for common, or repetitive tasks; 3) strive to place interaction targets toward the vertical center of the device to support a stable, balanced grip, as well as toward the middle of the device (horizontal center) to make targets easy to reach for both left and right handed users; and 4) favor small devices in order to support overall device control and satisfaction, but beware of making interaction targets too small as a result. My investigation of appropriate target sizes for touchscreen-based devices found that 1 cm^2 strikes a conservative balance between hit speed and accuracy, but that targets as small as 9.1 mm^2 may also satisfy users. Finally, data from my exploration of touchscreen gestures suggest that small gesture sets can be learned and performed with minimal (< 10 minutes) training, especially if there exists a direct spatial mapping between each gesture and its associated function.

My documentation of average consumer behavior and preferences concerning the one-handed use of mobile devices, together with the raw results and recommendation from these three studies constitute the *foundations* of my research.

Chapter 4
Applications: Touchscreen Design Strategies for One-Handed Mobile Computing

Interface design guidelines concerning the appropriate placement of interaction objects for supporting comfortable one-handed mobile device operation are useful to designers of interfaces, as well as those who use them. However, in practice, adherence to such guidelines provides only a low-level, mechanical benefit to users, and does not guarantee efficient, aesthetic, or enjoyable interfaces. Furthermore, as mobile devices store and connect to ever larger data sets, their small screens and constrained input channels pose significant challenges for providing users effective presentations of and efficient navigation techniques for such data. Thus beyond offering basic rules for addressing the pragmatic physical issues involved in using touchscreen-based devices with one hand, better overall interaction techniques and information structures are needed to empower users of these compact mobile portals so that their whole-device experiences become increasingly satisfying.

In an effort to improve one-handed mobile device usability in supporting high-level user goals, I explored four different interaction techniques and studied their benefits by applying each to an important information task that users face today. The work in these next three chapters offers insight into how my *foundational* design guidelines apply in practice to real world *applications* and information tasks.

4.1 Overview

When considering possible design approaches for touchscreen interfaces that support one-handed operation, it seems obvious that one strategy is to ensure that all interaction targets are large enough to be hit reliably with the thumb. While this approach allows users to operate the device fully with one thumb, it also constrains how designers use the available screen real estate for presenting information, and may actually increase the number of navigation steps required to complete a task. Another approach might be to place no constraints on the size or placement of interaction objects, and offer some other solution for allowing users to hit objects that are too small to hit accurately or too far to hit easily with the thumb.

In this chapter I explore the tradeoffs between these two design approaches by developing and studying alternative interfaces for navigating among a set of device programs. The designs are distinguished from one another by the level of restriction each places on the sizes and positions of interface objects, the interaction philosophy supported, the number of applications managed, and the zooming metaphor used for overview and navigation.

This work was performed in collaboration with John SanGiovanni, formerly of Microsoft Research. I, with the oversight of my advisor, take full ownership of the AppLens design, including its associated gesture language. John SanGiovanni was the inspiration behind the LaunchTile design and interaction style. I executed all technical aspects of the work, including the implementations of AppLens and LaunchTile, as well as the design and administration of their comparative user study. I was the first author on

the paper we wrote describing this work, which was published in the 2005 Proceedings of the Conference on Human Factors in Computing Systems (CHI 2005) [76].

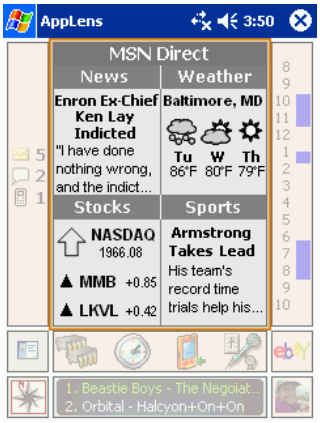
4.2 A Comparative Design Strategy

The approach taken in this investigation was to design and build the best interfaces we could for accomplishing the same functional goal, but by instituting two different design strategies. By evaluating the designs with real users and comparing and contrasting the results, we hoped to understand which design directions make the most sense for the domain of program navigation and management, as well as to learn something generally about the trade-offs between the two approaches in the process. Here I describe some of the features that the two systems shared, as well as where their designs diverged. In brief, both systems were designed to support the task of navigating among a set of user programs (e.g., email, calendar, etc.), but they differed in the one-handed interaction strategy used, the number of programs managed and the zooming metaphor used for providing content overviews and navigating among the programs. Table 3 captures the differences between the two designs in a format that facilitates comparison. Both prototypes were architected to scale to devices with varying screen resolutions, aspect ratios, and input methods, and were demonstrated successfully on both an HP iPAQ Pocket PC running Windows Mobile 2003 and an iMate Smartphone II running Windows Mobile Smartphone 2003.

As previously described, the motivation for this work was to explore two different design approaches for supporting thumb use of touchscreen software. The AppLens prototype placed no restrictions on the sizes or placements of objects, but supported a gesture command language for indirect selection of and interaction with objects that were

too small or too far for reliable thumb access. The LaunchTile prototype instead ensured that all interface objects were large enough for direct thumb interaction, and supported direct-manipulation gestures for dragging out-of-reach elements closer to the thumb.

Table 3. A comparison of the design features of the AppLens and LaunchTile interfaces.

	Visual Representation	Design Strategy	Number of Programs	Layout	Zooming Technique
AppLens		No restriction on size or placement of objects Indirect command gesture language	9	3x3	Tabular Fisheye
LaunchTile		Thumb-sized objects within reach of the thumb Direct manipulation gestures	36	6x6	Pure Zooming

One shared design goal of AppLens and LaunchTile was to provide access to rich notification information for multiple programs at once. Most current PDA interfaces are designed for focused interaction with a single task or program, with limited consideration or display real estate allocated for notifications (e.g., email, appointment reminders) or monitoring of ambient information streams (e.g., stocks, sport scores). In the proposed designs, each of the device-resident programs was associated with a dynamic notification

tile instead of the traditional static icon used to launch the program. In both designs the notification tiles were arranged within a 2D landscape that offered at-a-glace information for several applications at once, as well as on-demand application launch when users desired more detailed information and program access. However, the two designs differed in the number of notification tiles, and thus programs, managed within the landscape. AppLens provided access to nine programs, whose notification tiles were arranged in a 3x3 grid, whereas LaunchTile supported thirty-six programs in a 6x6 grid.

The two systems also employed different zooming interface techniques for displaying the high-level summary information for several notification tiles at once (zoomed out) versus displaying the details of a single interactive program (zoomed in). Zooming has been used extensively in desktop environments as a method for linking data details to the overviews that can show global relationships among the elements of a large data set. AppLens used a tabular fisheye approach [118] to smoothly transition between notification representations of all nine applications and the details of a fully functioning program, while LaunchTile used pure semantic zooming [18] to offer the same functionality for its thirty-six programs. Fisheye and pure zooming techniques both have shown promise in other data domains and tasks, but in the absence of clear guidelines about when each approach should be used [35], we hoped that by exploring each zooming approach we might gain insight into whether one has advantages over the other for the task of program monitoring and navigation.

4.3 Related Work

Past research that is relevant to this exploration include those efforts that have focused on reducing the number of hands required for device use (Chapter 2), determining

appropriate target sizes for touchscreen interaction (Chapter 3) and gesture-based systems (Chapter 3). A discussion of research concerned with hitting small targets with fingers on touchscreens is covered in Chapter 6.

A distinguishing characteristic of the AppLens interface is its application of tabular fisheye distortion to achieve a focus+context approach to program navigation and management. Spence and Apperley [137] introduced the "bifocal display" as one of the first examples of fisheye distortion applied to computer interfaces. Furnas extended the bifocal display to include cognitive perceptual strategies and introduced a set of analytical functions to automatically compute generalized fisheye effects [51]. Since then, fisheye distortion has been applied with mixed success across a variety of domains, including graphs [125], networks [128], spreadsheets [118], and documents [66, 122].

Bederson et al. [16] drew upon that early work in developing DateLens, a space-conserving calendar for PDAs. One of the strengths of DateLens was the pairing of distortion and scalability, which allowed the details of a single day to be viewed in the context of up to several months of appointment data. Also important was the design and use of effective representations for the variety of cell sizes and aspect ratios that resulted from tabular distortion. One drawback of DateLens, however, was that it required two-handed stylus interaction to actuate the small interface widgets. The AppLens design extends the principles of DateLens to include one-handed thumb access and generalizes the approach for use across a variety of domains.

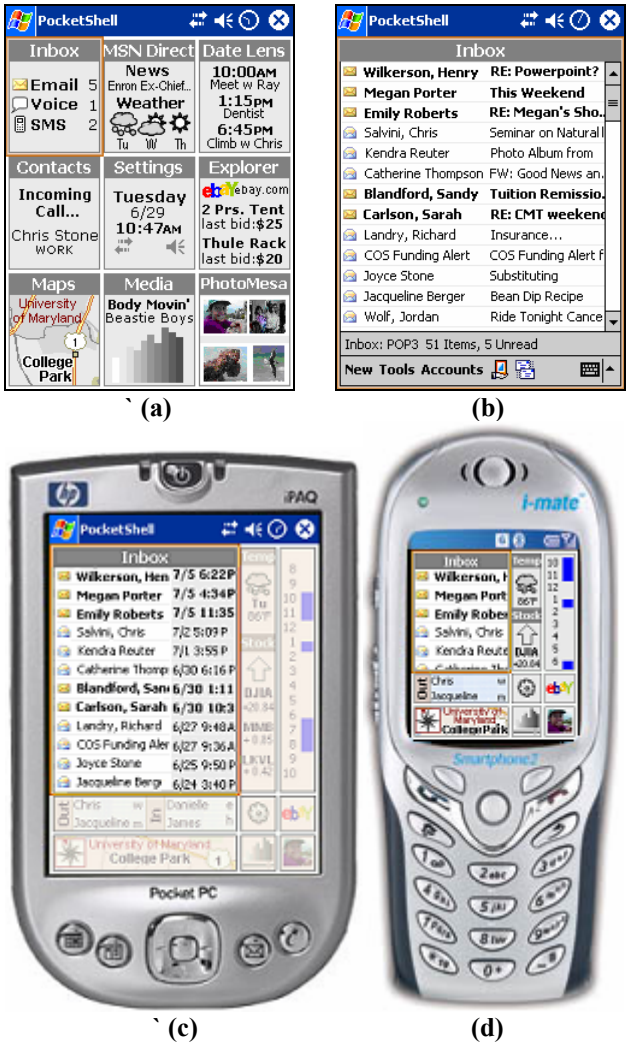
Although the LaunchTile interface preceded the Apple iPhone (www.apple.com/iphone/), the iPhone represents one of the first commercial touchscreen-based mobile devices that has been specifically designed for finger interaction. Like

LaunchTile, the iPhone design takes the approach of making all targets large enough to be hit easily with the finger. For the few iPhone tasks that do not feature finger-sized targets, additional accommodations are made to improve the accuracy of finger selection. For example, the on-screen Qwerty keyboard features smaller targets than would guarantee accurate selection, so word prediction algorithms are employed to enlarge the hit regions of more likely targets, as well as auto-correct unlikely character sequences. iPhone users are also offered a magnifying lens for precisely placing the input cursor for text editing tasks. While it does not seem that Apple made specific design choices to ensure users can reach all targets for one-handed operation (e.g., the entire screen is used for object layout), the phone's lightweight, slim form generally allows users to hit all areas of the screen with a single grip. Both the iPhone and LaunchTile designs support finger gestures to scroll through 1D lists, although with LaunchTile users literally drag the list, while with the iPhone users start the scroll action with a single "flick", and a model of inertia then slows the list naturally to a stop. Most significantly, the iPhone debuts capacitive touchscreen technology in a mobile phone, and so supports higher precision finger detection than standard mobile touchscreens, as well as multi-touch interaction.

4.4 AppLens

AppLens provides one-handed access to nine applications, and as suggested above, is strongly motivated by the DateLens tabular fisheye calendar [16]. The AppLens software design includes a scalable architecture that includes a grid, tabular layout algorithm, and default views for cell contents at a variety of sizes and aspect ratios. I also developed a general API to make it simple for applications to be built within this framework; to use

the API developers would need only to replace a small number of cell views with representations that are meaningful within the target domain.



` (a) (b)

` (c) (d)

Figure 23. Examples of the three AppLens zoom levels. Notification (a), Full (b), and Context (c, d).

4.4.1 AppLens Zoom Levels

AppLens (Figure 23) was implemented within the generalized tabular fisheye framework, using a 3x3 grid, and assigning one of nine applications to each cell. The support for tabular layout includes representations at three zoom levels: *Notification*, *Context* and *Full*.

Notification zoom distributes the available screen real estate equally among the nine application tiles (Figure 23a). One tile (shown centered) is reserved for settings, which can be used to configure the selection of applications which occupy the other eight notification tiles. Generally, tiles at *Notification* size display high level static and/or dynamic application-specific notification information.

Context zoom (Figure 23c) allocates roughly half the available display area to a single *focus* tile, compressing the remaining tiles according to a tabular fisheye distortion technique [16, 118]. A tile at *Context* size typically appears much like a fully functional application, but selectively shows or hides features to accommodate the size constraints of the smaller display, and is not interactive. Tiles on the periphery of a *Context* zoom, called *peripheral* tiles, may be rendered quite differently depending on their positions relative to the focus tile, which dictates the aspect ratio of the peripheral tile (i.e., whether it is a square, a wide-flat rectangle, or a narrow-tall rectangle). To reduce visual overload, peripheral tiles are displayed at 40% transparency. The contents of distorted peripheral tiles are not themselves distorted, but rather change representation to provide the most meaning in the space available.

The third and final *Full* zoom level expands a tile to a fully interactive application that occupies 100% of the display (Figure 23b).

4.4.2 Gesture-Based Cursor Navigation

Existing application designs for PDAs are often inappropriate for one-handed use due to their reliance on screen regions that users may not be able to reach while maintaining control of the device (e.g., accessing the Start menu in the upper left-hand corner of a display while holding the device in the right hand), and the use of standard widgets that

are too small for reliable thumb use (e.g., radio buttons, checkboxes, and on-screen keyboards). In support of traditional designs, AppLens uses an object cursor to identify the on-screen object that is the current interaction target. The cursor is depicted as a dynamically-sized rectangular orange border that users move from one on-screen object to another via command gestures, which are described in more detail below. Cursors are not new to PDA interface design: the WebThumb [154] web browser includes a similar notion of cursor, but which is controlled via directional hardware, and others [149] have explored device tilting to manipulate PDA cursors.

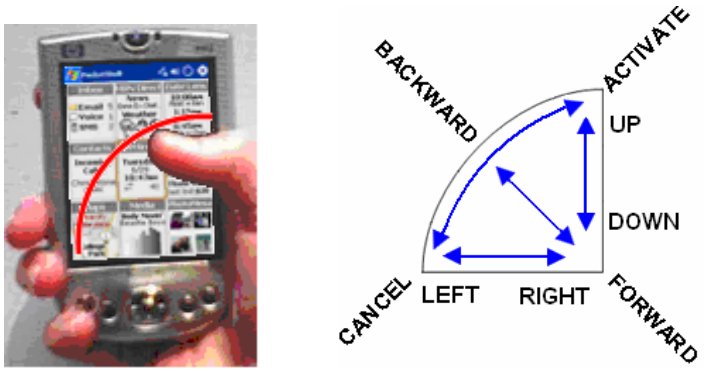
Neither the cursor nor gestures interfere with the most common stylus interactions of tap and tap+hold. Although gesture movements do resemble stylus drag commands, dragging is rarely used in handheld applications and could be distinguished from gestures by explicitly setting a gesture input mode.

We established a core set of commands that would allow users to navigate applications using only the input cursor. The command language supported directional navigation (UP, DOWN, LEFT, RIGHT) as well as two widget interaction commands: one equivalent to a stylus tap (ACTIVATE), and the other which negates widget activation (CANCEL), equivalent to tapping the stylus outside an activated widget. We also included the commands FORWARD and BACKWARD, which equate to TAB and SHIFT-TAB on Windows PCs, for convenience.

4.4.3 Command Gestures

Our use of gestures is motivated by Hirotaka's observation that the button positions on many cell phones require interaction using a low, unstable grip [65]. Most PDA joysticks face a similar drawback in that they are often located along the lower perimeter of the

device. AppLens avoids this problem since its gestures can be issued anywhere on the screen. Each AppLens gesture is uniquely defined by a slope and direction, or vector, which allows gestures to be robust and highly personalizable; users can issue gestures of any length (beyond an activation threshold of 20 pixels) anywhere on the touch-sensitive surface. This flexibility lets users interact with AppLens using the grasp that offers them the most stability in a mobile scenario.



(a) **(b)**

Figure 24. A depiction of the screen region that is accessible to the thumb of one hand (a), and the AppLens command gesture set (b).

We based the gesture set on the limited motion range of thumbs (Figure 24a), with the goal of creating a gesture language that could be learned with minimal training. The gesture set (Figure 24b) and a user study designed to evaluate its learnability were presented in detail in Chapter 3, Section 3.4. When using a device that has no touchscreen and only a numeric keypad, each gesture command maps to the key that corresponds logically to the gesture endpoint: 1-BACKWARD, 3-ACTIVATE, 7-CANCEL, and 9-FORWARD. Since nearly all phones have a joystick that can be used for directional navigation, the keypad-to-command mapping is not strictly necessary for moving the input cursor, but for completeness we made the following assignments: 2-UP, 4-LEFT, 6-RIGHT and 8-DOWN.

4.4.4 Using Command Gestures within AppLens

Users navigate between AppLens zoom levels using ACTIVATE and CANCEL gestures. As a rule, the ACTIVATE gesture behaves as a stylus tap on the current cursor target, thus its effects are target-specific. Since application tiles are non-interactive at *Notification* and *Context* zoom levels, the ACTIVATE gesture simply zooms in, animating the layout first from *Notification* to *Context* zoom, and then to *Full* zoom. Once at *Full* zoom, the input cursor transitions to the objects *within* the application, at which point the command gestures affect the current target widget. The CANCEL command negates the effects of the ACTIVATE command. At *Full* zoom, the effects of the CANCEL command depend on the location of the cursor and the state of its target. The CANCEL command will first deactivate an active target. If the current target is not in an active state, CANCEL will cause the application tile to animate from *Full* zoom to *Context* zoom, and if issued again, to *Notification* zoom.

4.5 LaunchTile

The second design, LaunchTile uses another way to interact with a grid of notification tiles. LaunchTile is an interactive zoomscape consisting of thirty-six application tiles, divided into nine zones of four tiles each (Figure 25c). The 36-tile configuration was an exploration of the maximum number of applications that can reasonably fit within the display space. Since the design is fundamentally scalable, however, it can display any number of tiles *up to* thirty-six, and fewer may even be preferable. For example, Zumobi (www.zumobi.com), a direct commercialization of the LaunchTile design that is VC-backed and invested in by Microsoft, chose to use sixteen tiles, arranged in four zones of four tiles (e.g., a 4x4 layout), shown in Figure 26.

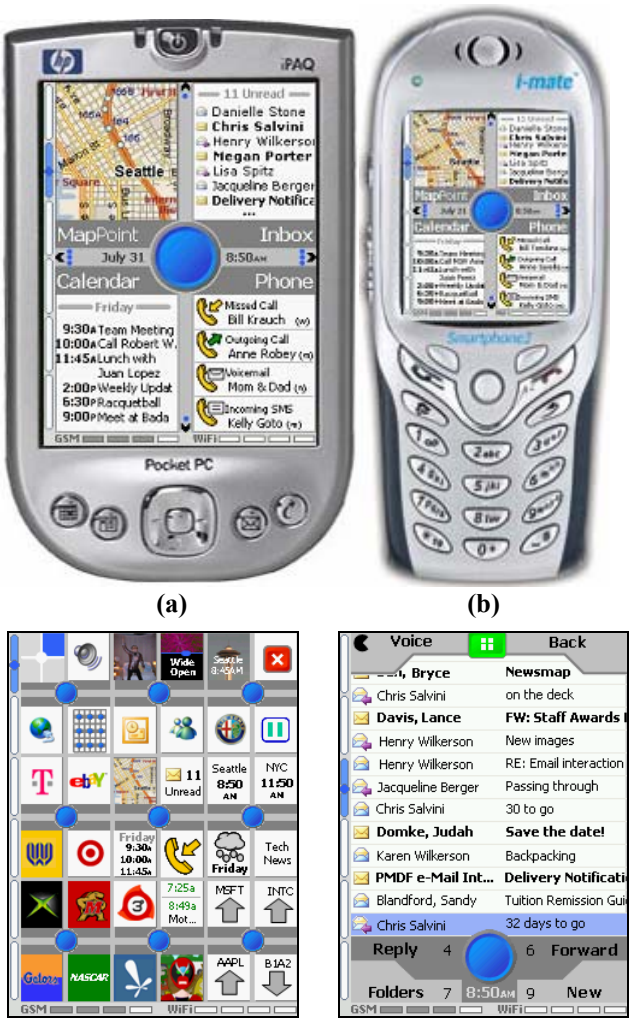


Figure 25. Examples of the three LaunchTile zoom levels. Zone (a,b), World (c), and Application (d).

Figure 26. The three zoom levels of Zumobi, a commercialization of the LaunchTile design.

As a central design element, LaunchTile uses a large blue onscreen button, hereafter referred to as Blue (Figure 25a), to unify the zoomscape with a consistent visual metaphor. One goal of Blue is to provide a consistent visual point of reference—onscreen tiles and menus maintain the same relative position to Blue during all zooming and panning operations. The LaunchTile zoomscape consists of three zoom levels: *World* (thirty-six tiles, Figure 25c), *Zone* (four tiles, Figure 25a,b) and *Application* (one tile, Figure 25d) zoom.

4.5.1 Zone View

The Zone view of LaunchTile divides the screen area into four equally-sized application tiles (Figure 25a). To view other tiles, the user pans the zoomscape to neighboring four-tile clusters, called *zones*. The zones are arranged as a 3x3 grid, each associated with a numerical designator from 1-9 as on a conventional telephone keypad. Zone 5 is the center zone, which defines the Home screen, and shows the four highest priority notification tiles, as configured by the user.

4.5.2 Panning Techniques

To support various input hardware and styles of interaction, there are several ways to pan the zoomscape within Zone view. If the device has a multidirectional control joystick, the user can push it in the direction of the targeted zone. From the Home screen (Figure 25a), the sixteen tiles in zones 2, 4, 6, and 8 are only a single tap away. Since most directional navigation hardware on mobile devices does not support diagonal actions, the sixteen additional tiles in the corner zones 1, 3, 7, and 9 are two taps away. Alternatively, if the device has a touch-sensitive display, the user can use her thumb directly to drag the

zoomscape. Dragging is performed “on rails”, so to speak, permitting the user to drag vertically and horizontally, but not diagonally. Although the zoomscape moves with the thumb during dragging, Blue remains centered and stationary. Because only one instance of Blue exists within Zone view, each zone is depicted with an empty center hub during dragging. Upon thumb release, the zoomscape animates to align Blue with the closest zone’s empty hub. This automated guidance ensures the user is never caught between zones.

Within each four-tile zone, additional visual cues communicate the user’s relative location within the zoomscape. First, directional arrows, centered along each edge (Figure 25a), designate where the other zones are. If a user only sees arrows pointing up and right from within a Zone, she knows she is currently in Zone 7. Small blue dots featured alongside the arrows represent the locations of the eight remaining zones in the zoomscape. The small blue dots might also be used to indicate an alert or status change in a neighboring zone, although this feature was not implemented in the prototype. One final way to pan the zoomscape is to tap a directional arrow directly. An oversized hit target ensures that the user can easily hit the arrow without using a stylus.

4.5.3 Zooming Out to the World View

From any one of the nine four-tile Zone views, the user may tap Blue (or press the 5 key) to zoom out to display the entire 36-tile zoomscape (Figure 25c). Since all thirty-six tiles are visible at once, this view reduces each tile to a small icon. From this World view, the user can easily see the absolute location of each tile, as well as monitor the status of all applications at once. In the World view, the display is divided into a grid of nine hit targets, each of which map to a four-tile zone. Tapping a zone (or pressing its

corresponding keypad button, 1-9) animates to Zone view, displaying the zone's four notification tiles.

4.5.4 Zooming In to an Application

To launch an application, a user taps any of the four notification tiles within Zone view, at which point an animated zoom sequence pans the zoomscape into the application until it fills the entire display (Figure 25d). If the device has a numeric keypad but no touchscreen, the user presses the numeric key that corresponds to the zone: pressing 1 launches the upper left tile, 3 launches the upper right tile, 7 launches the lower left tile, and 9 launches the lower right tile. This technique provides quick, single-tap access to each visible tile, and was inspired by ZoneZoom of Robbins et al. [121].

As the system zooms, Blue stays within view, maintaining its role as a central point of reference (Figure 25d). Application menu commands are represented as on-screen buttons clustered around Blue, which has migrated to the bottom of the display. Each menu button displays a numeric label, so that users of non-touchscreen devices may activate each menu by pressing the corresponding number on the keypad. A visual indicator on the left-hand side of the screen animates during zooming and reflects the user's current absolute zoom level within the LaunchTile zoomscape.

4.5.5 Zoom Control

Pressing Blue typically moves the user deeper into the zoom hierarchy, while a dedicated Back button moves the user up the hierarchy. In the Zone view however, Blue toggles between Zone view (four tiles) and World view (thirty-six tiles). Once an application is launched, three dedicated software buttons along the top edge of the screen support inter-

and intra-application navigation (Figure 25d). A green Home button immediately zooms the view out to the Home screen, and the Back button to its left returns the display to the previous screen in the user's navigation sequence. The other global command key was not used in the prototype, but might be used, for example, to input voice commands. On a non-touchscreen device, Back and Home commands are executed with dedicated physical buttons, such as those provided on a smartphone.

4.5.6 Application-Level Interaction

The original focus of the AppLens and LaunchTile designs was to support the task of navigating among a set of programs. However, to further explore how touchscreen-based applications *themselves* might be designed to support direct thumb interaction, we extended the LaunchTile design to the application level. Within an application, we took advantage of gesture-based techniques to support selection of items that might have been too small to hit reliably with the thumb. Others have previously demonstrated high-precision selection capabilities for fingers on touchscreens using a combination of an offset cursor (to avoid finger occlusion), and lift-off selection (to allow for visual confirmation) [116]. When possible, we made all LaunchTile targets large enough for thumb activation, but in cases when limited display real estate necessitated smaller targets, a toolglass was provided which could be positioned over target objects (e.g., contacts, email headers, message text). Blue served as the handle to the toolglass, large enough to hit easily with the thumb, but offset below it so as not to occlude the desired objects. Alternatively, the user could drag the application contents, such as to pan a map, scroll a list of email, or navigate to text outside the view area. Together, these two interaction techniques permitted the user to access a large application content space

directly with the thumb, while also allowing selection of objects smaller than a finger, such as would be required for editing text. Alternatively, users with non-touchscreen devices could use the multidirectional joystick to position the toolgalss precisely. Numeric keys 2 and 8 could be assigned for page up and page down control, while keys 4 and 6 could trigger menu items or horizontal page control, as appropriate.

Once the targeted item was in the toolglass selection area, a tap on Blue (or pressing the 5 key) activated the item, which either drew the user further into the application (e.g., open an email), or if users were at the lowest level of the zoomscape hierarchy, the tap on Blue would trigger a context-sensitive menu, whose items were positioned radially around Blue. At this point users could tap on menu items directly, or use the keys of the numeric keypad, to perform functions on the selected item.

4.6 Implementation

The AppLens and LaunchTile prototypes were built using the University of Maryland's PocketPiccolo.NET development toolkit for Zoomable User Interfaces (ZUIs) [17] and were designed to accommodate screens from 2" to 5", measured diagonally, with resolutions ranging from 176x220 to 800x600. Although the primary development and test platform was a the HP iPAQ 4155 PocketPC running Microsoft Windows Mobile 2003, both system ran unmodified on an iMate Smartphone II running Windows Mobile Smartphone 2003 (Figure 23c,d and Figure 25a,b).

Although the core architecture and gesture recognition for each system was implemented fully, the programs running within AppLens and LaunchTile were simulated with images. This allowed us to present the AppLens and LaunchTile designs to users in a way that preserved the look and feel of the real systems for generating early

feedback, but of course fell short of full interactivity. However, because the LaunchTile design principles extended to the applications themselves, we also prototyped email as an interactive example within LaunchTile.

4.7 AppLens and LaunchTile Formative Study

We conducted a formative study of the AppLens and LaunchTile systems to capture usability issues that their designs presented for new users, and to elicit general user reactions and comparative preferences

4.7.1 Participants

We recruited ten participants (8 male, 2 female) from a local private scientific research center. We obtained institutional review board (IRB) approval to conduct this study, and all participants granted us their informed consent. Three participants were in their 20s, five in their 30s, and two were 40 or older. While all participants considered themselves advanced computer users, four used PDAs regularly, and four had never used a PDA.

4.7.2 Measures

Participants provided subjective design-specific and comparative reactions to AppLens and LaunchTile through think aloud and verbal questionnaires.

4.7.3 Materials

The system prototypes ran on an HP iPAQ h4155 Pocket PC measuring 4.5 x 2.8 x 0.5 in (11.4 x 7.1 x 1.3 cm) with a 3.5” screen. A one-page document described the AppLens design and gestures set, followed by a list of eight associated tasks. A two-page document

described the LaunchTile design and methods for navigation and interaction, followed by a list of 11 tasks.

4.7.4 Tasks

For each interface, participants performed tasks which were designed to exercise the full range of navigation and interaction features. For example, LaunchTile tasks included navigating to specific zones, finding specific applications, and opening and editing an email message. AppLens tasks included navigating to specific application tiles, and answering questions about application content.

4.7.5 Procedure

Study participants were introduced to each design by reading its design document and performing each of the related tasks. During the tasks, the test administrator recorded think-aloud reactions and usability observations. After users completed all tasks, the administrator recorded answers to open-ended questions related to the interaction methods, such as likes and dislikes, features that were easy or hard to use or learn, and comfort level using one hand. The same procedure was repeated for the second interface. The administrator balanced the order of the interfaces among participants. After completing the tasks and questions for both interfaces, participants were asked questions designed to elicit comparative preference between the two systems. We limited each user session to 45 minutes, allotting roughly fifteen minutes to each interface, and the final fifteen minutes for comparative feedback.

It should be noted that users were not given any dedicated training for learning the AppLens gestures set, but could refer to the document describing the gesture language at

any time. Furthermore, the study tasks only required users to perform six of the eight gesture language commands: UP, DOWN, LEFT, RIGHT, ACTIVATE, and CANCEL.

4.7.6 Results

Reactions to AppLens were quite consistent across participants. Because only one question in the study focused on a specific interface design feature (*Context* view), we regarded commonality among participant responses as indicative of the interface characteristics that impacted users the most. I report here on only the strongest trends in user opinion. Half (five) of the participants commented that they liked the *Notification* view and the ability to access all nine application tiles within both *Notification* and *Context* views. Even though two participants found nothing redeeming about the fisheye *Context* view, all others found it useful at least some of the time. Seven out of ten participants found application navigation easy and enjoyable, but performed the majority of navigation by tapping tiles rather than using command gestures. Even so, participants were required to use gestures to zoom out from *Full* zoom, and two participants particularly enjoyed performing the gestures. Five participants agreed that the gestures were the most difficult aspect of the interface, but disagreed on why, citing confusion over the gestures for zoom-in vs. zoom-out, difficulty performing the ACTIVATE gesture, difficulty with directional navigation, and frustration due to the system misinterpreting the intended gesture. All participants found AppLens both easy to learn and effective for navigating among applications, all but one found one-handed use comfortable, and six out of seven participants stated they would prefer AppLens over their most familiar PDA operating system.

Perhaps due to the richness of the LaunchTile environment, reactions to LaunchTile were more mixed than they were for AppLens. Nearly half the participants reacted positively to the aesthetics of LaunchTile, and specifically to Blue. Seven participants appreciated the volume of information available through the two zoom perspectives World and Zone, while six thought that zooming between those perspectives was among the easiest aspects of the interface. The majority (seven) of participants felt comfortable using one hand to perform the tasks, and eight felt they were able to effectively navigate among applications, which they accomplished primarily by zooming out to World view, and then zooming into the desired application.

Surprisingly, the majority (seven) of participants had difficulty panning by dragging within LaunchTile, commenting most often that it was unintuitive, but also that it was slow. This subset of participants was slightly skewed toward participants who used the LaunchTile interface first, and so their bias against dragging as a style of navigation would not have been caused by prior experience with the AppLens directional gestures. Six participants struggled with the multi-modal nature of Blue, stating they were unsure about its role in different contexts, and especially within applications. A related problem was that users had trouble differentiating between the roles of the Home, Back, and Blue buttons from within an application. Most of the participants were tentative and had difficulty performing tasks within the email application. Ultimately, participants were evenly divided (three vs. three) about whether they would choose to use LaunchTile to manage applications on their own device. This was a weaker reaction than that seen for AppLens, as six out of seven participants stated they would choose to use AppLens to manage applications on their device.

In comparing AppLens and LaunchTile to one another, most participants recognized that the more applications they could view at once, the less information each application could convey. Given this, seven out of nine participants thought AppLens provided better at-a-glance value. Nearly half of the participants were reluctant to compare the speed of information access between the two interfaces because of the fundamental mismatch between the number of applications each interface managed, and thus a mismatch between the *amounts* of information that could be accessed from each. However, seven out of nine participants thought AppLens supported faster data access, presumably because they thought AppLens struck a better balance between the number of applications managed and the quality of the information content that could be displayed simultaneously. Additionally, AppLens was considered easier to use (seven out of nine), and eight out of nine stated they would prefer AppLens to LaunchTile for use on their own device. In response to our general question about the utility of one-handed use, seven participants thought one-handed interaction would be useful at least some of the time, with three of those stating an a priori preference for one-handed use in all situations. However two participants expressed that they would never want to use a PDA with one hand, regardless of the interface design.

4.8 Discussion

Two notable themes emerged from the comparative study between AppLens and LaunchTile. The first was that participants were generally reluctant to use gestures. Results from my gesture study presented in Chapter 3, Section 3.4, suggested that directional gestures can be learned quickly, yet for both of the systems studied here, users favored tapping targets directly over issuing command or dragging gestures. While this

trend may simply be a training issue, it may also be a warning for the adoptability of gesture schemes. A second theme was that a large number of users commented on the perceived utility of the display of high-value content from multiple applications at once, bolstering our intuition that flexible notification-based designs may provide an effective balance between functionality and data presentation within the real estate constraints of handheld computing.

While AppLens appeared to be the preferred design, we must take care in assigning reasons for the preference. First, AppLens was both a simpler design and a shallower prototype than LaunchTile. Its appeal may have been that users felt proficient and better able to manage nine applications (versus thirty-six) with minimal training. Its simplicity also made AppLens less prone to the performance limitations of the hardware, which negatively impacted zooming quality in LaunchTile. However, more experience with the two designs might have tipped the scales the other way, as Bederson has pointed out in [15] that even complex interfaces have the potential to be highly satisfying after users have expended the effort to become experts. In fact, a vocal minority of participants who happened to also be expert PDA users, stated a strong preference for LaunchTile because of the large number of applications it was able to manage. While we do not believe LaunchTile deserves the designation of "expert interface", fifteen minutes is clearly not sufficient time for users to become proficient with the variety of interaction techniques it supports.

In [15] Bederson hypothesizes that user satisfaction is related to how well an interface supports "flow", which correlates inversely to the degree to which an interface gets in the way of, or interrupts, user tasks. Blue is an example of a LaunchTile feature

that interrupts flow: it performs different functions in different contexts, requiring users to keep track of the system state to predict the outcome of tapping Blue. This type of functional overloading is a well-known problem for usability, but is commonly used nonetheless. For example, both the Microsoft and Apple desktop media players use a single button for both Play and Pause. Just as with LaunchTile, these designs compromise straightforward functional mappings in favor of a visually simpler design. The difference, however, is that both media players change the button icon to reflect the current state (i.e., the function the button will perform when pressed) so that users don't have to remember the state or deduce the state from less obvious cues. A similar adaptation for Blue may reduce or even eliminate user confusion in the LaunchTile design.

4.9 Conclusion

Based on the participants' strong interest in one-handed PDA use, and generally positive reactions to their interaction experiences, this work offers further evidence of the value of research in one-handed designs. In addition, favorable feedback about the notification tiles in both designs suggests that notification may play an important role in the effective utilization of the limited display resources available in mobile computing. However, participant feedback favoring AppLens suggests that designs should strive for a balance between the number of simultaneous notification channels offered to users and the quality of the information provided therein. Unfortunately, since users were otherwise very receptive to the two designs, we learned little about the relative values of the competing design choices embodied by the two systems, such as whether using large, nearby targets that can be hit directly is better than using small or far targets and an indirect gesture

language, or whether pure zoom is better than fisheye zoom for showing contextual information.

Chapter 5
Applications: Search Strategies for One-Handed Mobile Computing

The goal of my work in the previous chapter was to investigate general design strategies for building touchscreen interfaces for thumb interaction, which I explored by applying competing solutions to the common user task of navigating and selecting among a set of familiar software programs. In choosing to study a real-world application domain, the study outcomes not only lent insight into the absolute and relative values of designing for thumb-sized targets versus offering an indirect gesture command language for navigating interfaces with arbitrary object layouts, but also gauged user interest in notification-based information browsing.

In this chapter I take a similar approach in that I explore the potential of a generalizable one-handed interaction strategy through its application to a specific mobile problem domain. This work was performed in collaboration with the members of the Visualization and Interaction (VIBE) Group during my internship at Microsoft Research in the summer of 2005. I was the principal designer and developer of the interface investigated, as well as the principal designer and administrator of its associated user study. I was the first author on the paper we wrote describing this work, which was published in the 2006 Proceedings of the Conference on Human Factors in Computing Systems (CHI 2006) [77].

5.1 Motivation

In both desktop and mobile computing, it is quite natural that the programs we launch (e.g., email, word processor) and the data objects we access (e.g., documents, web pages) on a regular basis constitute a relatively small and stable set. This is why most operating systems allow users to configure their own application and document shortcuts (e.g, on the Windows and Macintosh desktops and the Windows Start Menu) and may also provide dynamically generated shortcut lists based on application and/or document access frequency (e.g., on the Windows Start Menu and Window Mobile Today screen). The notification-based designs introduced by AppLens and LaunchTile offered a new perspective on the common task of revisiting and monitoring a finite collection of favorite information sources on a mobile device, which may be extended to include applications, documents, web pages, and so on. But not only are mobile devices accumulating more device-resident data such as email, appointments, photos and music, they are also increasingly used as front-end interfaces to external data sets, including web sites, traffic information, and Yellow Pages data. As the volume of data that users can access on and from their mobile devices explodes, a relatively new and pressing issue in mobile computing is how to support users in locating and discovering data of interest from large, potentially unstructured and unfamiliar data sets, given the input and output constraints of mobile devices. Since AppLens and LaunchTile were never designed to scale beyond about thirty-six objects, new methods must be sought.

The traditional desktop approach for managing and navigating personal data sets has been the folder hierarchy. But as data volumes continue to grow, users must either spend increasing time organizing their data into folders and enforcing naming schemes,

or give up entirely. Both methods lead to different challenges when a user revisits her data—either she takes more navigational steps to descend through the folder hierarchy (assuming she remembers where she filed the data in the first place), or she must visually scan through long lists of items. The problems are the same on mobile devices, only that the practical limits of folder hierarchies (and user patience limits) may be reached sooner since small screens can only display a few items at a time.

The answer for the desktop has been to adopt keyword search from the Web domain, which allows users to quickly find highly relevant data from among a vast, unstructured and chaotic information space—not so dissimilar to the environments of many users' personal computers. In the mobile arena, keyword-based search has yet to appear for accessing device-resident data. However, for data residing off the device, many SMS query-answering and browser-based solutions have emerged within the last couple of years [162, 167, 168], including offerings from every major search engine [163, 165, 166, 170].

While the existing mobile search solutions do cater to small screens and low bandwidth, the fact that they are modeled after desktop-based web search poses three main usability issues for the mobile setting. First, they rely on text entry as the method of input, even though the persistent trend toward smaller devices has consistently compromised the efficiency of entering text on devices. Second, the focus has been on searching *off* the device, which ignores the problem of finding data on devices' expanding storage cards, and under-utilizes their increasing processing power which might otherwise be leveraged to offer a more effective search experience. Finally, both the SMS and web search models support directed search tasks, but are less appropriate

for browsing and exploratory search scenarios ("sense-making") that are quite complementary to the mobile setting (e.g., "What is both fun to do and inexpensive around here?").

As an alternative to the existing keyword-based search solutions, we developed FaThumb (pronounced "fathom"), a hybrid browsing and searching interface that de-emphasizes tedious keyword entry in favor of fluid, iterative data filtering through the navigation and selection of hierarchical faceted metadata. Additionally, the low priority that FaThumb places on text entry favors one-handed use scenarios for both keypad and touchscreen-based devices. Although keypad-base phones are already quite easily used with one hand, text entry is notoriously slow because multi-tap requires users to tap keys multiple times to enter the majority of characters, and predictive entry schemes require careful monitoring and occasional correction. Most touchscreen devices, on the other hand, are not designed to accommodate one-handed text entry at all. This fact is apparent from the results of my exploratory web survey, presented in Chapter 2, Section 2.3.3, which found that on average over four times as many respondents who have touchscreen devices use two hands for text entry tasks than use one hand (27% vs. 6%, as a percentage of total respondent population), but that almost twice as many of the touchscreen respondents would prefer to use one hand to two hands (19% vs 11%) for text entry (Figure 7). For either type of device, entering text with one thumb is slow because each letter is entered in pure succession, as opposed to the parallelism offered by using two thumbs on a mobile device, or ten fingers on a full Qwerty keyboard.

The goal of FaThumb, therefore, was to offer a competitive alternative to text-based keyword search that could be performed using localized tapping actions, which

according to my study of thumb movement in Chapter 3, Section 3.2, should be easy to perform with one hand. Our prototype design targeted a keypad-based mobile phone because it is the platform that is most widely used, and yet has the most severe restrictions for text entry. However, I describe how the principles of FaThumb are readily applied to a touchscreen-based interface for one-handed search in Section 5.5.7. After introducing FaThumb below, I describe the user study we conducted to understand the conditions under which a facet-based approach outperforms a keyword-based approach for searching large data sets on a mobile device.

5.2 Related Work

Many information access interfaces present data attributes (metadata) that users can include in queries to large data sets, rather than expecting users to remember them. Dynamic query interfaces [136] encourage iterative composition and refinement of complex queries by providing continuous visual updates of the query results as users restrict the data attributes included in the query. When network latency prohibits dynamic queries, query previews [44] provide an overview of the data to lend insight into distribution characteristics and help avoid queries that return either massive or empty result sets. Many desktop systems (e.g., [47, 160]) have successfully integrated these concepts to improve exploration and understanding of large data sets. FaThumb retains the spirit of these approaches, and maps hierarchically-organized attributes to the phone keypad (or virtual buttons on a touchscreen) in an attempt to offer the greatest access efficiency in the smallest display footprint.

Hierarchical classifications have been used previously in search interfaces. Search results that include hierarchical labels can help users identify relevant items or further

refine (or broaden) searches. Search engines such as Yahoo (search.yahoo.com/dir) and Open Directory (www.dmoz.org) order results by relevance, but display each with a human-assigned hierarchical category; following the category hyperlink allows users to browse via the hierarchical category directory. Other systems help users quickly identify appropriate subsets from voluminous results by organizing results into hierarchical categories [41, 48, 169]. Interestingly, the research of Hutchinson et al. [68] on facet-based browsing techniques to support search in children's interfaces shares a key similarity with mobile device search, which is that text-based keyword entry is difficult for that user population. Their interface is another successful example facet-based search, although their facets were not structured hierarchically because of the problems it presented for children. FaThumb is most directly influenced by Flamenco [61] which couples faceted hierarchical attributes with query previews for use across *all* stages of search: query formulation, refinement and results browsing.

A large number of research solutions have been proposed to facilitate web search from a handheld device, primarily by improving results understanding and navigation for small screens [28, 70, 71, 102], with only [158] designed specifically for keypads. FaThumb instead emphasizes efficient query formulation from a 3x3 grid of buttons, for either keypad-based or touchscreen interaction. Standard bookmarks and saved queries [41] help speed page revisitation, but most systems rely on device-specific text entry methods for ad hoc keyword search. Word prediction and completion algorithms such as Tegic's T9 have the potential to reduce the number of entered characters, but also have the drawback that most fail for non-dictionary words, and may still require users to select from several alternatives. While FaThumb supports keyword search, it favors dynamic

result refinement through navigation and selection of hierarchically organized attribute values.

Unique navigation and presentation techniques are required to accommodate the constrained input and output capabilities of small devices. Robbins *et al.* [121] classified these strategies into three categories in their ZoneZoom design: 1) reformat contents for vertical scrolling; 2) organize content into screen-sized chunks with associated navigation techniques; and 3) use zoomable user interface techniques to navigate data at multiple scales. FaThumb's approach most closely resembles category (2) above, but uses a zooming technique inspired by ZoneZoom for hierarchical navigation: ZoneZoom provides image zooming capabilities by spatially mapping areas of the display to hardware keys; pressing a key causes the display to zoom into the corresponding area of the user interface. Of course for touchscreen-based devices, uses can simply tap the zones directly with a finger or stylus.

5.3 Terminology

5.3.1 Search vs. Browse

Information seeking strategies take many forms depending on the task at hand, user knowledge, and target data set. For example, a user may have in mind a specific document she wishes to find on her desktop. Though the task may be considered one of directed search, her approach will be influenced by whether she remembers the name of the file (in which case she would look for the file by name) or only the period of time in which it was last modified (causing her to look at modified documents within a constrained time span). If the task involves finding a research paper on the web, she

might instead search by some combination of topic, author, or related papers. This approach may appear superficially similar to one she would use if instead she were generating a list of references, although in that case it might be considered a *browsing* activity given its open-ended nature. I will use the following definitions for the purposes of this chapter: a *directed search* is one in which the user knows the precise target in advance, while a *browsing* task is one characterized by specifying criteria that describe a data need, and which may evolve during the search activity. These concepts map closely to what others term *navigational* and *informational* queries for web search [26, 123].

5.3.2 Faceted Metadata

Leveraging data attributes for search has received much attention in recent years. Hearst *et al.* [61] demonstrated the use of data attributes (metadata) organized into orthogonal dimensions (facets) as a means not only to structure search results, but as a tool to guide users in formulating powerful Boolean queries. This approach not only reduces cognitive load through recognition, but allows users to reliably restrict results by attribute values rather than by keyword alone. Hearst's evaluations over both text and image-based data sets have established the efficacy of faceted metadata within integrated search and browse environments.

Although the metadata values within a facet can be organized hierarchically, attribute values need not have a unique classification. This means that the faceted hierarchy is in fact a directed acyclic graph, and that an attribute value may be reached via multiple paths in the hierarchy. The same holds true for each data record, which may be found via any of its attributes. For example, the "Terrier" record may appear in a pet

database under the “Small” value in the “Size” facet, the “Brown” value of the “Color” facet, and the “Dog” value of the “Type” facet.

The FaThumb prototype was developed to explore whether a facet-based search approach could scale down to the small screen and input constraints of a mobile platform, while still providing users the ability to find data within a large data set. In our evaluation of FaThumb, we use a Yellow Pages data set for the Seattle metropolitan area (about 39,000 listings), but the FaThumb design is intended to generalize to a variety of data sets, including personal (contacts, email, etc.) and public (web pages, movie listings, etc.) repositories.

5.4 Data Preparation

Before describing the FaThumb interface in detail, I will describe the data set used in the prototype so that the examples in the following section may be better understood. We obtained a Yellow Pages data table for Washington state with attributes of business name, address, phone number, latitude, longitude and associated Standard Industrial Classification (SIC) code number (http://www.osha.gov/pls/imis/sic_manual.html). For the purposes of investigating a realistically rich data set, we augmented the metadata to include neighborhood, relative distance from four pre-set static locations (a fictional user’s “current” location, school, home and place of work), price ($ to $$$$), rating (one to four stars), hours, and days of operation. Location-based values were computed from the latitude and longitude for each item, while values for the remaining attributes were assigned randomly, with only restaurants receiving price and rating classifications.

Although several classification hierarchies for Yellow Pages data exist [164, 170, 171], they are characterized by large numbers of items at each level, and are thus best

suited for desktop presentation and navigation [83]. So, drawing on existing on-line Yellow Pages resources and intuition, we developed our own attribute classification hierarchy for small screens with the following properties: eight top-level facets, a maximum of eight metadata attributes at each lower level, a maximum depth of five, and just over 200 terminal attributes (to access approximately 39,000 data items). Some attributes were naturally hierarchical (e.g., location, which can be organized by increasing granularity: address, neighborhood/town and region), while for others, a hierarchical structure was imposed as strategy for organizing the data (e.g., relative distance was broken down by type: from me, from work, from school, from home). To ensure Yellow Pages entries could be navigated by business type, we developed a mapping between SIC codes and more user-friendly business types, which were organized hierarchically into a top-level *Category* facet.

5.5 FaThumb Interface Design

The follow sections describe the interface design for the FaThumb interface. Since the FaThumb prototype was built to target a mobile phone, the user interactions are described in terms of the actions that would be performed on a numeric keypad-based device. Bear in mind, however, that FaThumb's fundamental contribution of searching data sets via facet-based attribute navigation has broad relevance to the constraints of a wide variety of mobile devices. As an example, in Section 5.5.7 I describe how nearly the same interface and interaction approach might be adapted for touchscreen-based devices.

5.5.1 Interaction Framework

The FaThumb interface is composed of four distinct structural regions (Figure 27). Among the top three regions (Filter, Results and Facet Navigation) an orange activation border serves to identify the *active* region, which designates the one receiving input from the keypad. In Figure 27a, the Facet Navigation region is the one that is active. Users change the active region by pressing the up or down directional keypad arrows, causing the border to animate to the adjacent region. The bottom region (Menu) is reserved for two menus accessed via associated hardware buttons. The menus' availability and functionality are dependent upon the state of the active region.

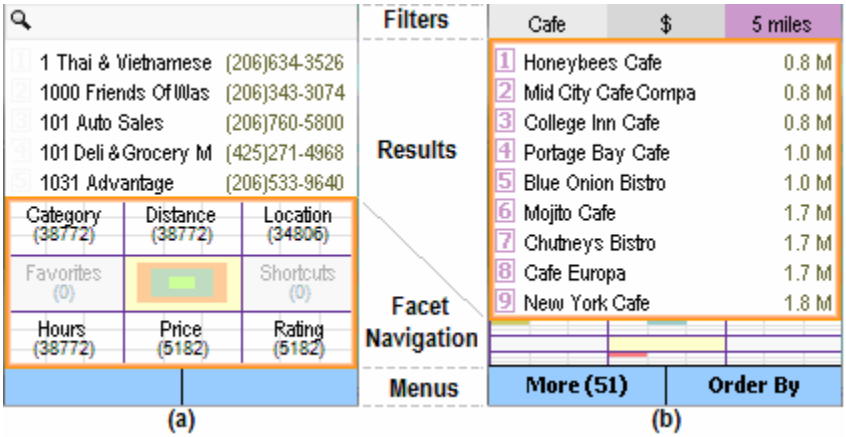


Figure 27. The FaThumb search interface. The default interface configuration (a), and the results for inexpensive cafés within five miles of the user (b).

5.5.2 Facet Navigation Region

The Facet Navigation region represents a navigable tree of hierarchically organized facets, which contains the metadata attributes of the Yellow Pages data set. Each non-leaf node in the tree is depicted as a 3x3 grid of nine zones (Figure 27). Each zone displays a facet or attribute label and a number indicating how many data records in the set can be characterized by the label. Figure 27a displays the root of the facet tree, from which we

see that Yellow Pages listings can be described according to business *Category*, relative D*istance* from both predefined and real-time locations, absolute *Location*, H*ours* of operation, *Price* classification, and consumer *Rating*.

Three zones at the root level have specialized functionality. The *Favorites* and *Shortcuts* facets are reserved for future use. *Favorites* will store entire queries (combinations of selected attributes and/or free-text search terms) the user has chosen to save, and *Shortcuts* will populate dynamically with attribute labels from arbitrary locations in the facet hierarchy based on usage patterns, such as most recently or most frequently used values. The middle zone, on the other hand, is not a facet at all, but an *Overview* zone which serves as a spatial overview during navigation.

Structural Philosophy of the Facet Navigation Region

For mobile devices that lack touchscreens, FaThumb is optimized for keypad interaction. The Facet Navigation region is intentionally designed to map spatially to numbers 1 though 9 on the numeric keypad (e.g., Figure 28a). Conveniently, the 3x3 grid design also provides a compact interaction region of large targets for navigating the facet hierarchy with the thumb when implemented on a touchscreen-based device. While this design restricts the branching factor of the hierarchy (with a maximum 8 at each level), its depth and balance are dictated only by the target data set. For any domain, we believe consistency of facet location is crucial to promoting user mastery of the hierarchy. Thus we opted for a terminating tree, meaning users must return to the top of the tree to explore paths that lead from other top-level facets. On the other hand, as data sets grow, it may be appropriate to dynamically generate nodes within the facet tree to provide more efficient distribution of data among the available zones (e.g., consider date facets labeled

by day for high frequency data, but by year for low frequency data). Dynamic strategies may be most effective at lower levels of the tree, since users may be more willing to browse a less familiar but more useful set of choices once they have already narrowed the data set using a few familiar initial selections.

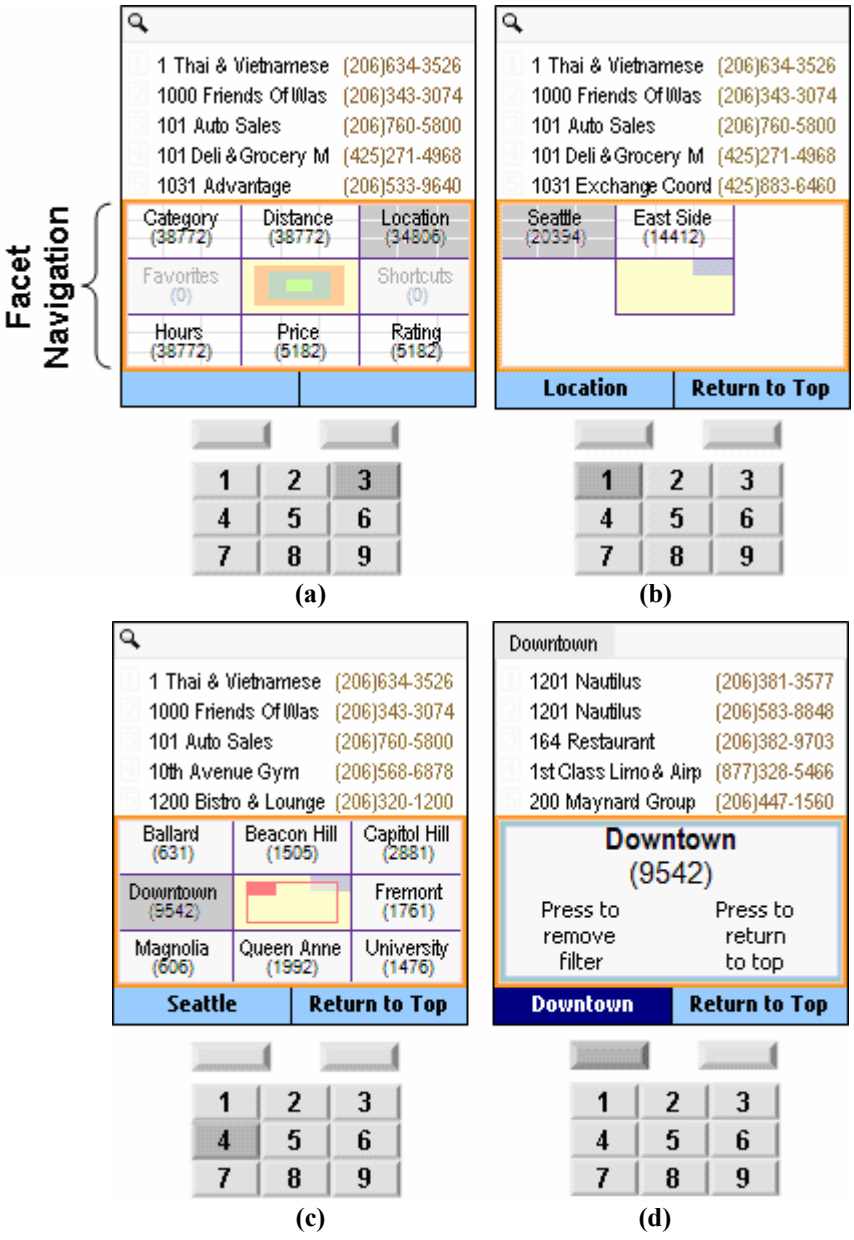


Figure 28. An example of facet navigation using FaThumb. From the root of the facet hierarchy, pressing 3 navigates to Location (a). At the next level, pressing 1 navigates to Seattle (b). Pressing 4 then navigates to Downtown (c), and finally pressing the left menu button adds Downtown to the query (d).

5.5.3 Navigation Region Interaction

Users perform facet-based queries by interacting in the Facet Navigation region. A user navigates the facet space by selecting the zone she wishes to explore within the 3x3 grid. Pressing a number on the keypad (or tapping the zone on a touchscreen) zooms the display into the associated zone: internal tree nodes zoom to the next hierarchical level, while leaf nodes expand to occupy the entire navigation region (Figure 28d). Each traversal propagates navigation state information to the Results region, which serves as a query preview by showing the data set filtered by the selected facet value. A zooming animation accompanies each navigation step.

At each traversal, the left menu changes to display the selected attribute label, both to remind the user of the filter that has been temporarily applied to the data set, as well as to allow the user to add the attribute permanently to the current query; pressing the left menu button adds the attribute label to the top Filter region as a search term (Figure 28d). Pressing the left menu button a second time removes the attribute term from the Filter region. When displaying nodes other than the root, the right menu button serves as a shortcut to the top of the facet hierarchy. Alternatively, pressing the hardware back button that is standard on most devices returns the user to the previous level and updates menus and results accordingly. As an example of query formulation via facet selection, a user interested in inexpensive restaurants in downtown Seattle, might (a) navigate to and select *Location*→ *Seattle*→*Downtown*, (b) return to the root, navigate to and select *Category*→*Restaurants*, (c) return to the root, navigate to and select *Price*→$. Elements of (a) are shown in Figure 28. Since the *Restaurant* facet is subdivided further, for

example, by *Cuisine*, *Style*, *Ethnicity* and so on, the selection in step (b) illustrates that attributes can be saved to the query at any level of the hierarchy, not just the leaves.

Overview

Although navigational context might take many forms (global vs. local, descriptive vs. iconic, etc.) our goal for the initial prototype was to explore designs for reinforcing spatial and motor memory to support users in learning key sequences to specific locations in the face tree.

The *Overview* representation, displayed in the middle zone of the Facet Navigation region, can be thought of as a stepped-pyramid viewed from above; each hierarchical level corresponds to a layer of the pyramid, with the root as the pyramid's foundation. In the root view (Figure 28a) the *Overview* serves as a key for FaThumb's color cues, which assign each level of the facet hierarchy to a unique color. When a user presses a number on the keypad to navigate into a facet, the *Overview* captures the selection by displaying a block whose color corresponds to the current depth, and whose position within the *Overview* corresponds to the spatial position of the selection on the keypad. For example, pressing 3 at the root to navigate to *Location* adds a purple block to the *Overview* in the upper right corner since 3 is in the upper right position of the numeric keypad (Figure 28b). As the user navigates deeper into the hierarchy, smaller levels are added to the pyramid to preserve the user's previous selections (Figure 28c), and so convey the entire spatial selection sequence from the root. This design is able to give both a sense of where the user is in the hierarchy and what command sequence lead to that location.

Visual Encodings

Visual cues help convey fruitful navigation paths to users: leaf nodes are shown in flat purple (e.g., all nodes in Figure 28c), while zones corresponding to internal tree nodes are decorated with a faint grid to indicate the presence of deeper facet levels (e.g., *Location* in Figure 29e). The distribution of data across the zones (depicted as numbers) serves as a preview to encourage or discourage exploration of a path, depending on the task. Zones corresponding to facet tree nodes with no associated data records are de-emphasized using gray text (e.g., *Favorites* in Figure 29c), and cannot be selected. Each facet-based search term added to the Filter region is visually tagged with a colored border in the Facet Navigation region. The visual tags propagate to higher levels when the selected facet is buried within a zone's sub-tree as a way of providing information scent for the query terms (again, see *Location* in Figure 29e).

5.5.4 Results Region Interaction

Once a user is satisfied with the search terms that have been added to the query (any set of facet filters and/or free-text terms), she can explore the remaining records by using the Results region. When the Results region becomes the *active* region, it expands from the original five records to accommodate the first nine records (Figure 29a). The visible items are then numbered from 1 to 9 to support single-key access to listing details. (The current prototype stops short of showing the listing details but users in our study were able to easily extrapolate the behavior.)

The left menu indicates the number of results which follow those displayed, if any (Figure 29a). Pressing the left soft-key displays the next page (up to 9) of results. Once the end of the result list is reached, the left menu allows users to continue from the

beginning of the list. In anticipation of limited bandwidth and user patience, the current prototype only displays the top 200 records. It seems reasonable that users will not want to browse through more than 200 records on a small screen, and so will learn that they can continue to add facet criteria, filter terms, and sorting operations until the desired data are within the top 200. Users can page backward through results using the hardware back button. We chose to not use the up and down directional keys for scrolling of the results because these buttons are dedicated to moving the activation cursor between regions of the interface.

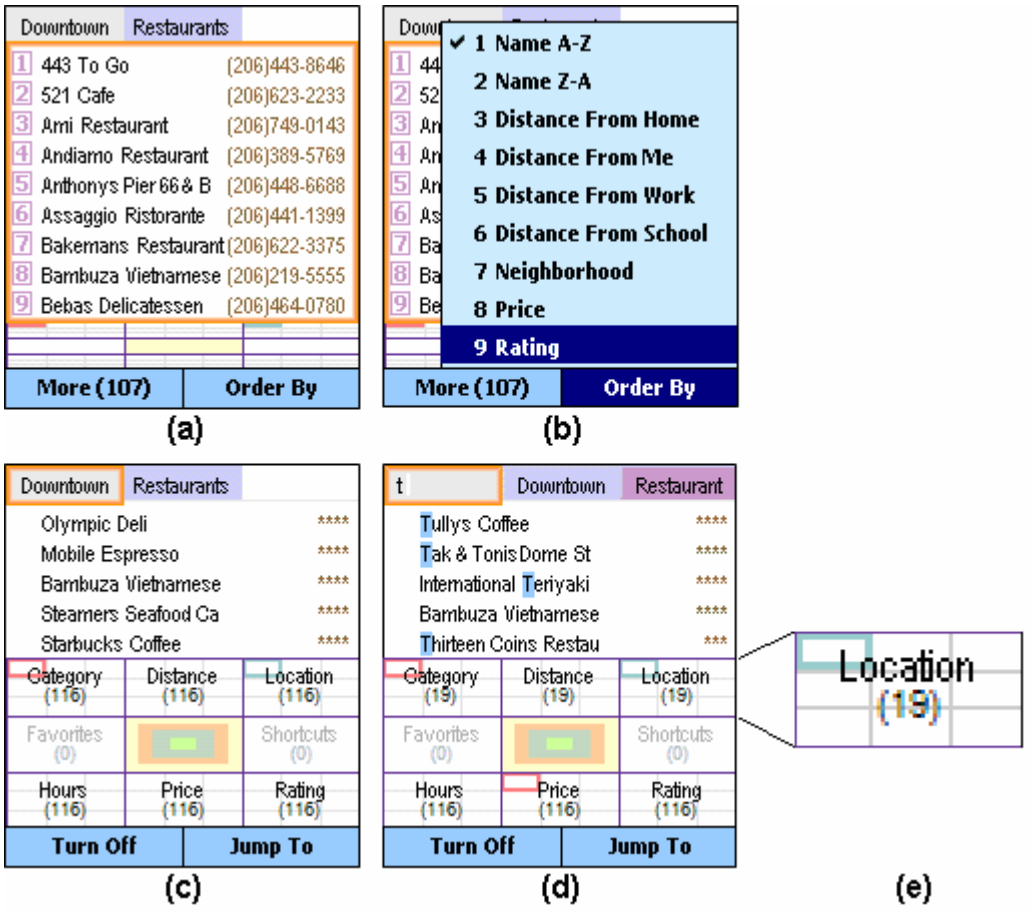


Figure 29. FaThumb results interaction sequence. When active, the results region expands to display up to 9 items (a). The right menu button opens the Order By menu (b); pressing 9 sorts results by Rating. The up arrow moves the activation region to the filter bar (c). Typing in a search term updates the results (d). A facet node in detail (e)s.

A result record consists of two pieces of data: an identifier (which for the Yellow Pages records is the business name) and an attribute (which by default is the phone number). Since data records have not one, but many attributes, users can cycle through the alternative attributes by pressing the left and right navigational arrows.

By default, results are ordered alphabetically by identifier, but they can instead be ordered by other data attributes. Pressing the right soft-key triggers a menu of available order options, which for the Yellow Pages data include: by alphabetical order (A-to-Z or Z-to-A), by relative distance (from user, home, school, and work), by neighborhood, by price and by rating (Figure 29b). A checkbox is displayed in the menu next to the current ordering. Each option is numbered 1 to 9 and selecting the associated number on the keypad reorders the result list and displays the selected sort attribute in each record's attribute column. For example, in Figure 29c, the results have been ordered by consumer rating.

5.5.5 Filter Region

The Filter region holds both the facet-based attributes and free-form text terms, generally referred to as filters, which define the current search query. Filters are displayed in the Filter region as colored, labeled blocks that each occupy one third of the screen width. However, the Filter region can hold an arbitrary number of filters. While a maximum of only three filters are visible at any time, white arrows are displayed along the left and right sides of the Filter region to indicate the presence of off-screen filters.

Facet versus Text Filters

Facet-based filters partition the data into two sets: items with metadata that match the given attribute, and items without. Free-form text filters also partition the data, but

instead by performing a string match over all meaningful record text. Although an data attribute may have a hierarchical "address" describing its position within the facet hierarchy, only the text of the leaf node in the hierarchical path is displayed in the filter bar. However, the entire address from node to root is used as the filter criterion. In contrast, no assumptions are made about the context of a free-form text term, which is matched against a predefined subset of the data and metadata. This approach introduces a fundamental asymmetry between facet selection and text entry in query formulation: selecting "Italian" as a facet filter differs from entering "Italian" as a text filter. In the former case, only Italian restaurants will appear in the result set. The latter query term will include all of the listings of the former, as well as those that have "Italian" in any indexed attribute, such as a business named "Italian Auto Specialists".

Filter Region Interaction

When the Filter region becomes *active*, the leftmost filter is automatically selected and shown with an orange selection border (Figure 29c). The left and right directional navigation buttons move the selection border to adjacent filters within the Filter Region, scrolling to off-screen filters as necessary. Moving the activation border beyond the rightmost filter adds a new (empty) text filter to the query.

As long as the Filter region has focus, users can enter free-form search text using the numeric keypad. If the active filter is a text filter, text will be appended to the existing filter text. If instead the activation border is over a facet-based filter, the existing filters will slide to the right to make room for a new text filter (Figure 29d). Multi-word text in the same filter is treated as a phrase, while multiple separate text filters are treated as separate search terms.

Text Entry

The FaThumb prototype supports multi-tap text entry, whereby each keypad number 2-9 is assigned 3 or 4 alphabetic characters. Pressing a key once enters the first character, pressing twice enters the second character, and so on. A 600ms timeout determines whether a consecutive tap on the same key overwrites the last character or appends a new character. After each timeout, the entire query is reissued and results are updated, effectively providing incremental text search. Substrings in the Results region that match any of the free-form text filters are displayed highlighted in light blue (Figure 29d).

Multi-tap is the original method for text entry using a numeric keypad. While T9 single-tap word prediction has gained popularity, without modification it is unlikely to be effective for search over domains that use proper names (e.g., Yellow Pages) or user defined names (e.g., local data). However, FaThumb can easily be adapted to incorporate the most appropriate text input technique for the data set in question.

Filter Region Menu Options

When the selection border is over a filter, menu options only affect the active filter. The left softkey allows users to preview the removal of a filter. When pressed, the selected filter is disabled and removed from the query: the filter text becomes gray and the Results and Facet Navigation regions repopulate to reflect the new query. Pressing the left softkey again re-enables the filter, again propagating to the Result and Facet Navigation regions. This toggling behavior lets users easily see how the filter narrows the results set. This behavior is especially important when the user is inspecting data with which they are unfamiliar. If the activation border is moved to the Results region while some filters are inactive (grayed), those filters are removed permanently from the filter region and the

current query. Remaining filters then slide left to remove gaps and visually reinforce the current query. Zero-length text filters are also removed.

For facet-based filters, the right menu serves as a shortcut to its position within the Facet Navigation hierarchy. Pressing the right softkey animates the activation border to the Facet Navigation region, at the same time animating a traversal through the hierarchy from the current location to the target filter. The shortcut may be convenient in the case that a user wants to expand a query to explore options at the same level in the facet hierarchy as a previously selected facet. We believe such convenient exploration interaction eliminates the need for an explicit OR query operator.

5.5.6 FaThumb Implementation

FaThumb was implemented on top of Piccolo.NET (http://www.cs.umd.edu/hcil/piccolo/), chosen because it runs on both desktop PC's running Windows and on the Microsoft Windows powered Smartphones and Pocket PCs. Piccolo.NET supports zooming and animated transitions, which are both key design elements of FaThumb. The initial FaThumb prototype ran in a simulated phone environment on a desktop PC, configured with the same display and input limitations as a Smartphone. For the user study, we used an external keypad programmed to have the same key layout as the Smartphone. The multi-clause faceted queries defined by the FaThumb users' interactions were implemented by building and executing dynamically-generated Structured Query Language (SQL) statements for portability across a wide variety of database systems. The initial prototype connected to a desktop-based Microsoft SQL Server database, because support for full SQL (SQL Mobile) was not available on the Windows Mobile Smartphone platform at the time of our investigation. We decided

to use a simulated phone environment on a PC so that we could explore the value of facet-based search even before the availability of hardware to support such sophisticated data processing.

5.5.7 FaThumb for Touchscreen Displays

Although the FaThumb interface was designed with particular care toward supporting indirect interaction through numeric-keypad mapped actions, its principles are quite adaptable to one-handed touchscreen interaction.

(a) (b) (c)

Figure 30. Example screen designs for a thumb-based touchscreen version of FaThumb. The default layout, with the input focus on the Facet Navigation region (a) and the display of results when the input focus is on the Results region (b). When the input focus is on the Filter region, the Facet Navigation region updates to support text entry (c).

One design option, which I present here, is to retain the notions of the activation border and active regions. Users might move the activation border between regions either by using the directional pad, which is standard on most touchscreen devices and shown at

the bottom of the device in Figure 30, or alternatively by using on-screen gestures: ↑ to move the border up a region, ↓ to move the border down a region. In fact, gestures might be used for any action that users would typically perform using the direction navigation pad on the phone, such as using left-to-right → or right-to-left ← finger sweeps to scroll through the data attributes (right column) in the Results region, or move between filters in the Filter region.

When the Facet Navigation region is active (Figure 30a), users can simply tap any of the 8 zones (excluding the *Overview*) to navigate the Facet Navigation hierarchy. Because the 3x3 Facet Navigation grid occupies a relatively compact area of the device, assuming an appropriate hand grip, all zones should be within comfortable reach of the thumb. In addition, the 42 pixel high zones would be both fast and accurate to hit according to the results of my target size study (Chapter 3, Section 3.3) because on 3.5" screens, each zone would be over 1cm high. When the Results region is active (Figure 30b), the 5 results expand in size to make each one easier to hit with the thumb. The additional space created for each item might be used to display additional details associated with the item. Since the user's grip would need to be somewhat centralized on the device for the thumb to reach all 5 items, the soft menu items could be enlarged so that users would only have to reach as far as the bottom of the screen to navigate forward through the results list, as opposed to having to reach all the way to the associated hardware buttons. Finally, when the Filter region is active (Figure 30c), the Facet Navigation region could be repurposed as a standard numeric keypad for entering filter text.

5.6 User Study

We conducted a user study designed to help us understand the value of facet-based navigation compared to text-based search using a 3x3 interaction space. It seemed clear to us that text-based search should be better for certain types of tasks, like looking for a particular business when the name is known. It also seemed clear that facet-based navigation should be better for less well-defined or more complex searches. For example, finding the closest, expensive, Chinese restaurant would be quite hard (or impossible) using standard web-based keyword search because it might only allow users to search on business name and location. Of course, specialized Yellow Pages interfaces do allow users to search on facets like location and neighborhood, but may still not index attributes like rating and price. To make our comparison as competitive as possible, we supported text search over all facet names and attributes except for those relating to distance and hours of operation. That is, users could type “Chinese” and “$$$$” and get a list of expensive Chinese restaurants. The study was designed to test the above hypotheses as well as assess subjective preferences for the two techniques, in addition to collecting first iteration feedback about the FaThumb user interface.

5.6.1 Participants

Participants were recruited with the following characteristics: each must have owned a mobile phone with numeric keypad entry for 3 months and have sent at least 2 text messages a week. Seventeen participants (10 female) ranged in age from 20 to 53 with an average of 29.2 years of age and sent an average of 17 text messages per week. Participants received 2 software gratuities for their time. This study was conducted at Microsoft Research, and all participants granted us their informed consent.

5.6.2 Method

The study design was a 2 (input type: text entry v. facet navigation) x 3 (search type: directed v. simple browse v. complex browse) x 4 trials within subjects design. Input type was counterbalanced across participants, as were the task sets (2 isomorphic task sets were rotated through the conditions such that each task was performed by half the users via text entry, and half via facet navigation). However, search type was presented to subjects in order of increasing complexity, moving from directed to simple to complex browse.

5.6.3 Equipment

The study was run using a desktop simulation of the Smartphone user interface. A 2.8 GHz Pentium 4 Compaq PC with 2G of RAM was used to drive two side-by-side NEC MultiSync LCD 1880SX monitors set at 1280x1024 resolution. Users interacted with the FaThumb interface using an Ergodex DX1 Input System configurable keypad arranged to mimic the Smartphone keypad. A Compaq keyboard and Microsoft Intellisense mouse were used as input for question answering at the completion of each task. The hardware setup for the study is shown in Figure 31.

Figure 31. The FaThumb study equipment setup. The study control software is displayed on the left screen, which was controlled by the Qwerty keypad, and the FaThumb interface is displayed on the right screen, and controlled by the Ergodex keypad.

5.6.4 Tasks

To investigate whether facet navigation could outperform keyword search, even with an unfamiliar hierarchy, task were carefully balanced for query complexity. Under these constraints, we did not expect to faithfully preserve ecologically valid tasks. Tasks were therefore chosen to be representative of those we imagined real users would perform on their phone under various conditions: at work, at home, and while in mobile situations.

Twelve pairs of isomorphic tasks were developed. To be representative, four task pairs were assigned to each of three search types, which increased in complexity based on the number of facet selections or free-text terms required (directed search, simple browse, and complex browse). For example, directed search tasks involved searching for a specific business, while browse tasks presented varying numbers of target attributes. We also classified tasks as "shallow" or "deep" according to the minimum number of hierarchical levels users would be required to visit to complete the task, although we restricted our final analysis to only the search type. Tasks were ordered by complexity (first by search type, then depth) to provide users familiarity with the hierarchy before proceeding to more complex tasks, which might be unfairly penalized otherwise. Example tasks are presented in Table 1. Although task descriptions contained the required search terms, the terms were not emphasized visually in any way and so had to be derived as part of the task. For illustrative purposes here, however, the search terms have been underlined in the table below.

Table 4. FaThumb User Study: Example study tasks.

Search Type	Depth (Levels)	Task Text (query terms underlined here only)
Directed Search	**Shallow** (3)	What is the address of the BlueBottle Art Gallery?
	Deep (4)	What is the address of the Safeway Grocery closest to me?
Simple Browse	**Shallow** (2+2)	What 4-star (****) restaurant is closest to home?
	Deep (4+3)	What seafood places are there in Issaquah?
Complex Browse	**Shallow** (2+3+3)	What inexpensive ($) repair store in Queen Anne is closest to me?
	Deep (3+3+4)	What's the name of the 24-hour pharmacy in Fremont?

5.6.5 Measures

Dependent variables collected during the study included task time, error rate, logs of interactions per task, user satisfaction ratings and overall preference. We also collected observed usability issues and comments from our participants. All measures other than satisfaction ratings and preferences were automatically collected via the logging tool installed on the participant's machine.

5.6.6 Procedure

Participants were run in pairs with the experimenter present. After greeting the participants, the experimenter presented the study procedure and walked the participants through an interactive tutorial, highlighting all key interaction features of the interface design. Upon completing the tutorial, eight practice tasks representing each condition of the study were carried out with the experimenter watching. Once the practice phase was completed (approximately 1 hour), the study proper was begun. Participants performed all tasks relating to either text entry or faceted navigation in a single block of trials before moving on to the other input type.

Study control software was run on the left monitor and the FaThumb interface was presented on the right. A trial consisted of the following steps. After reading the task, users hit a start button to begin task timing. Upon finding the search target, users hit a stop button and entered the answer in a text field. While correctness was not required to move onto the next trial, we did analyze for percent correct after the session. Participants hit a 'next task' button to proceed. Upon completion of all 12 search trials (3 search type x 4 trials) for a given input type, users completed a satisfaction questionnaire and proceeded to the next block of trials. At the end of the entire session, users provided overall preference ratings and general comments. The experimenter debriefed the participants as to the purpose of the study, provided the software gratuity and escorted them out of the building. Total session time was approximately 2 hours.

5.7 Study Results

5.7.1 Task Times

A 2 (input type: text entry v. facet navigation) x 3 (search type: directed search v. simple browse v. complex browse) x 4 trials repeated measures Analysis of Variance (RM-ANOVA) was carried out on the task time data. In order to run the analysis, averages were used to fill cells of missing data. Missing data was a problem because incorrect answers were sometimes provided and could not be included in the task time analysis (~22% of the data). In addition, 4 trials were lost due to system failure and 3 trials were thrown away due to misleading task phrasing. Main effects of search type, $F(2,32)=13.9$, $p<.001$, and trial, $F(3,48)=3.9$, $p=.015$, were observed. Interactions of input x search, $F(2,32)=12.7$, $p<.001$, search x trial, $F(6,96)=4.36$, $p<.001$, and a 3-way interaction of input x search x trial, $F(6,96)=3.7$, $p=.002$, were also observed.

On average, facet navigation was slightly faster than text entry, (62.9s v. 63.9s), but the difference was not significant. As for search type, post hoc analyses with Bonferroni corrections showed that the simple browse condition was significantly faster than either directed search or complex browse at the p=.05 level. The trials did show an increase in complexity as per design, since the fourth trial was significantly slower than the others (no other significant differences between trials were observed).

For an explanation of the search x input interaction, I will refer to Figure 32. Starting with text entry, it is clear that going from directed search to simple and then complex browse tasks slows the user down. This is not surprising since knowing the name of a search target (directed search condition) is advantageous to the text entry task. However, this was not the case for facet-based search task times. The faceted simple browse condition was significantly faster than the others. However, it is also clear that when the task provided a specific target name as in the directed search condition, facet navigation was much slower. In other words, if one knows the search target by name, searching via text entry will be significantly faster than finding the same target using facet navigation. If instead a user is browsing for targets unknown in advance, facet navigation will outperform text entry, with time increasing with task complexity. All other interactions can be explained by the designed increasing complexity of the trials within a block.

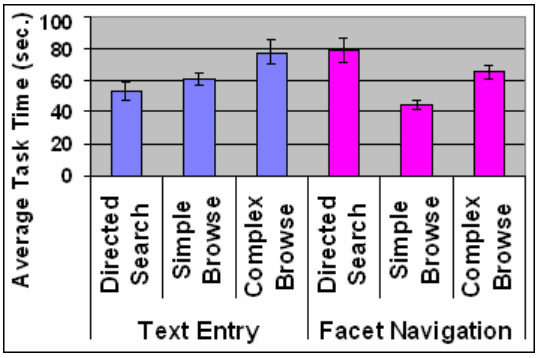


Figure 32. FaThumb Study: Mean task times for each search condition by input type.

5.7.2 Percent Correct

A 2 (input type: text entry v. facet navigation) x 3 (search type: directed search v. simple browse v. complex browse) x 4 trials repeated measures Analysis of Variance (RM-ANOVA) was carried out on the percent correct data from the search trials. No significant main effects were observed at the p=.05 level. However, input x trial, $F(3,48)=4.6$, $p=.007$, and search x trial, $F(6,96)=2.8$, $p=.01$, interactions were observed. The input x trial interaction can be explained by the fact that as trials got increasingly more complex, text entry became more difficult, while faceted navigation did not suffer as much. The search x trial interaction suggests that the simple browse search tasks suffered more from the increasing complexity (depth) of trials than did the other two search task types.

5.7.3 Satisfaction

A 2 (input type: text entry v. facet navigation) x 13 (satisfaction questions) RM-ANOVA was carried out on the satisfaction questionnaire ratings. Significant main effects of input type $F(1,16)=10.2$, $p=.005$, and questionnaire item $F(12,192)=7.02$, $p<.001$ were observed. In addition, a significant interaction between input type and questionnaire item was observed, $F(12,192)=2.4$, $p=.006$. Text entry was rated significantly lower in terms

of satisfaction than was facet navigation overall (average text entry rating = 4.4, average faceted navigation rating = 5.6, with 7 being high). Scales for mental demand, physical demand, and frustration were reversed so that high ratings always indicated positive opinion. Average ratings for each input type are indicated in Figure 33. User ratings were high overall for both input types given this was a first iteration study of FaThumb.

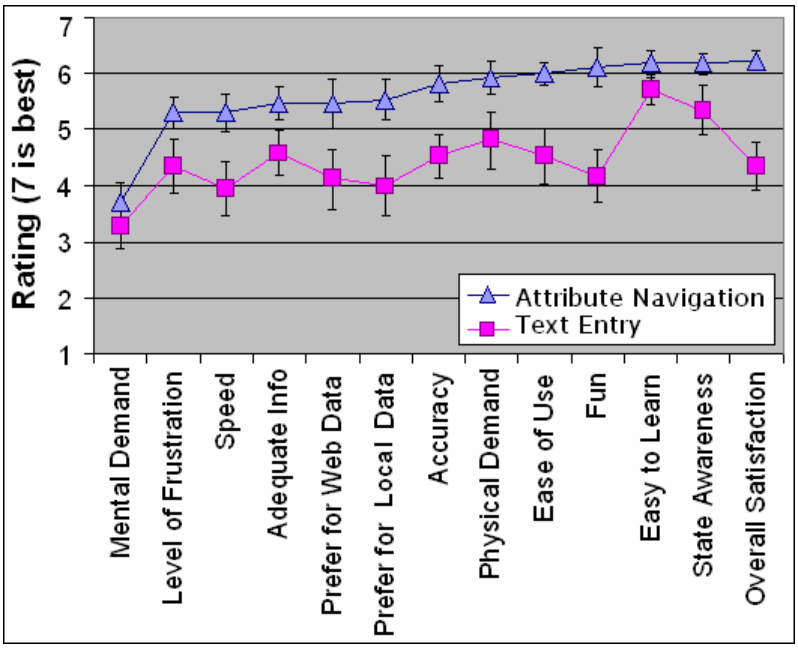


Figure 33. FaThumb Study: Mean satisfaction ratings for text entry and attribute navigation.

5.7.4 User comments

User comments were optional, and those that were made reflected overall positive reactions to the integrated FaThumb system (which included both facet navigation and text entry). We report only on the strongest trends, starting first with those for text entry, then those for facet navigation.

Although two users' comments were strongly supportive of text entry, the remainder (12 users) tended toward neutral or negative. Participants most consistently (5) expressed frustration with text entry when tasks included constraints that were not searchable, such as days or hours of operation. Some (four) offered (correctly!) that text

entry would likely be most effective when a specific business name was known in advance, but would prefer facet navigation otherwise.

By contrast, participant comments suggested general enthusiasm for facet navigation, including that it was easy to learn and use (four participants), fun (four), fast (three) and effective in increasing response confidence (two). The most common negative comment was frustration with identifying the appropriate business classification for task targets. Because the four facet navigation practice tasks provided users only minimal exposure to the hierarchy prior to the timed phase of the study, users were forced to rely on exploration and intuition to locate facets, which at times resulted in lengthy navigation times.

5.7.5 Usability Observations

Participants primarily requested minor feature additions rather than offered fundamental usability suggestions. Thus we draw on our analysis of user error types and experimental observations to better understand usability issues in the FaThumb design. To investigate reasons for error, we reconstructed navigation paths for tasks answered incorrectly, and then classified error types into 5 categories:

Incomplete search criteria (~36%): The majority of errors occurred when users failed to use all search criteria provided, or used incorrect criteria. Since these errors manifested only for tasks that required multiple search terms, the most likely explanation is that complex tasks taxed user memory. Had tasks originated from personal needs, such errors may have been greatly reduced.

Failure to sort (20%): Users sometimes forgot to (or chose not to) sort the data set, which resulted in task error. This suggests that some users do not map tasks involving comparison terms (closest, cheapest, best, etc.) to sorting operations.

Incorrect sort criteria (~13%): While some portion of the sorting errors can be attributed to memory errors, the sort interaction clearly posed a usability problem. To order results, users were required to map a linear list of menu items to the 3x3 layout of numbers on the keypad. Despite visual feedback provided during a key press, users were observed returning to the menu to verify selection. Serial item selection of menu items using the up and down arrows may have improved confidence and reduced errors, but would have been slower. For touchscreen designs, we may see this behavior less frequenty, as tapping items directly with the thumb may instill more confidence in users than the indirect selection required by the phone keypad. Otherwise, a 3x3 grid may be a viable option for the menu design, with the caveat that 2D layout disrupts visual scan, and would thus be more suited to menus with familiar content.

Results scan errors (~16%): When sorting could not be used to place the target item at the top of the results list, participants made identification and copy errors when scanning results. These problems were likely due to the fact that the prototype did not allow users to highlight or select items, which otherwise would have allowed users to visually anchor or verify choices. Item selection within the result list presents similar problems to those for menu item selection. In this case, however, a 3x3 layout is inappropriate since the dynamics of the list make fast linear scan a priority. Using up/down arrows to highlight items is also problematic, since it interferes with their dedicated use in activating interface regions. Fortunately, highlighting an item by

pressing numbers 1-9 has a lower commitment cost than for menus, which close immediately upon item selection. Here, users can verify a selection before pressing "enter" for further details. Again, for touchscreen designs, users could simply tap the desired item directly to see more details.

Business classification errors (10%): The small size of the facet navigation zones limited the length of facet and attribute labels, while the narrow, relatively shallow navigation tree lead to business categories that were quite general. Both issues resulted in classification ambiguity. At the same time, some facets were arguably misclassified (Salons were located under Category→ Shopping→ Personal→ Hair, rather than Category→ Services→ Businesses). Iterative refinement of the hierarchy and user familiarity would certainly reduce these errors, but they are likely to be typical of errors encountered for any new classification hierarchy. This makes it all the more encouraging that users provided such strong preference for facet navigation.

5.8 Discussion

Based on user feedback and our own observations, we found users adapted quickly to the facet navigation and selection, suggesting the spatial arrangement and hierarchical structure of facets holds great potential for search. However, users were at times frustrated when items were not classified as expected, and some either did not understand the role of sorting for search tasks or did not understand how sorting was supported in FaThumb.

FaThumb's primary challenge may be its generalization to alternate data sets. Given the positive response to facet navigation, successes must have either been more common than failures, or otherwise were more gratifying than user mistakes were

frustrating. Even so, first-time users struggled with the classification scheme for some tasks. We would like to extend the FaThumb paradigm to a wider variety datasets such as email, contacts, appointments, photos, cached web pages, traffic information, stocks, music collections, or movies listings, in order to understand the elements of classification structures that promote satisfying user experiences.

Another potential concern is that some users did not sort the result list when necessary. One user gave insight into the problem when he commented he would have liked "closest to me" as an attribute in the facet tree. This suggests some users may not view attribute selection and sorting as fundamentally distinct operation types. We would therefore like to investigate whether integrating a subset of sort operations into the facet space improves performance for appropriate classes of tasks.

We designed FaThumb as a search solution for devices with limited input and display capabilities—in particular keypad-based mobile devices. While we have demonstrated the viability facet-based search, technical limitations prevented us from validating the design under mobile scenarios. It would be important to port the prototype to a mobile phone and study FaThumb in the field with up-to-date databases either stored locally or accessed remotely via GPRS. Another goal is to implement FaThumb for touchscreen interfaces and study whether the same benefits seen for keypad based interaction apply for one-handed thumb-based search on a touchscreen.

5.9 Conclusion

The FaThumb interface was developed to explore an efficient solution to searching large data sets from mobile devices because the most common approach of keyword-based text entry tends to be quite tedious. By emphasizing search through the navigation and

selection of data attributes within a compact 3x3 interaction region, the FaThumb design adheres to my own design principles for supporting one-handed device operation, and represents a strategy that can be applied to both keypad-based and touchscreen-based devices. Results from the user study indicate that there are indeed tasks for which the facet navigation approach we developed can out-perform keyword entry for the important task of searching the ever expanding data sets available from mobile devices. In particular, the user study confirmed our basic hypotheses—if a user knows something specific about a data item, like its name, text entry (even using multi-tap) is faster. However, if the user knows only data characteristics, facet navigation is faster. Ultimately, both techniques are needed because real world tasks require both. Comments made by study participants suggest that when users become familiar with the taxonomy, the FaThumb facet navigation technique is highly desirable. These results have successfully demonstrated the potential the FaThumb approach holds for increasing user productivity and satisfaction when using mobile devices to locate and discover data in large, possibly unfamiliar, repositories. The benefit that FaThumb's design only requires one hand for both keypad and touchscreen devices further encourages its utility and viability because it supports search in mobile settings, where text entry can be a detriment to users due to its high physical and mental demands.

Chapter 6
Applications: A Technique for Generalized One-Handed Interaction with Touchscreen Interfaces

The research of Chapter 4 investigated general design strategies for supporting thumb use of touchscreen interfaces. The LaunchTile system presented in that chapter, as well as the new iPhone interface, both take the approach of ensuring interface targets are large enough to be hit directly with a finger or thumb. However, there are practical reasons why this type of "lowest common denominator" approach to mobile touchscreen design is unlikely to be the dominating strategy for future interfaces. First, screen real estate is a precious resource for small devices, and placing limits on visual expressivity can hurt the design in other ways. As an example, increasing object sizes to accommodate thumbs means fewer objects are displayed per screen, so more screens are required to present a given amount of data. This may not be just an annoyance when using a thumb, but can unnecessarily slow information access when two hands *are* available. Second, today's mobile UI toolkits tend to support only small, stylus-oriented widget palettes, and so touchscreen software developers are likely to be offering stylus-oriented interfaces unwittingly for years to come, or otherwise take it upon themselves to develop or seek out thumb-oriented toolkits. Finally, the vast majority of existing touchscreen software is designed for stylus use, and so users of these interfaces might benefit greatly if the interfaces could be enabled for one-handed use. Our goal with this work, therefore, was to consider an alternative to the strategy of building interfaces with large finger-sized targets, and instead allow designers to focus on the most effective ways to present and

navigate information regardless of users' hand availability, and in so doing, extend thumb-based support even to rich, stylus-oriented touchscreen designs.

In this chapter I present ThumbSpace, an interaction widget which aims to address both the reach and accuracy problems that users experience when operating touchscreens with thumbs. Its design is inspired by the substantial body of research that has focused on the challenge that distance plays in large display interaction. The ThumbSpace design reflects our interpretation of how solutions for accessing objects out of arm's reach on large displays can be adapted to the problem of accessing objects out of thumb reach on handheld devices.

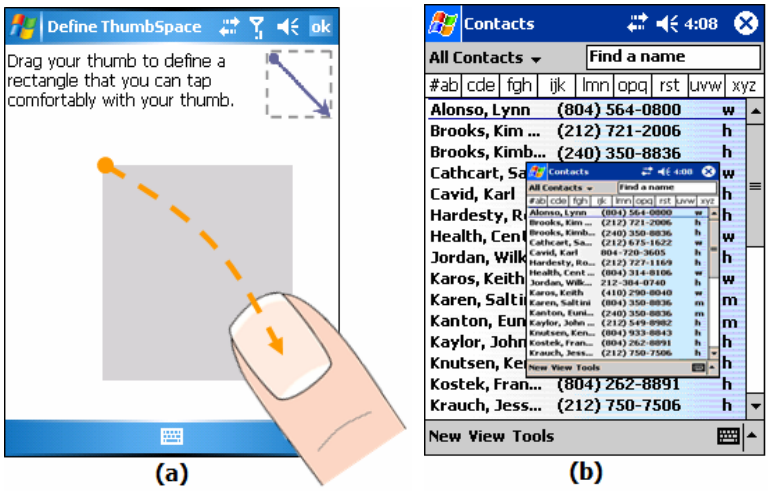


Figure 34. An example of how users define a ThumbSpace (a), and an example of a ThumbSpace depicted as a Radar View (b).

Conceptually, ThumbSpace serves as an absolute touchpad superimposed on the display, and to which all screen objects are mapped (Figure 34b). Reach limitations are addressed by allowing users to personalize the size and placement of ThumbSpace (Figure 34a), thereby accommodating individual differences in hand preference, geometry, motion range, grip, and use scenario. Not only does ThumbSpace support access to all display objects within an accessible portion of the screen, but we apply

dynamic visual feedback and selection tuning to alleviate high error rates that are typical in hitting small targets with big thumbs. Finally, ThumbSpace is only available upon user activation, and so is unobtrusive when not wanted.

At the core of our design is an assumption that users of touchscreen devices will be compelled to touch the interface, even when not using a stylus. But problems of high visual demand, targeting accuracy and finger occlusion suggest peripheral hardware solutions might be more appropriate. For example, Blackberry devices have shown how thumb wheels and trackballs can be used quite effectively for controlling a non-touchscreen device. After presenting the ThumbSpace design, I present a study which compares the efficacy of ThumbSpace to alternative touchscreen techniques and to peripheral hardware input methods to better understand the tradeoffs of these approaches. I then present a follow-up study that investigates the relative benefits of using ThumbSpace for a device with a larger screen than the one featured in the first study (2.8” vs. 3.5”).

I was the sole investigator of this work, which was conducted between September 2006 and August 2007, and supervised by my advisor. The design and evaluation of the earliest prototype of ThumbSpace is published in the Proceedings of the International Conference on Human-computer Interaction (INTERACT 2007) [74].

6.1 Related Work

6.1.1 Finger Operation of Touchscreens

Many current touchscreen interfaces consist of widgets similar in size and function to those featured on a desktop PC. While acceptable for interaction with a 1mm stylus tip, my own research suggests touchscreen targets smaller than 9.6 mm [112] can result in

unacceptably high error rates when accessed one-handed with the thumb (see Target Size Study, Chapter 3, Section 3.3). Recently, Vogel and Baudisch [147] developed the Shift technique as an improvement over Sears and Shneiderman's offset cursor [131], a highly regarded approach to precision touchscreen targeting for over 15 years. The offset cursor couples a selection cursor positioned off the tip of a user's finger with a stabilization algorithm to achieve character-level selection accuracy. The downside to the offset cursor is that users have to aim *below* the intended target. Shift instead allows users to aim directly for the target, and after a variable delay, displays a portal view (callout) of the screen area covered by the user's finger. The callout includes a crosshair cursor to show the position of finger contact, which users can adjust before lifting their finger to perform selection. The delay function that determines when to show the callout is proportional to the size of the target under the finger—short for small targets, and longer for large targets. The result is that Shift is only shown when users are likely to need it, and does not interfere with selection otherwise.

While Shift holds great potential for one-handed selection of targets within reach of the thumb, further investigation is necessary to understand whether pixel-level selection is appropriate under mobile conditions, and whether Shift works equally well for objects along the perimeter of the screen, which occur frequently in today's designs. ThumbSpace, on the other hand, is expected to support targets at edges just as well as those away from the edges. More importantly, Shift was designed for two-handed index finger operation of mobile devices, and so does not address the limitations of thumb reach that ThumbSpace does.

6.1.2 Reaching Distant Objects

ThumbSpace draws its inspiration from table-top and wall-sized displays, which both confront problems with out-of-reach interface objects. A general problem in large display interaction is that the increase in real estate also increases the average distance between on-screen objects. Unfortunately, Fitts' Law [50] dictates that increasing travel distance without a commensurate increase in target size will increase access time. Solutions have thus typically focused on 1) decreasing movement distance to targets and/or 2) increasing target sizes.

Improving target acquisition for mouse-based interaction has often involved clever manipulation of the control-display (CD) ratio. Slowing mouse movement over interaction targets (Semantic Pointing [21]), jumping the cursor to the nearest target (Object Pointing [54]), and predicting the user's intended target (the Delphian desktop [7]) are three such examples. The drawback of these techniques is that their effectiveness decreases as the number of nearby objects increases. Other approaches in smart cursor control make targets easier to hit by increasing the cursor size, such as the area cursor [72] and Bubble Cursor [53]. Unfortunately, these techniques are not directly applicable to touchscreen interaction; touching the screen directly means a 1:1 correspondence between motor and display movement, so there is no CD ratio or cursor to manipulate.

Direct screen interaction with fingers or pens is common in tablet, mobile, and wall computing. Techniques to improve object access speed in these arenas have focused on minimizing the movement distance to targets. However, most research that focuses on icon placement or selection [12, 20, 60] is inappropriate for PDA interfaces because drag and drop and object placement are used much less frequently than interactions such as

tapping buttons, links, and check boxes. Another approach has been to provide a nearby miniaturized version of the display, or Radar View [108], that can be manipulated directly. However, both the Radar View and the pen-based extension to the Bubble Cursor, the Bubble Radar [2], again focus on object placement tasks, rather than general application interaction.

6.2 ThumbSpace Design

Our goal with ThumbSpace has been to develop an interaction strategy whereby rich touchscreen interfaces can be effectively controlled with a thumb, without sacrificing the expressiveness of information presentation or the efficiency of navigation when two hands are available.

Given individual variations in thumb agility, hand size, strength, and usage scenario, the first principle of ThumbSpace is to support each user's most comfortable and stable grip. Each user therefore defines her own ThumbSpace—a region of the touchscreen surface that she considers easy to reach and interact within. The user may redefine the position and size of ThumbSpace anytime after an initial configuration step in which she drags her thumb along a diagonal to define the upper left and lower right corners of a rectangular region (Figure 34a). All thumb interaction then occurs within this personalized ThumbSpace, which remains fixed across all applications.

To support access to all interaction targets within the confines of the ThumbSpace, the region behaves as Radar View. Consider, for example, a Radar View applied to the Windows Mobile Contacts application in Figure 34b. We see that a simple version of this approach would have several problems: 1) the Radar View representation occludes a large number of DisplaySpace objects; 2) the Radar View features are

unreadable; 3) the detailed Radar View representation contributes to visual clutter; and 4) the Radar View objects are far too small to hit reliably with the thumb.

6.2.1 Initial Design

Our first ThumbSpace design addressed problems (1-3) by avoiding the use of a miniature representation entirely. Instead, we offered only a whitewashed region to visually suggest where the user should focus her attention (Figure 35a). In this design, ThumbSpace was visible at all times. Although no miniature representation was shown, ThumbSpace behaved as a Radar View by honoring an input mapping between the ThumbSpace and the original display. ThumbSpace was partitioned so that each object in the original display was associated with a sub-region (proxy) in the ThumbSpace; tapping a proxy in ThumbSpace selected the associated object in the original display. Assuming the ThumbSpace represents a linear scaling of the original display (e.g., Figure 34b), the partition of ThumbSpace into proxies would be that of Figure 35b.

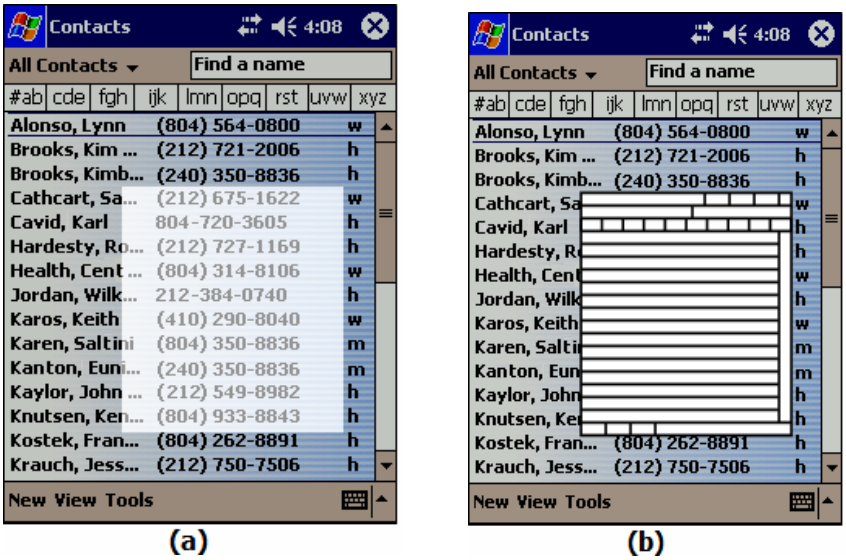


Figure 35. The first-iteration ThumbSpace representation (a), and a possible partitioning of the ThumbSpace into proxies for a Contacts application (b).

The final challenge (re: (4) above) in using a miniature representation as an interaction widget is that the proxies will likely be too small to hit reliably with a thumb. This is true even when the users can see the representation (as in Figure 34b), but our initial ThumbSpace design introduced further uncertainty because it failed to provide visual cues for how its sub-regions mapped to display objects. ThumbSpace managed these uncertainties by providing a visual feedback loop during object selection.

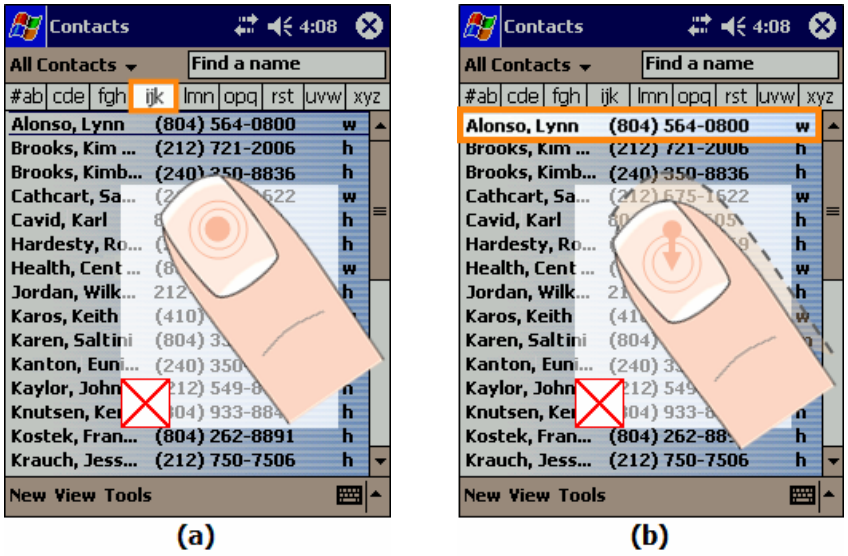


Figure 36. An example of how a user selects an object using ThumbSpace. Assuming the user wants to select the first name in the Contacts list, she first aims at the ThumbSpace proxy for 'Alonso' (a). Since the user's initial ThumbSpace contact point maps to 'ijk', she adjusts the selection by dragging her thumb toward the bottom of the screen (b). The user confirms the selection by lifting her thumb.

Object selection with ThumbSpace is performed in three phases: *aim, adjust,* and *lift*. The *aim* phase relies on users forming a mental model of the mapping between the ThumbSpace proxies and display objects, with the natural assumption being that the ThumbSpace represents a linearly scaled version of the original display. Based on this model, the user touches the sub-region of the ThumbSpace she believes best corresponds to the intended target (Figure 36a). In the *aim* phase, ThumbSpace can be likened to an absolute touchpad—if the user guesses correctly, ThumbSpace provides direct access to

objects that would otherwise be difficult (e.g., too small) or impossible (e.g., out of reach) to hit directly with her thumb.

Once the thumb touches a ThumbSpace proxy, the associated display object is visually identified with an object cursor, depicted by a thick orange border. The user then enters the *adjust* phase of selection. During the *adjust* phase, ThumbSpace acts like a relative touchpad for controlling the object cursor. If the user rolls or drags her thumb more than a fixed number of pixels up, down, left, or right, the object cursor animates to the closest display object in the direction of movement (Figure 36b). In the *adjust* phase, ThumbSpace interaction is similar to Object Pointing [54] in desktop environments, which ignores the white space between interface objects, and instead jumps the mouse pointer to the nearest object in the direction of movement once the pointer leaves an object's border. Our *adjust* strategy differs slightly from Object Pointing because the *adjust* threshold is independent of the display object sizes.

Finally, the user confirms the selection by lifting her thumb. This manner of object selection is inspired by the lift-off strategy for touchscreen object selection developed by Potter [116], which allows users to visually confirm and adjust a selection before committing to the action. Our initial ThumbSpace design allowed users to cancel a selection by dragging her thumb over a red X, which would appear throughout the *adjust* phase in the corner furthest from the point of initial thumb contact.

6.2.2 Evaluation of Initial ThumbSpace Design

To understand the efficacy and usability characteristics of our initial design, we conducted a quantitative study to compare ThumbSpace to direct thumb interaction for accessing targets. Sixteen participants (8 male, 8 female) used both techniques (direct

touch and ThumbSpace) to select objects at two sizes (small: 20 px vs. large: 40 px), density (sparse vs. dense), and position (near vs. far).

As we report in [74], results showed that overall error rates were comparable between the two techniques, but that participants were consistently slower using ThumbSpace. Even so, ThumbSpace made clear progress toward two of our design goals: 1) "decouple target size from thumb size"—users were as effective at selecting small targets as large targets, evident from the fact that neither speed nor error data were affected by target size; (2) "improve selection for targets that are out of thumb reach"—users accessed far targets more accurately using ThumbSpace than direct interaction. Finally, users felt ThumbSpace held promise, giving it relatively high ratings on a 7-point scale for task execution satisfaction (5.1), fun (5.4) and learnability (5.4), and the majority of users indicated that ThumbSpace would be preferable to direct interaction after sufficient practice. These results encouraged us that a strategic redesign could have a strong impact on closing the performance gap between ThumbSpace and direct thumb input.

6.2.3 ThumbSpace Redesign

Because ThumbSpace was inspired by the challenge of accessing all areas of a touchscreen while using a device with one hand, our initial design inadvertently *favored* targets that were out of thumb reach. This is because the ThumbSpace itself interfered with near targets, making them more difficult to hit. But our ideal solution should make hard tasks easy, without negatively impacting tasks that are already easy. Our response was to allow users to trigger ThumbSpace on-demand. In this way touchscreen operation will be no worse than direct thumb interaction when users choose not to trigger

ThumbSpace, but for targets that are hard to access with the thumb (e.g., small or far), user have the option of using ThumbSpace. We chose to use a hardware trigger because it is reliable and unambiguous. Users might choose any of the re-mappable hardware buttons on their own device, but for our test platform, the Cingular 8525, we used the center of the directional navigation pad (e.g., "enter") for its positional convenience.

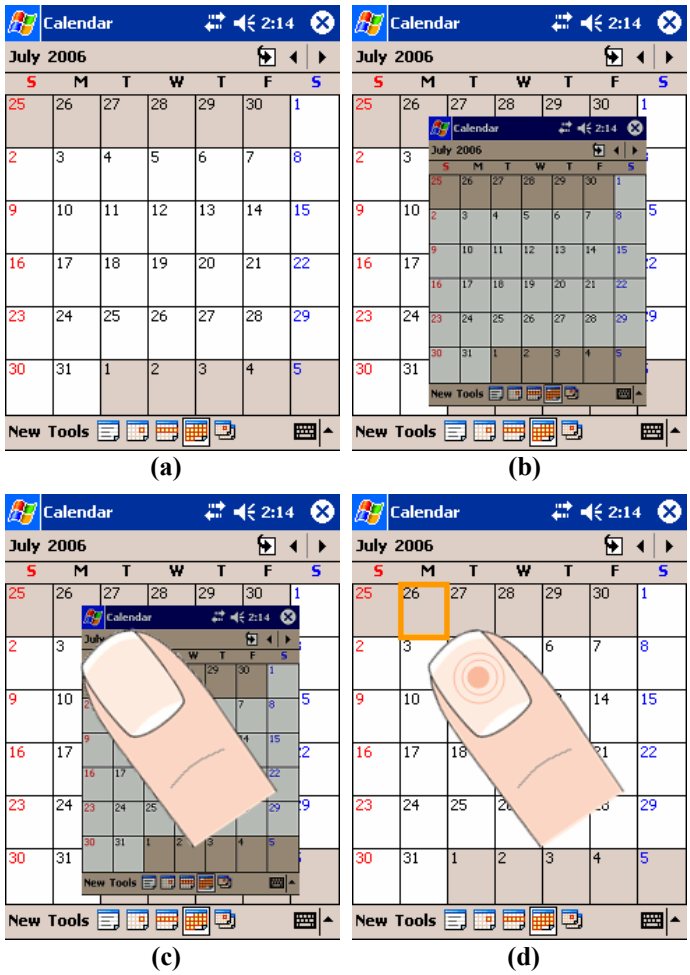


Figure 37. An example of selecting an object with the final ThumbSpace design. A calendar application (a). Pressing the middle button of the directional navigation pad launches a mini representation of the calendar within the user's predefined ThumbSpace (b). If the user wants to hit June 26th, she aims for the 26th in the mini representation (c). Upon touching the screen, the display object associated with the proxy user hits is highlighted and ThumbSpace disappears (d). The user may drag her thumb to adjust the input cursor, or lift her finger to perform the selection.

The second goal of our redesign was to reduce object selection time. Observations and interaction logs both suggested that our study participants did not make "good"

guesses in the *aim* stage of object selection (Figure 36a), and so spent considerable time making adjustments (Figure 36b). To reduce time spent in the *adjust* phase, we needed to support users in making better guesses in the first place, which for us meant providing better visual cues. We reasoned that the most common use of ThumbSpace would be to access targets that are out of thumb reach. Since ThumbSpace is by definition *within* thumb reach, it would not interfere with such targets, and so could be used to provide the necessary visual cues.

Our new design used a faithful miniature of the display as the ThumbSpace (Figure 37b). However after the *aim* phase, the ThumbSpace disappears to avoid interfering during selection adjustment. In the prototype of the new design, we did not implement a cancel feature, although cancel could be provided in the future, for example, by moving the selection cursor off the screen in any direction.

Our final design adjustment was to constrain each user's ThumbSpace rectangle to the same aspect ratio as the display itself. In our initial design, users were allowed to define a ThumbSpace with arbitrary dimensions, which we believed made the mental transformation between ThumbSpace and the display more challenging than necessary. Matching aspect ratios can help users locate ThumbSpace proxies quickly, because visual cues are reinforced by spatial memory, rather than requiring mental gymnastics.

6.3 Direct Touch vs. Peripheral Input Hardware

Our fundamental assumption in developing ThumbSpace was that for interfaces designed for stylus input, tapping targets directly with the thumb would be faster and more natural than using peripheral input hardware, such as scroll wheels, trackballs or directional navigation pads. But the popularity of RIM's Blackberry, which uses a thumbwheel or

trackball almost exclusively for object selection, suggests either that this assumption is incorrect, or that the Blackberry software and interaction experience have been optimized for the indirect navigation devices. While it is certainly true that thoughtful hardware/software coupling can offer powerful advantages in task efficiency and user satisfaction (which then would not be true of the arbitrary touchscreen interfaces we wanted to support), there are several key reasons peripheral hardware input might be more appropriate than touchscreen interaction for one-handed mobile computing.

6.3.1 Stable One-Handed Operation

Limiting information navigation and selection activities to a single hardware component that can be comfortably accessed with a one-handed grip (as the scroll wheel does) not only frees the other hand for additional activities that arise in a mobile setting (such as carrying bags), but also means users can operate the device without changing their grip, even when accessing information across a variety of applications. If the hardware is strategically located so that the grip is stable, users can perform information tasks while mobile without risk of dropping the device, which otherwise diverts physical and attention resources that are more appropriately directed toward navigating the environment.

6.3.2 Occlusion

In addition, using hardware on the periphery of the device keeps the screen free from visual interference. This is in contrast to interaction methods that involve touching the screen directly, which must contend with finger and hand occlusion. Since today's devices are equipped with standard resistive touchscreens, they can accept only one input

point at a time; if the screen is touched in more than one place, only the average point is registered as the input point. When using a stylus, this characteristic is not typically a problem since a stylus has a small tip for precise targeting, and offers a thin, low profile extension to the hand that keeps bulky fingers away from the screen. Since fingers have much larger contact areas than styli, users must aim at targets with the center of their finger pad, which can then completely cover small targets. In this case, the average input point may fall outside the target bounds and generate a selection error.

6.3.3 Reduced Visual and Mental Demand

Object selection with direct touch involves a unified process of continual visual monitoring and physical adjustment. This can be a detriment for mobile settings as users may need to divide their visual attention between the device and the environment. Input via peripheral hardware, on the other hand, has two separable components: *aim* and *selection*; aim positions a cursor, and selection activates the object under the cursor. While the aim phase *may* require continual visual monitoring, as when using a mouse, it does not have to. Scroll wheels, trackballs and directional pads convey abstract commands, rather than position information, to move the cursor from one selectable object to the next. With these methods, the cursor position depends only on the number and type of commands issued, so constant visual monitoring is unnecessary if the user preplans the command sequence to translate the cursor to the desired target.

Another advantage of peripheral hardware is that the state of the cursor is maintained between commands. This allows the aim phase to be interleaved with myriad stimuli that compete for the user's visual and mental attention in mobile computing. Furthermore, cursor stability supports high precision selection. Finally, by lowering the

visual, mental, and physical demands of device interaction, peripheral hardware solutions effectively push device activities to the background, which is a positive influence on the safety of mobile users and others surrounding them.

6.4 Study 1: Direct Interaction vs. Peripheral Hardware

Countering the numerous advantages that peripheral hardware input enjoys over touchscreen interaction for one-handed mobile device operation, there are challenges as well. Cursors must move serially from object to object. Each of these movements takes time away from the ultimate goal of selecting a target. This characteristic gives direct touch a potential advantage in average interaction speed. Of course both speed *and* accuracy influence user satisfaction, so our goal is to understand the relative advantages of peripheral hardware vs. touchscreen input methods for task speed, accuracy, and satisfaction for one-handed device use.

6.4.1 Independent Variables

Input Methods

In selecting input methods to include in our investigation, we wanted to compare ThumbSpace to the common alternatives used with today's devices. For peripheral hardware we chose the scroll wheel (*ScrollWheel*) and the directional navigation pad (*DPad*)—ScrollWheel for being the original distinguishing feature of the non-touchscreen Blackberry devices and DPad because nearly every cell phone has one. It was clear that for touchscreen interaction, we would compare ThumbSpace to direct touch (*DirectTouch*), since that is how users operate touchscreens one-handed today. We assumed ThumbSpace would offer users more accurate targeting at the expense of speed,

so also chose a technique that specifically addresses targeting accuracy for fingers on touchscreen. Recently, Vogel and Baudisch [147] showed users made fewer errors using their Shift technique over direct touch for hitting targets ≤ 2.9 mm, but not ≥ 4.3 mm. Given the possibility that Shift would help users in hitting the small targets (3.6 mm) used in our study, we included it as a more competitive variant of DirectTouch.

Mobility

Many previous studies have established the negative impact movement has on mental demand and task performance (e.g., [111]). If our assumption that the hardware input methods are more stable and require less mental, visual, and physical demand than the touchscreen methods, then increased activity would be expected to degrade performance more when using touchscreen methods than hardware methods. To understand this relationship, we studied users performing tasks while both standing and walking. During the walking condition, users chose a comfortable walking pace along a 19' x 7.5' tape figure eight.

Target Sizes

Our initial study of ThumbSpace [74] confirmed that user performance was independent of target size, so here we chose only a single target size of 20x20 pixels (3.6 mm^2), representative of standard Windows Mobile widgets (e.g., checkboxes: 15 px, buttons: 21 px, text boxes: 19 px).

(a) (b)

Figure 38. ThumbSpace Study 1: The input regions shown classified as "hard to reach" in dark gray and "easy to reach" in light gray (a), and an image of the Cingular 8525 used in the study.

6.4.2 Tasks

Tasks were based on selection activities that would typically be performed with two hands using a stylus. Since our goal was to understand appropriate one-handed interaction techniques for rich interfaces we selected a two-dimensional input space of arbitrarily placed objects. Potential targets were placed within a 6x8 grid of 40x40 px^2 cells. We chose this number of targets because it is comparable to the number of targets offered in some standard applications that have 2D layouts, such as the month view of Windows Mobile Calendar (52 objects). For analysis purposes, we partitioned the targets into a 3x4 grid of regions, 4 targets per region (Figure 38a). Regions were labeled "easy to reach" (light gray) or "hard to reach" (dark gray) based on the majority opinion of study participants.

Because the targets were smaller than their assigned cells (20 vs. 40 px), the target for each trial was placed in the center the designated cell. All other distracter targets were randomly assigned one of two sizes (20 px and 13 px, with probabilities 0.2 and 0.8 respectively), and were positioned at random locations within their cells. The randomized

locations were used to create the illusion of a non-uniform layout space, while the variable sized objects increased the percentage of the background displayed. The targets were placed on a map background because we expected context to be useful during use of the Shift technique.

Tasks were presented as a dialog that indicated the trial target and the action to take to begin the task (Figure 39a). To help users distinguish the goal target from among others, its center was colored yellow. The message was always placed at the center of the user's personal ThumbSpace, unless it overlapped the target, and then was placed either above or below the target. For ScrollWheel, users pressed the scroll wheel to begin; for DPad and ThumbSpace, users pressed the center of the directional pad to begin; for DirectTouch and Shift, users tapped the dialog to begin.

Figure 39. ThumbSpace Study 1: Representative screen shots from the experiment. An example of a task instruction (a). Performing a task using ThumbSpace (b). Performing a task using Shift (c).

A rectangular orange cursor was used as the input focus. For ScrollWheel and DPad, the cursor started at the upper left target, since this is a typical home position for cursors in today's interfaces. For DirectTouch, Shift and ThumbSpace, the cursor only appeared once the user had touched the screen to perform a selection. When a target was

selected, the input cursor would animate toward the midpoint of the target and vanish to provide visual feedback, while an audio sound also indicated trial success or failure.

6.4.3 Hypotheses

Based on intuition and our own previous research suggesting users are faster and more satisfied when interacting with a compact region of the device (see the Thumb Movement Study of Chapter 3, Section 3.2) [75] , we had the following hypotheses regarding the touchscreen interaction methods.

(H1) DirectTouch would be faster than either Shift or ThumbSpace; (H2) ThumbSpace and Shift would be more accurate than DirectTouch; (H3) ThumbSpace would be faster than Shift in the "hard to reach" regions (Figure 38a); and (H4) Shift would be faster than ThumbSpace in "easy to reach" regions.

In comparing peripheral hardware input methods to the touchscreen methods, we hypothesized: (H5) DPad and ScrollWheel would be more accurate than DirectTouch, Shift, and ThumbSpace; (H6) Shift and ThumbSpace would be faster on average than both DPad and ScrollWheel; (H7) DPad and ScrollWheel would be less impacted by walking than the touchscreen methods.

6.4.4 Implementation and Apparatus

As a real-world input system, ThumbSpace will need to cooperate with a PDA's operating system in order to capture and reinterpret thumb events. However, to first establish its viability, we implemented ThumbSpace as an input handler to custom applications written in C# (.NET Compact Framework 2.0) for Windows Mobile Pocket PCs using the PocketPiccolo.NET graphics toolkit [18]. The software ran on a 400 MHz

Cingular 8525 PocketPC phone (Figure 38b) with a 43.2 mm x 57.6 mm display area of 240 x 320 pixels. The device was chosen because it was the only device available that had all the hardware components we studied (a touchscreen, directional navigation pad, and scroll wheel), thus avoiding a potential confound.

Since touchscreens recognize only a single average point from a finger touch, that point is not only hard to predict, but also unstable, due to continual updates as the soft finger tip deforms on the flat screen surface. This problem is well known to make pixel-level targeting difficult, so various approaches have been used to stabilize finger input [131, 147]. In our work we take the approach used in Shift [147], transforming each input point by applying a recursive dynamic filter to stabilize pointer movement during slow corrective movements [147]. We found cutoff frequencies of 5 and 20 Hz interpolated between 54 and 144 mm/s worked well for both ThumbSpace and Shift. Given the small size of our targets, Shift was configured to escalate (display the callout) as soon as the finger touched the screen, but we did not correct for users' perceived contact points as in [147].

Several techniques for touchscreen object selection have been reported and studied in the literature. Two of the most common approaches are land-on and take-off, for which the first or last input points, respectively, are taken as the point of selection. We tested both techniques in pilots with mixed results. Since targets are completely hidden once the user's finger has landed, users cannot adjust the selection in an informed way. We reasoned that land-on reflected the user's best guess at the target, and chose that approach for the DirectTouch condition. For consistency with the other touchscreen-based input techniques, we filtered the input points using the same parameters as

ThumbSpace, and registered a change in selection only if the user moved her thumb at least 20 pixels away from the point of contact, in effect stabilizing the selection.

6.4.5 Method

The study was a 5 (*Input*: DPad, ScrollWheel, DirectTouch, Shift, ThumbSpace) x 2 (*Mobility*: standing, walking) x 12 (*Region*) x 4 (*Position*) repeated measures within-subjects factorial design. Presentation of *Input* and *Mobility* were counterbalanced across participants, and the 48 region x position trials were randomized within blocks.

Dependent variables collected included task time, error rate, satisfaction ratings, and interface preference rankings.

6.4.6 Participants

Twelve right handed volunteers (8 male, 4 female) ranging in age from 21 to 31 ($\mu = 26$) were recruited via fliers posted in the Department of Computer Science. We obtained institutional review board (IRB) approval to conduct this study, and all participants granted us their informed consent. Participants received $15 for 1.5 hours of their time.

6.4.7 Procedure

Before each block of trials, the study administrator explained and demonstrated the input method that would be used. Participants were instructed to select each target as quickly as possible without sacrificing accuracy. For walking conditions, participants were asked to walk at a comfortable pace along the figure-8 during all trials.

Participants then assumed a standing position or began walking, and began the practice trials. For DirectTouch, DPad and ScrollWheel, users performed 20 random practice trials the first time they saw the input method, and 10 the second time. Since

Shift and ThumbSpace were new to users, they performed 40 random practice trials the first time they encountered the input method, and 20 the second time. After the practice trials, users performed the 48 timed tasks. After each block, participants filled out a short subjective questionnaire about the *Input* x *Mobility* condition. Users proceeded in this manner for all 10 *Input* x *Mobility* blocks. A final block of trials offered users the option of using DirectTouch or ThumbSpace, to gather information about when, if ever, users would choose to launch ThumbSpace based on the position of the target.

Participants then completed a "usability" phase, which provided the opportunity to use all input methods with a realistic interface. For this phase users remained standing as the study software presented 10 tasks for each of the 5 Input conditions (DPad, ScrollWheel, DirectTouch, Shift, and ThumbSpace) in that order (roughly familiar to unfamiliar) for a Windows Mobile Calendar interface (Figure 37). Following the usability phase, users ranked the input methods from 1= "favorite" to 5= "least favorite" by target type (2: easy-to-reach and hard-to-reach), mobility condition (2: standing and walking), and expected overall preference once sufficient practice and expertise had been achieved. This last question included an option for using ThumbSpace when desired, and Shift otherwise.

6.5 Study 1 Results

6.5.1 Task Times

Task time was measured from the onset of the trial (when the scroll wheel or center of the directional pad was released, or the user's finger was lifted from the task dialog) to the completion of the trial (when the scroll wheel or center of the directional pad was pressed, or the user's finger was lifted from the screen). Task times for each region were

determined by averaging the region's four position trials. Trials with selection errors and outliers more than three standard deviations from the mean within each input type were excluded from the aggregation. Huynh-Feldt corrections are used where sphericity did not hold.

A 5 (*Input*: DPad v. ScrollWheel v. DirectTouch v. Shift v. ThumbSpace) x 2 (*Mobility*: standing v. walking) x 12 (*Region*: 1-12) repeated measures Analysis of Variance (RM-ANOVA) was carried out on the average task time data. Main effects of input $F(4, 24)=67.7$, $p< .001$, and region $F(11, 66)=68.5.4$, $p<.001$, were observed. In addition, an interaction of input x region $F(44,264)=50.0$, $p<.001$, was also observed.

On average, DirectTouch was significantly faster (865 ms) than all other interaction methods; ScrollWheel was significantly slower (3311 ms) than the others, while Shift, ThumbSpace, and DPad did not differ significantly from one another (Figure 40a). These results support H1, but not H6, since DPad was as fast as Shift and ThumbSpace.

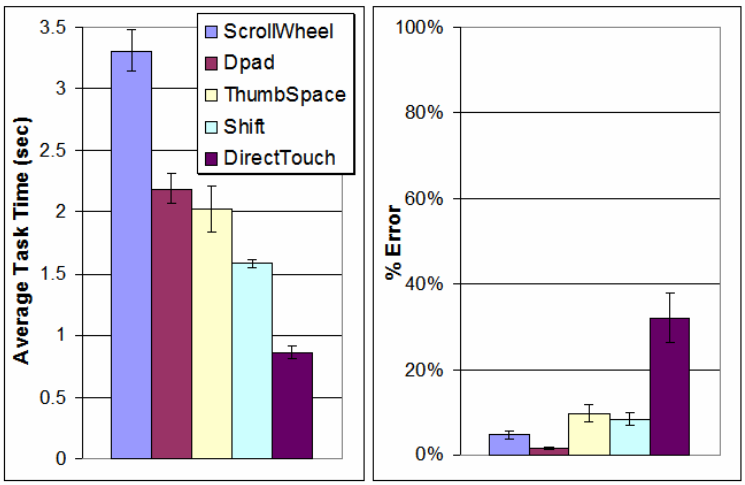


Figure 40. ThumbSpace Study 1: Average task times (a) and error rates (b) by input type.

Average task times increased consistently with region number, where region 1 was the fastest and region 12 was the slowest. While this seems at first unintuitive, the result can be understood by considering the input x region effect, whereby DPad and ScrollWheel task times generally increased by region, while DirectTouch, Shift and ThumbSpace times remained constant across regions (Figure 41a), resulting in an average overall increase in task time by region. The pattern observed between Shift and ThumbSpace in Figure 41a indicates that the two techniques were most similar in the "hard to reach" regions (1,2,3), but that Shift generally offered a speed advantage over ThumbSpace in regions (4-12), the majority of which were considered "easy to reach". While this trend supports H4, H3 failed to be supported since ThumbSpace was never faster than Shift.

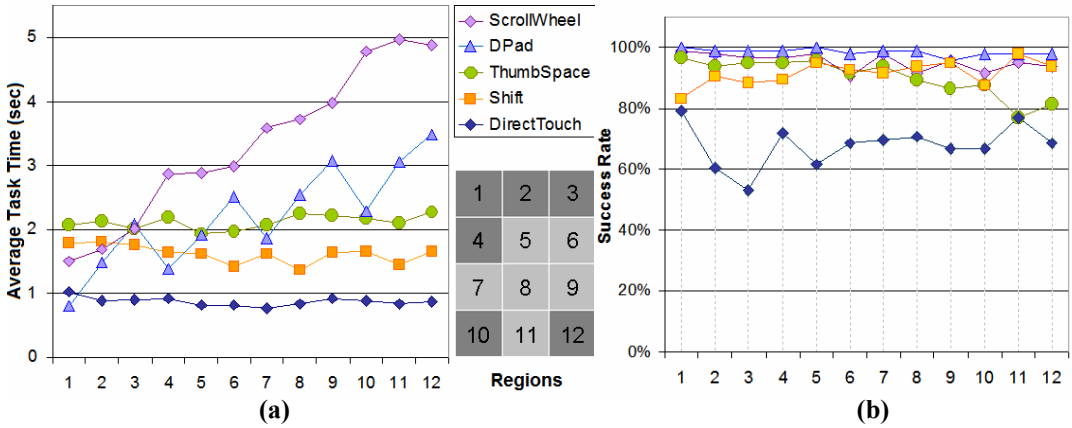


Figure 41. ThumbSpace Study 1: Average task times (a) and success rates (b) by region for each input type.

Note that task time patterns for DPad and ScrollWheel clearly illustrate the 2D and linear target access approaches of each. For DPad and ScrollWheel, region 1 is fastest because the cursor starts there. For DPad, the time taken to access targets in the other regions is linear in the number of 2D steps required, creating equivalence groups: (2,4), (3,5,7), (6,8,10) and (9,11). With ScrollWheel, accessing a target in a region

requires traveling through all lower numbered regions, hence the increasing time by region. It is important, therefore, to be clear that average task times for DPad and ScrollWheel will depend on the total number of interface objects and the relative access frequencies of each.

6.5.2 Error Rate

A 5 (*Input*) x 2 (*Mobility*) x 12 (*Region*) RM-ANOVA was performed the average percent error data. Main effects of input $F(1.4,14.9)=21.2$, $p<.001$, mobility $F(1,11)=5.2$, $p=.04$, and an interaction of input x region $F(44,484)=2.6$, $p<.001$, were found.

On average, users were slightly more accurate while standing than walking (10% v. 12% error). With 32% error, DirectTouch was significantly less accurate than any of the other input methods (Figure 40b), supporting H2. DPad (1% error), on the other hand, was significantly more accurate than ScrollWheel (5% error), Shift (8% error) and ThumbSpace (10% error), which as a group were statistically indistinguishable from one another. These error results fail to confirm H5, since we expected ScrollWheel to be as accurate as DPad. User comments indicated that ScrollWheel's error rate was likely due to accidental movement when the scroll wheel was pressed to perform selection. Different scroll wheels may vary in their susceptibility to this type of error.

Examining the input x region data, accuracies of DPad, ScrollWheel, Shift, and ThumbSpace were generally stable and comparable across regions (Figure 41b). A few notable exceptions were Shift's reduced performance in region 1, and ThumbSpace's lower accuracy in regions 11 and 12. User accuracy with DirectTouch was the most variable across regions, and was dramatically less accurate than the other methods for a great majority of regions.

6.5.3 Satisfaction

We performed a 5 (*Input*) x 2 (*Mobility*) x 13 (*Question*) RM-ANOVA on participant satisfaction ratings, selected from a 7 point scale, (1=low, 7=high satisfaction). A main effect of question $F(12,120)=7.9$, $p<.001$ and an interaction between input and question $F(48,480)=4.3$, $p<.001$ were found.

Examining the input x question data, DPad, Shift, and ThumbSpace scores were on average fairly high (e.g. majority ≥ 5), and were generally comparable across questions. Where they differed most, DPad was considered more accurate than Shift and ThumbSpace (6.5 vs. 5.4 and 5.1, respectively), and ThumbSpace was considered less preferred for "easy to reach" targets than were Shift and DPad (4.3 vs 5.5 and 5.1, respectively). DirectTouch ratings were similar to DPad, Shift, and ThumbSpace, except that it was considered less accurate, more frustrating, less satisfying, and less useful for "hard to reach" regions than were the other three. Finally, ScrollWheel stood out for being less fun, less satisfying, slower, and more physically demanding than the majority of other inputs.

6.5.4 Preference

Based on experience during the study, we asked participants to provide an absolute ranking of the 5 input methods, plus a sixth option of using a Shift+ThumbSpace combination, from 1=Best to 6=Worst for the majority of device interaction. The Shift+ThumbSpace was most popular on average (2.1), followed by Shift (2.6), ThumbSpace tied with DPad (3.3), then DirectTouch (3.8), and ScrollWheel (5.4). Although Shift received the most top rankings (4 users), it was closely followed by DPad and Shift+ThumbSpace (3 users). In fact, 75% of participants ranked Shift+ThumbSpace

as their first or second choice, as opposed to only 42% for Shift, 33% for DPad, and 25% for ThumbSpace.

6.5.5 Discussion

With respect to our broad questions about the relative value of peripheral hardware vs. touchscreen interaction for mobile, one-handed computing, we found little evidence that one approach should be recommended over the others. While DPad enjoyed surprisingly high satisfaction ratings, was relatively fast, and was the most accurate selection method, it ranked below two of the touchscreen methods with respect to absolute preference, and was the only method that generated unsolicited comments about hand fatigue. The fact that ScrollWheel was the least preferred input method is almost certainly related to the large number of targets presented in the study, which made average selection time slow, and except for the top row of targets, the 2D left-to-right, top-to-bottom cursor movement was never the most direct path to a target. However, we should only conclude that ScrollWheel input may not be suitable for dense 2D interfaces, and not that ScrollWheel is an ineffective input method in general. On the contrary, Blackberry devices are a great example of how thoughtful, complementary software designs can render scroll wheels both effective and enjoyable. Moreover, Blackberry's recent move to trackballs, which support 2D cursor movement, will likely eliminate this problem entirely.

Of the touchscreen methods studied, it is clear that using the thumb directly (e.g., DirectTouch) to hit small targets (e.g., ≤ 9mm) is unacceptably error-prone, at least when considering use with standard resistive touchscreen technology. While a corrective algorithm such as a nearest-target selection strategy could be used to improve direct

thumb interaction for sparsely populated displays, users will still require more precise selection capabilities when aiming at small objects in close proximity to one another. For such scenarios, the study data suggest that Shift and ThumbSpace are two software solutions that can greatly improve user accuracy in hitting small (3.6 mm) targets with the thumb. This finding has important implications for thumb use of touchscreen interfaces, since small targets that are common in today's designs (e.g., PocketPC) are generally easy to hit with two-handed stylus interaction, yet rather inaccessible to thumbs.

In terms of overall speed, accuracy, and satisfaction ratings, the Shift and ThumbSpace techniques were statistically indistinguishable from one another, although trends in the data suggest Shift holds some practical advantage over ThumbSpace. In particular, user speed, accuracy, and satisfaction ratings were equivalent or higher for Shift than for ThumbSpace in "easy to reach" regions. This result is not surprising since "easy to reach" regions are precisely those for which we assumed ThumbSpace would not be required. Instead, the primary concerns in "easy-to-reach" areas are the imprecision of selection due to the thumb's large contact area with the touchscreens, and the related problem of target occlusion—issues that Shift was specifically designed to address. Since ThumbSpace offered little noticeable advantage in the remaining "hard to reach" regions, it is understandable that users chose Shift over ThumbSpace as the method they would prefer for device interaction in the general case.

Despite the general favor that Shift enjoyed over ThumbSpace in this study, the data also offer evidence that the outcome might have differed if another device model had been chosen for study. For example, on average, participants ranked Shift *with*

ThumbSpace (Shift+ThumbSpace) as the most preferred usage option, even though they had not encountered this specific combination during the study. Since target size remained constant, a reasonable interpretation of this finding is that users perceived the two techniques to have different advantages based on the location of the desired target. This is supported by performance trends, which revealed that Shift and ThumbSpace had the most similar target access times in the "hard to reach" regions along the top of the device, which were also those for which ThumbSpace averaged higher accuracy scores. A likely explanation for this finding is that the top regions represent those that users found more difficult to reach. Although the device used in the study satisfied our requirements for input capabilities (scroll wheel, directional pad, and touchscreen), in fact its 2.8" screen is 20% smaller than many common devices that have 3.5" displays, including several models of the HP iPAQ PDA (whose screens are 1.0 cm wider and 1.5cm taller) and the iPhone (whose screen is 0.8 cm wider and 1.8 cm taller). Due to the relatively small size of the Cingular 8525, it is possible that the upper regions were only *relatively* "hard to reach", not *actually* "hard to reach". Thus the performance data for the top regions may represent only the borderline cases between those regions which ThumbSpace and Shift are comparable, and those for which ThumbSpace offers an advantage over Shift.

Finally, the fact that Shift+ThumbSpace ranked at the top of the preference ratings despite neither having an advantage in speed or accuracy over DPad bolsters our intuition that users prefer pointing at targets directly for selecting on-screen objects rather than using indirect hardware methods. A comment by one user, "I'm not sure I am actually faster using Shift (vs. DPad), but I *feel* that I am" offers some insight into why

this could be the case. But without software techniques like Shift and ThumbSpace, direct thumb interaction is unacceptably error-prone for small targets. Since most touchscreen devices already feature a directional navigation pad as a standard hardware feature, the addition of software techniques like Shift and ThumbSpace provides users even greater flexibility in adapting their interaction choices according to use scenario, attention demands, personal preference, and the inevitable speed/accuracy/stability tradeoffs among techniques.

6.6 Study 2: ThumbSpace vs. Shift for Palm-Sized Touchscreen Devices

In my exploration of the impact device size has on one-handed thumb movement (see my Thumb Movement Study in Chapter 3, Section 3.2) [75], I found that users generally have difficulty reaching all areas of the screen when using a large device, exemplified in that work by a model of the HP iPAQ 4150 (2.78" x 4.47" x 0.53"). Contrasting this with data from the study above, there is no strong evidence that users had problems reaching all areas of the Cingular 8525 screen. But not only is the screen of the iPAQ 20% larger, but its physical form is 1.2 cm wider than the Cingular 8525. This is important because touchscreen devices are not experiencing the same dramatic trend toward miniaturization as non-touchscreen devices, in part due to the current focus on media play (photos, album art, video and TV) for these devices, where the more pixels a device has, the bigger (or better) the image (and presumably user experience). Indeed, the strong sales of the iPhone—a touchscreen-based device which encapsulates a phone, music/video player, and web browser—prove that users are willing to forego small size for desirable features. To the extent that some proportion of this device class continues to have touchscreens, users will face issues of whether and how to operate the device in one hand, and based on

the results from the previous study, it remains an open question whether ThumbSpace can offer advantages over Shift for devices at the larger end of the device size spectrum.

In addition to developing a better understanding of whether reach is indeed a problem in using "large" touchscreen devices with one hand, and if so, whether ThumbSpace is effective in alleviating the associated penalties, there are two other reasons to motivate further investigation of ThumbSpace. The second reason to study ThumbSpace is that, on average, users from the previous study stated they would prefer to use Shift *with* ThumbSpace (Shift+ThumbSpace) over using either of the techniques exclusively. Yet because the study participants were never exposed to this combination formally, it is unknown whether that opinion would hold up in practice. To be sure, there are reasons to expect this would *not* be the case. Giving users a choice between ThumbSpace and Shift adds a mental calculation to every action, which not only increases the attention demands of the task, but also increases the total selection time [31]. Due to the penalties of making a decision, users may default to making no decision at all, and instead use the technique that is less demanding on average for all targets. But then which technique are users likely to favor? Even assuming that the adjustment phases for Shift (e.g., the crosshair cursor) and ThumbSpace (e.g., the object cursor) consume equal attention and time resources, ThumbSpace incurs an additional time penalty when it is triggered. On the other hand, if reach is a significant issue for one-handed use of common touchscreen devices, the comfort and stability offered by a personalized ThumbSpace may be a valued tradeoff for a slightly longer selection time. Since we do not expect ThumbSpace to be chosen all the time, understanding whether users are

willing to trigger ThumbSpace at all is as important to its viability as its relative performance to Shift.

Finally, the third reason for comparing ThumbSpace and Shift on another device is that after years of using the HP iPAQ as a study platform, it was immediately evident that the Cingular 8525 was less responsive to finger input than the HP device. From a layperson's perspective, the difference between the displays might be described as though the Cingular 8525 had a "stiffer" screen. Although imperceptible when using the focused point of a stylus, the Cingular 8525 required more finger pressure to detect and maintain thumb contact than did the HP iPAQ. Hardware nuances across devices are of course to be expected, and there is no reason to assume the differences between the HP iPAQ and the Cingular 8525 would have impacted Shift and ThumbSpace unequally. Yet because both techniques require users to fine-tune selections by rolling or dragging the thumb, it is reasonable to assume that screen sensitivity will play a role in user experience. The advents of the iPhone and LG Prada phones, and the anticipation of several other capacitive touchscreen devices (e.g., Nokia Aeon, Synaptics Onyx) promise that finger-sensitive touchscreens will be commonplace in the future. The proliferation of devices with capacitive touchscreens is certain to change attitudes about finger use of touchscreens, as the user experience with this technology is qualitatively different; not only is targeting precision higher on capacitive screens, but dragging requires only a very light touch, both of which make interaction significantly easier in our estimation. In light of the trend toward more sensitive screens, it makes sense to revisit Shift and ThumbSpace on more sensitive equipment that was used in the first study. However,

since resistive touchscreens are vastly more common than capacitive screens at this time, we will focus on resistive technology first.

To answer questions left open from the previous study, I conducted a final study with the goals of 1) evaluating Shift and ThumbSpace on a "large" touchscreen device with increased (but standard) sensitivity; and 2) understanding usage patterns when participants are given their choice of input methods.

6.6.1 Independent Variables

Four input methods (*Input*) were studied: *Shift*, *ThumbSpace*, Shift with ThumbSpace (referred to as *Combined*), and the baseline of using the thumb to hit targets directly without software enhancement (*DirectTouch*). For consistency with the previous study, only one target size (3.6 mm) was considered, and targets were again assigned to one of 12 device regions (*Region*). Finally, to take into account the possibility of learning effects within an *Input*, tasks were repeated across two blocks (*Block*).

6.6.2 Implementation and Apparatus

The study was performed on an HP iPAQ 4155. The codes for the Shift and ThumbSpace techniques were the same as those used in Study 1. Because the pixel sizes differ between the 8525 (0.18 mm/px) and the iPAQ (0.24 mm/px), pixel values were updated for the Shift camera offset, the Shift camera diameter, and the target sizes to maintain the absolute measurements (in mm) from the previous study. Otherwise, the only software modifications between Study 1 and Study 2 were to the study control software to reflect changes in the study design, described below.

(a) (b) (c)

Figure 42. ThumbSpace Study 2: Representative screen shots from the experiment. An example of the task prompt (a). Performing a task using Shift (b). Performing a task using ThumbSpace (c).

6.6.3 Tasks

For comparability across studies, the selection tasks were those from Study 1. Targets were 15 pixel (3.6 mm^2) squares, arranged in a 6 x 8 grid, and grouped into 12 4x4 *Regions* for analysis. Each trial began with a message identifying the target to hit (Figure 42a). Because of concerns in Study 1 that the yellow target was too similar to the yellow roads of the map, the trial target was shown in red. Breaking from the approach in Study 1, users started the trial timer by pressing the center of the directional navigation pad, regardless of input method. For Shift, users then aimed for the desired target; for ThumbSpace, users pressed the center button a second time to launch ThumbSpace, and then aimed for the target within ThumbSpace. For the *Combined* input condition, users started trials in the same manner as they did the other *Input* conditions, but chose on-the-fly whether to then aim directly at the target using Shift or whether to press the center of the directional pad a second time to launch ThumbSpace.

One argument for using a single start method in Study 2 is to standardize users' grips across input methods. Because the directional input hardware is generally useful in one-handed operation, it seems reasonable that a "home" grip would accommodate thumb reach to that location. The drawback, of course, is that Shift then incurs the time penalty associated with moving the thumb from the center button to the target, just as ThumbSpace does. While this may seem unfair at first, it is important to consider that it is only the *act* of triggering ThumbSpace that is unavoidable—the manner in which it is accomplished is not an inherent property of ThumbSpace. One can imagine several trigger designs that require shorter travel distances for the thumb, such as a conveniently positioned bezel-mounted button, a gesture, a squeeze, using another finger, setting a sticky-mode, etc. Because each of these options would have a subtly different impact on ThumbSpace usability, it might be more accurate to say that comparing ThumbSpace to another technique without fully exploring the optimal trigger method unfairly biases the results against ThumbSpace. Since my goal at this phase was to understand the viability of ThumbSpace as a proxy for interacting with a surface area that is larger than a user's reach, and not to design of the most effective trigger mechanism, I chose to abstract away the variable cost of triggering ThumbSpace by equalizing the movement burden across the input techniques. Results of the study should therefore be interpreted as having used the most optimistic trigger penalty of "a button press".

6.6.4 Method

The study was a 4 (*Input:* DirectTouch, Shift, ThumbSpace, Combined) x 12 (*Region*) x 4 (*Position*) x 2 (*Block*) repeated measures within-subjects factorial design. Presentation of three inputs (DirectTouch, Shift, ThumbSpace) were counterbalanced across

participants, but the Combined input was always presented last to ensure participants had encountered both Shift and ThumbSpace prior to their combined use. For each *Input*, users performed 24 practice trials, 2 per regions, followed by two blocks of randomized *Region* x *Position* test trials. Prior to the Combined trials, users performed a "usability" phase in which they were given 2 minutes of self-directed exploration using Combined input for both a Calendar (Figure 37) and Start Menu interface.

Dependent variables collected included task time, error rate, satisfaction ratings, and interface preference rankings. For the Combined input condition, the user input choice for each trial was also recorded.

6.6.5 Participants

Twelve right handed participants (5 male, 7 female), ages 18-29 ($\mu = 21$) were recruited from the general campus population at the University of Maryland, College Park. One of the participants studied information management, while the remainder had humanities or social sciences backgrounds. Because of the potential for hand size to bias users toward one input technique or another (e.g., users with small hands might benefit most from ThumbSpace), we aimed for an even distribution of hand size in our participant population. Participants' hands were classified into three broad categories S (n=3), M (n=5), and L (n=4), based on comparison to my own hand, which fits an M/L women's glove. Participants with finger lengths that measured within ±1 cm of a paper-based outline of my own hand were classified as M; and those falling below and above that range were classified as S and L respectively. We obtained institutional review board (IRB) approval to conduct this study, and all participants granted us their informed consent.

6.6.6 Procedure

The procedure for Study 2 was modeled after that of Study 1. The session began with participants reading an overview of the study. Prior to performing the practice trials for an *Input*, users read a brief introduction to the technique, which also reminded them to strive for speed and accuracy. Participants were allowed to rest between blocks, but none took the opportunity to do so. For the Combined input condition, participants were urged to use whichever technique (Shift or ThumbSpace) they felt offered them the highest speed and accuracy for each trial. The administrator explicitly stated that participants were free to use a single technique for all trials, or mix and match the techniques across trials. After completing all trials for an *Input* condition, participants filled out a satisfaction questionnaire for the technique. After all trials had been completed, users recorded which regions of the device they found "easy to reach" and "hard to reach", and ranked the four *Inputs* from most preferred (4) to least preferred (1). Total session time lasted between 45 and 75 minutes for which participants were paid $15.

6.7 Study 2 Results

6.7.1 Task Times

Task time was measured from the onset of the trial (when the center of the directional pad was released) to the completion of the trial (when a user lifted her finger from the screen). Task times for each region were determined by averaging the region's four position trials. Trials with selection errors and outliers more than three standard deviations from the mean within each input type were excluded from the aggregation.

Huynh-Feldt corrections are reported where sphericity did not hold and post hoc analyses of main effects were conducted using Bonferroni correction.

A 4 (*Input:* DirectTouch, Shift, ThumbSpace, Combined) x 12 (*Region*) x 2 (*Block*) repeated measures Analysis of Variance (RM-ANOVA) was carried out on the average task time data. Since there was no main effect of *Block*, it was removed from further analysis. Main effects of *Input* $F(3, 33)=38.0$, $p<.001$, and *Region* $F(6, 65.9)=4.7$, $p<.001$, were observed. In addition, an interaction between *Input* x *Region* $F(33, 363)=1.8$, $p=.006$, was also found.

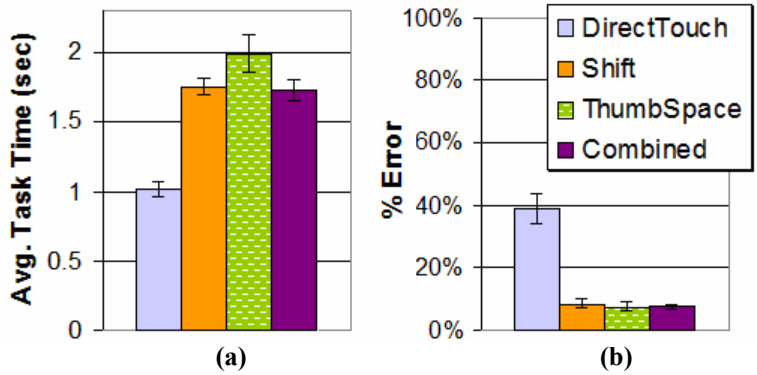


Figure 43. ThumbSpace Study 2: Average task times (a) and error rates (b) by input type.

DirectTouch was significantly faster (1017 ms) on average than the other input methods. Although ThumbSpace was the slowest input method on average (1993 ms), it was not found to differ significantly from Shift (1753 ms) or Combined (1724 ms), shown in Figure 43a. As can be seen from Figure 44a, average task times were highest in the top regions (1,2,3,4) and lowest in the central regions (5,7,8,9,11). The 3x4 grid that accompanies the graph in Figure 44 depicts the screen locations associated with each region, and conveys the majority user opinion about which areas were "hard to reach" (dark gray) and "easy to reach" (light gray); region 10 did not have majority support for either classification. The data trends in Figure 44a correspond well to user opinion; the

slowest regions were all considered “hard to reach”, while the fastest regions were all considered “easy to reach”. However, post hoc comparisons only revealed region 1 to be significantly slower than any others (5,6,7,8,9).

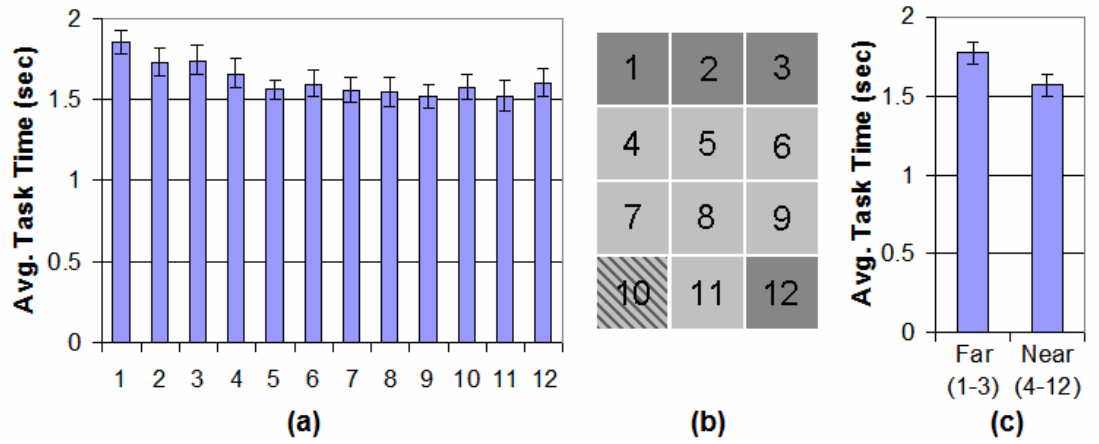


Figure 44. ThumbSpace Study 2: Average task times by input region. Average task times by region (a). User-defined “hard to reach” (dark) and “easy to reach” (light) regions (b). Average task time by distances “far” and “near” (c). Regions (1-3) were considered “far” because the majority of users thought those regions were hard to reach, and distance is the most reasonable explanation for why they thought this. All other regions, by contrast, were considered “near”.

Examining the *Input* x *Region* task time data (Figure 45a), we can make out a few trends. First, Shift generally took less time the closer a target was to the bottom of the screen. This finding is consistent with the fact that users would have had to move their thumbs shorter distances for lower targets. DirectTouch, which required the same amount of thumb travel as Shift, displays a similar trend. Interestingly, users were slower using Shift in regions along the left-hand side of the device (1,4,7,10) compared to regions within the same row—possible evidence that users were experiencing problems with reach in those areas. ThumbSpace, on the other hand, supported fairly consistent access times across Regions (4-11). The relatively long access time for ThumbSpace in Region 12 reflects the fact that when aiming at the lower right corner of a ThumbSpace, the purpose is to hit a target in the lower right corner of the screen. However, one the user hits the screen, her thumb will be blocking the lower right corner, and so must then make

special efforts to visually confirm the selection. The Combined input method, which ideally reflects each user's personal, optimized choice between Shift and ThumbSpace for each target, supported more stable access times across regions than did the use of either Shift or ThumbSpace alone. Furthermore, the access times of Combined input were faster than or statistically indistinguishable from both Shift and ThumbSpace within each region.

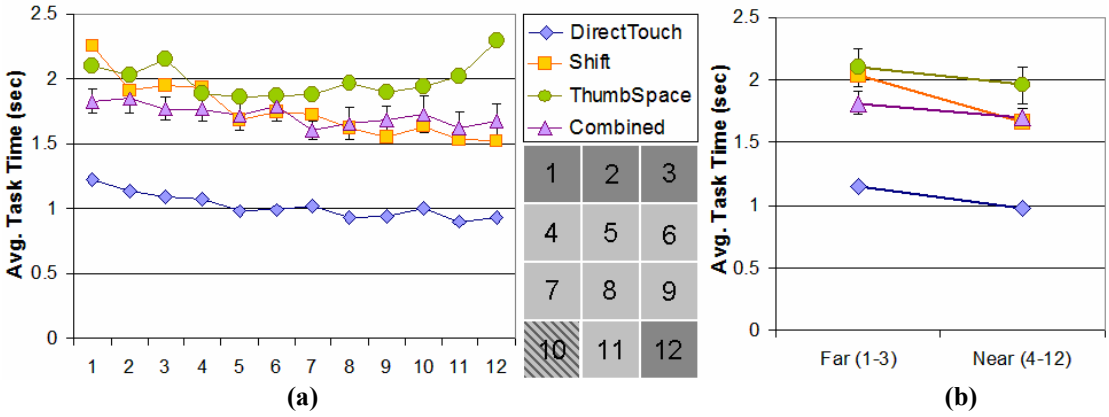


Figure 45. ThumbSpace Study 2: Average task times by region for each input type. Average task times for all 12 regions (a). Average task times for the aggregate distances "far" and "near" (b).

Since the goal of ThumbSpace is to help users hit targets that are out of thumb reach during one-handed touchscreen interaction, we also wanted to analyze the data with respect to "near" and "far" targets. Referring to the region grid of Figure 45, we see a clear delineation between the top row of regions and the remainder. It is highly likely that regions (1-3) were considered "hard to reach" were because they were "far", and that the "easy to reach" regions were so classified because they were "near". Furthermore, based on the "easy to reach" classification of surrounding regions as well as results from my Thumb Movement Study in Chapter 3, Section 3.2 [75], region 12 is "hard to reach" not because it is distant, but because it is actually *too* near, which can make thumb access awkward. It is possible that region 10 was also considered awkward to access, but based

on the property of distance alone, we are comfortable classifying region 10 as "near" since regions 4 and 7 were rated as "easy to reach" and are certainly no "nearer" than region 10. Thus regions 10 and 12 were assigned "near" status.

To make sense of some of the trends seen in Figure 45a, we ran a 4 (*Input:* DirectTouch, Shift, ThumbSpace, Combined) x 2 (*Distance:* Near, Far) RM-ANOVA on the aggregated region data. In addition to the main effect of *Input* reported above, the analysis revealed a main effect of *Distance* $F(1, 11)=12.5$, $p=.005$ and an interaction of *Input* x *Distance* $F(3,33)=3.4$, $p=.03$. Post hoc comparison of task times by *Distance* confirmed that users took significantly longer performing tasks in Far regions (1774 ms) than they did in Near regions (1570 ms), shown in Figure 44c. Considering the *Input* x *Distance* interaction data, Figure 45b suggests that Shift was more sensitive to differences in task distance than the other three input methods. Planned comparisons of the *Input* x *Distance* data using Shift as the reference input revealed that Shift/Far was slower than Combined/Far (2037 ms vs. 1816 ms, $p=.018$) but that Shift/Near was faster than ThumbSpace/Near (1659 ms vs. 1958 ms, $p=.013$).

Given that four contrasts were performed, we would have preferred $\alpha=.0125$ to protect against Type 1 Errors, and so will consider the *Input* x *Distance* contrast results as only borderline significant. The fact that Shift was faster than ThumbSpace in Near regions may be in part due to the occlusion problems that can occur when using ThumbSpace for targets that overlap the user-defined ThumbSpace. Another explanation may be that users are less accurate in their initial aim using ThumbSpace than using Shift, and so must perform more thumb dragging to position the object cursor over the desired target. However, the fact Shift/Near did not differ significantly from Combined/Near

indicates users made appropriate decisions when choosing between Shift and ThumbSpace for Near targets. Although Shift did not differ significantly from ThumbSpace in Far regions, users were significantly faster using Combined input in Far regions. This result is somewhat surprising since users were restricted to using Shift or ThumbSpace to hit far targets in the Combined condition. One explanation is that users may have become more efficient with the techniques over the course of the study, and that this learning effect benefited Combined disproportionately because it was always presented last. This could be true even though our block design did not reveal learning effects in time or error rate *within* input types.

6.7.2 Error Rate

A 4 (*Input*) x 2 (*Region*) RM-ANOVA was performed for average input error data. A main effect of *Input* $F(1.4, 15.2)=40.5$, $p<.001$ and an interaction between *Input* x *Region* $F(33,363)=2.3$, $p<.001$ were observed.

Despite the speed advantage offered by DirectTouch, users were significantly less accurate using DirectTouch (39% error) than the other three input methods, which were all equally accurate (7.4-8.4% error). Users were generally at least as accurate in regions (1-10) using ThumbSpace as Shift, but Shift supported more accurate selection in the lower left-hand regions (11, 12). Once again, Combined input was at least as successful as the other input methods across the regions, except in region 12 where Shift dominated.

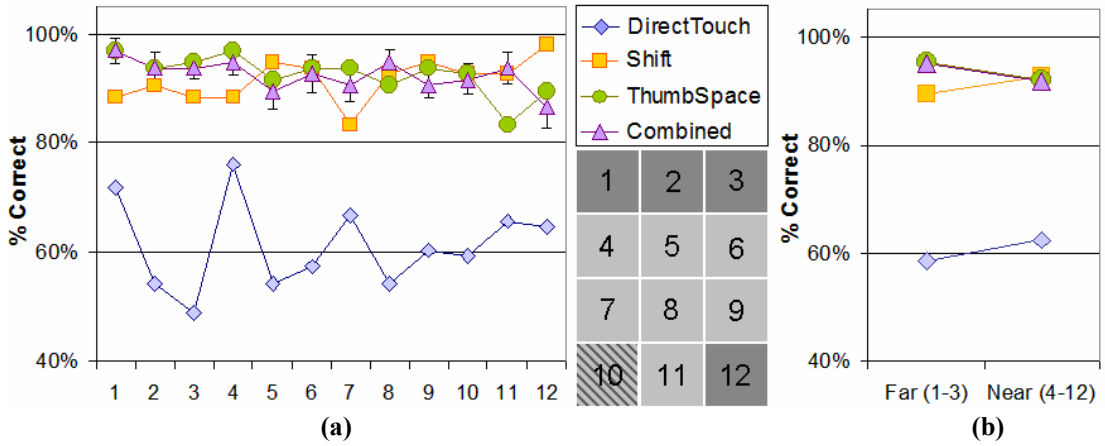


Figure 46. ThumbSpace Study 2: Success rates by region for each input type. Success rates for all 12 regions (a). Success rates for the aggregate distances "far" and "near" (b).

We again ran a 4 (*Input:* DirectTouch, Shift, ThumbSpace, Combined) x 2 (*Distance:* Near, Far) RM-ANOVA on the aggregated region data to better understand error trends by "near" and "far" regions. Other than the effect of *Input* reported above, only an interaction of *Input* x *Distance* $F(3,33)=4.5$, $p=.009$ was found.

Considering the *Input* x *Distance* interaction data in Figure 46b, users tended to be more accurate in Near regions than Far regions using Shift and DirectTouch, but users tended to be more accurate in Far regions than Near regions using ThumbSpace and Combined. Planned comparisons of the *Input* x *Distance* data using Shift as the reference input revealed a borderline significant ($\alpha=.0125$) result that Shift was less accurate than ThumbSpace for Far targets ($p=.013$), but was indistinguishable from Combined ($p=.027$). For Near targets, Shift was as accurate as both ThumbSpace and Combined. Thus for both distances, Combined supported equivalent or better performance results as Shift and ThumbSpace individually.

6.7.3 Input Choice

While it is certainly possible that each participant chose only a single input method to use for all regions in the Combined condition, the data tell a different story. Looking at the frequencies with which participants chose Shift vs. ThumbSpace by *Region* for Combined *Input* (Figure 47), we see that ThumbSpace was chosen more often than Shift for targets in the top half of the device (1-6), and that Shift was used increasingly toward the bottom of the device, especially in regions (11,12).

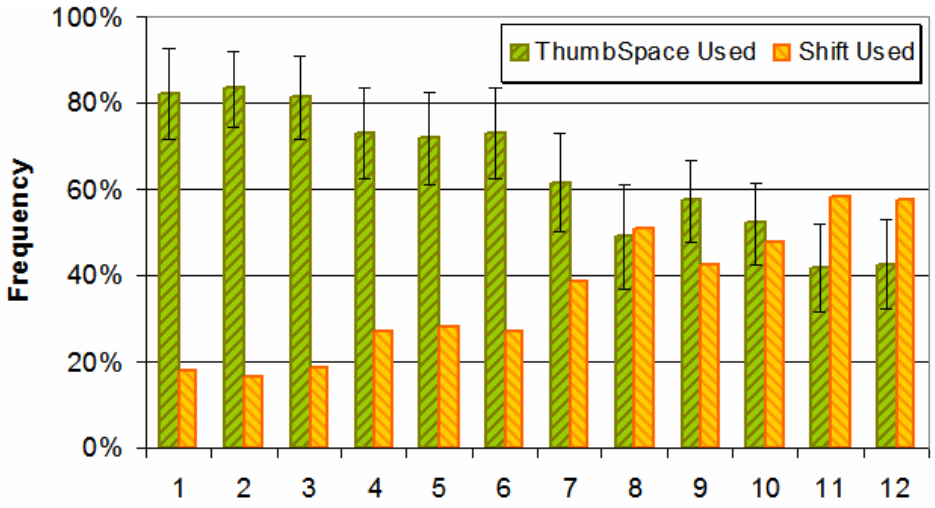


Figure 47. ThumbSpace Study 2: The relative frequencies of ThumbSpace vs. Shift use for the Combined input condition, for each of the 12 input regions.

The fact that participants primarily used ThumbSpace for Combined input in the top half of the device (1-6), explains why the error rates of ThumbSpace and Combined matched one another closely in these regions (Figure 46a). The relatively fast access times Combined enjoyed in these regions is especially interesting considering users chose ThumbSpace a majority of the time. It seems clear that learning was involved in users improving their access times from ThumbSpace to Combined. Generally, Combined allowed users to perform tasks at least as fast and accurately as did the other two methods, except in region 12, where Figure 47 reveals participants used ThumbSpace

nearly as often as Shift. Since we suspect ThumbSpace will always be a poor choice for use in region 12, it seems users would simply need more time to develop the reflex to use Shift over ThumbSpace in region 12.

The usage patterns of Shift and ThumbSpace for the Combined condition intriguing for three reasons. First, Figure 47 demonstrates that users modified their input strategies based on the device region. Second, the performance data of Figure 45 and Figure 46 indicate that the choices users made were "good" with respect to both time and errors. Finally, the benefit of allowing users to make a real-time choice for *Input* type according to the *Region* of access outweighed any time cost associated with making the decision.

6.7.4 Satisfaction

A 4 (*Input*) x 13 (*Question*) RM-ANOVA was performed on participant satisfaction ratings, selected from a 7 point scale, (1=low, 7=high satisfaction). Main effects of *Input* $F(3, 33)=11.9$, $p<.001$ and *Question* $F(12,132)=3.9$, $p<.0001$ were found, as well as an interaction between *Input* and *Question* $F(36,396)=3.9$, $p<.001$.

While it is unsurprising that there would be a difference in average scores among the satisfaction measures, in fact all measures were rated relatively high on average (≥ 5.1), especially learnability of the techniques (6.5), which rated significantly higher on average than nearly all the other measures. DirectTouch received significantly lower satisfaction scores on average than ThumbSpace and Combined (4.6 vs. 5.9 and 6.2), while at 5.5, Shift was not considered statistically more or less satisfying than the others. Overall, Shift, ThumbSpace and Combined were rated very similarly to one another

(Figure 48), except for the measures of Comfort and Stability, where Shift rated appreciably lower then the other two and quite similarly to DirectTouch.

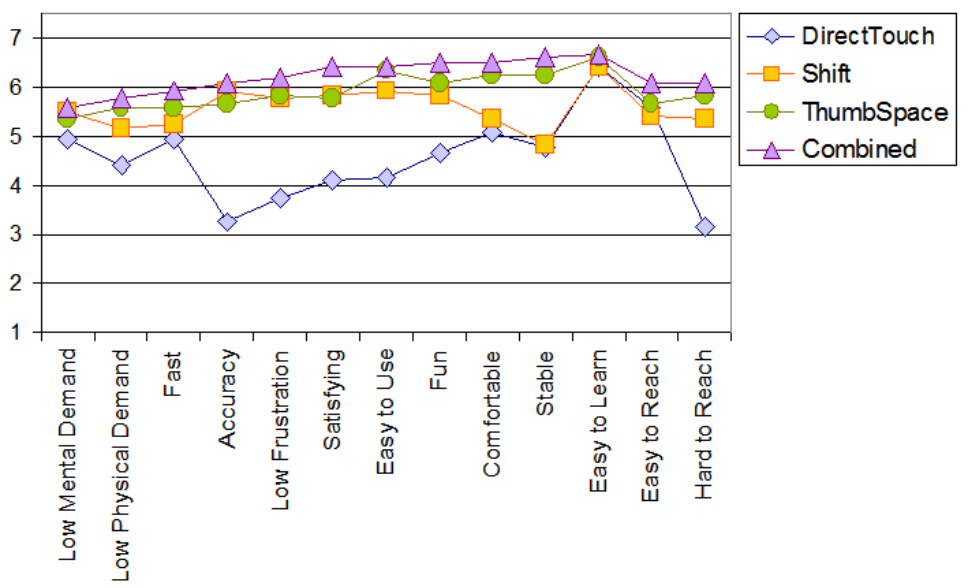


Figure 48. ThumbSpace Study 2: Average satisfaction ratings for each input type.

6.7.5 Preference

Based on experience during the study, we asked participants to provide an absolute ranking of the four input methods from 1=Worst to 4=Best. An single factor (*Input*) RM-ANOVA on the ranking results revealed a significant effect *Input* $F(3,33)=19.9$, $p<.001$. Post hoc tests revealed that DirectTouch ($\mu=1.3$) was significantly less preferred than the ThumbSpace ($\mu=3$) and Combined ($\mu=3.6$) but not Shift ($\mu=2$), and that Shift was significantly less preferred than Combined. These numbers more specifically show that nearly every participant ranked the methods from least to most preferred in the order Direct, Shift, ThumbSpace, Combined.

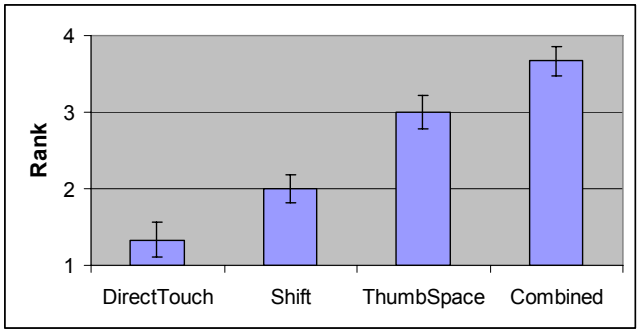


Figure 49. ThumbSpace Study 2: Ranking of the input types, from least preferred (1) to most preferred (4).

6.8 Discussion

ThumbSpace was conceived to address the basic issue that most touchscreen software designs require two-handed stylus interaction because targets can be too small to hit reliably with a finger (see my Target Size Study in Chapter 3, Section 3.3) [112], or too far to hit comfortably with the thumb while holding the device in one hand (see my Thumb Movement Study in Chapter 33.2) [75]. The reality of mobile device use, however, is that use scenarios vary widely and hand availability may be in continual flux. The richness of most touchscreen interface designs places high priority on information presentation, but at the cost of interaction inflexibility; users themselves then bear the burden of freeing both hands to operate the device [79], delaying device use until their physical resources are available, or instead assuming the consequences of reduced interaction stability, comfort, and/or accuracy.

One option is to redesign the software interfaces for touchscreens so that they accommodate one-handed thumb interaction, such as by including only large, centrally located targets. However, such solutions typically constrain the amount of information that can be displayed at a time, and so may serve only to shift the problem from selection (in)accuracy to navigation (in)efficiency. Another solution is to provide peripheral

navigation hardware, but as Study 1 showed, the challenge then lies in supporting timely average access rates, especially for 2 dimensional layouts.

ThumbSpace represents a third option, which aims to increase the types of use scenarios that traditional touchscreen designs can accommodate, without limiting the presentation design. Instead of reprioritizing the design from two-handed, high-attention interaction to one-handed, high-navigation interaction, ThumbSpace offers users the ability to operate the same interface under different mobility conditions and constraints. ThumbSpace also benefits designers, who can focus on general requirements for effective presentation and efficient navigation without the need to consider additional limitations of object sizes and placements.

Through the iterative design and evaluation of ThumbSpace we have experienced first hand the power of design nuance; while the theory of the ThumbSpace approach has changed little since its prototype, we have witnessed the strong impact that subtle modifications have had on ThumbSpace's usability and user acceptance. The results of the pilot study and Study 1 confirmed that users require ThumbSpace to be a true miniature representation of the display to help speed understanding of the mapping between ThumbSpace objects and display objects, as well as to improve target aim. In switching from a smaller to a larger device between Study 1 and Study 2, we observed that device size affects both the utility and perception of ThumbSpace. With smaller devices, such as those with 2.8 inch screens, the main problem users face is not that of thumb reach, but hitting small targets (3.6 mm) accurately with a finger, for which the Shift technique was on average very well suited. Even so, trends in the data together with user feedback suggested that a combined approach of using Shift for near targets and

ThumbSpace for far targets could offer better one-handed support than limiting users to just one technique. By moving to a larger device with a 3.5" screen in Study 2 and explicitly testing the use of ThumbSpace and Shift both individually and together, we confirmed overall advantages in speed, error and preference for the combined Shift+ThumbSpace approach.

Looking at the results of Study 2 in more depth, users were fastest hitting targets unaided with their thumbs, but the high average error rates (around 40%) made it the least preferred interaction approach. Shift and ThumbSpace, on the other hand, traded virtues according the position of the targets. For "near" targets, Shift and ThumbSpace had comparable error rates, but Shift supported faster selection times. For "far" targets, Shift and ThumbSpace had comparable access times, but ThumbSpace offered users higher accuracy. When users were allowed to choose Shift or ThumbSpace on a target-by-target basis, users were at least as fast and as accurate in "near" and "far" regions as using each technique exclusively. Examining the usage distributions of Shift and ThumbSpace by region for the combined input approach, it seems that users had developed an intuition for the relative benefits of each, favoring Shift for its speed in near regions and ThumbSpace for its accuracy in "far" regions. Although choosing between Shift and ThumbSpace added a mental step to each selection, it had no noticeable impact on selection speed, and in fact was overwhelmingly preferred by users (75%) than were either ThumbSpace (25%) or Shift (0%) alone.

To understand the practical implications of Study 2, it is necessary to examine the parameters under which the results were obtained. First, we studied thumb access to small (3.6 mm) touchscreen targets only, since they are common to traditional interfaces

and present significant challenges for finger interaction. For interfaces with larger targets, Shift automatically delays the onset of its callout proportionally with targets size so that users do not experience a visual disruption if objects are large enough to hit easily with a finger (e.g., ≥ 10 mm). But because the time cost of triggering ThumbSpace is constant, users may become less willing to use ThumbSpace if their success rate increases when "stabbing for" large but distant targets directly—even if the action requires a grip adjustment or momentary device instability. It is also important to reiterate that the effective physical cost of triggering ThumbSpace in Study 2 was a button press, and so the results reflect an idealized design that may be difficult to replicate in practice.

Even with these considerations, the Study 2 results make a compelling case for touchscreen devices to include Shift as a default interaction technique to support finger use in general, together with ThumbSpace for accommodating one-handed thumb operation. The finding that choice itself does not degrade performance suggests there is little downside to offering ThumbSpace as an interaction option, and its personalization in terms of size, position, and use occasion, mean it can flexibly and unobtrusively accommodate variations in hand size, interface design, and device dimension. That both Shift and ThumbSpace were considered Fast, Accurate, Easy to Learn, Easy to Use make them a very promising combination for generalized one-handed touchscreen operation. To be clear, Shift and ThumbSpace would be offered at the operating system level across all applications, and so would not be the concern of interface designers, who would be free to focus on information presentation and navigation independent of the vast and variable usage constraints of mobile computing.

Chapter 7
Conclusion

7.1 Contributions in Context: Implications for Next-Generation Design

In Chapter 1, pp 11-14, I outlined the contributions of this dissertation with respect to the specific results I obtained while investigating both *foundational* human factors in mobile computing with one hand, and *applications* of this knowledge for supporting real-world data access tasks with one hand. Here I revisit my experimental findings to offer insight into their broader implications for next-generation mobile design.

7.1.1 Foundations: Guidelines for One-Handed Mobile Device Interaction

Contribution (C1): Motivated the need for single-handed interface design based on the reporting of current usage patterns and preferences for one- and two-handed mobile device interaction.

When I began my research in 2003, the term "smart phone" was only just gaining prevalence in the vernacular, at least in part due to Microsoft's early 2002 announcement that their mobile operating system would be called Windows Mobile Powered SmartPhone 2002 [97]. Even so, smart phones accounted for only a very small percentage of the mobile market, where there was still a very clear division between cell phones (for making phone calls) and PDAs (for personal information management tasks). Furthermore, touchscreens were featured primarily on PDAs, which had a rather elite, professional user base.

Given the climate of the time, it's not surprising that some eyebrows were raised over the utility of focusing my research on one-handed use of touchscreen-based devices, and it was this skepticism that drove my initial exploratory studies. Yet results from my field study (Chapter 2, Section 2.2.3) and Web survey (Chapter 2, Section 2.3.3) confirmed that one-handed mobile device use was already widespread, and that a majority of mobile device owners, including owners of both keypad-based and touchscreen-based devices, desired better one-handed control of their devices. These early studies provided strong evidence that research in one-handed mobile device design would have important applications to the population in general, and that the benefits would not be limited to a niche user segment.

Today, mobile phones are integral to the daily lives of about a third of the world's population. And with increasing numbers of web services targeting mobile platforms, we can predict that users will become even more attached to their mobile devices, using them in a wide range of environments for an ever widening variety of tasks. With new platforms such as the Apple iPhone and the LG Prada phone pushing touchscreen-based phones into the mainstream, we can expect more people than ever will be making the switch from traditional keypad phones to touchscreen phones; further evidence of this possibility is the prediction by industry analysts at IMS research (www.imsresearch.com) that touchscreens could be featured in up to 30% of the devices on the market by 2011. However, it is unlikely that users will give up the habit and convenience of one-handed device operation in the transition. These trends suggest that the one-handed touchscreen operation is becoming an increasingly relevant and important method of device interaction.

Unfortunately, the software interfaces of most touchscreen-based mobile devices available today are designed primarily for stylus, and thus two-handed, interaction. The iPhone is one of the few examples of a touchscreen-based phone that supports single-handed operation reasonably well; however, its grid-like layout and finger-sized targets represent only one of many possible design approaches for supporting thumb operation of a touchscreen, and it is a choice that necessarily sacrifices some flexibility in information presentation. Thus, broadly speaking, the problem domain of one-handed touchscreen interaction remains relatively unexplored, and so suggests ample opportunities exist for innovation in improving users' one-handed touchscreen experiences.

Bottom Line. Devices should support one-handed use by some means for all device features. Many common mobile scenarios require users to devote one of their hands to carrying objects or interacting with the environment, which then leaves users only one hand for operating a mobile device. Furthermore, the incidence of such circumstances is often independent of the user's task at hand, so it would be impractical to assume we can identify a strict subset of tasks that should be adapted to one-handed operation. Furthermore, the usability cost for any task being inoperable with one hand is enormous—not only is it the difference between the task being possible vs. impossible, but it disrupts task flow and frustrates the user. While one approach to avoiding such negative use experiences would be to optimize all device features for one-handed use, users will also benefit from a more generalized interaction strategy, even one that is relatively inefficient by comparison to two-handed operation, assuming it supports a task that would otherwise be impossible to perform with one hand.

Contribution (C2): Provided researchers and practitioners with practical, empirically-backed guidelines for one-handed mobile interface designs.

(C2.1) Support localized interaction to minimize grip adjustment.

This design recommendation is backed by the target location results from my thumb movement study, reported in Chapter 3, Section 3.2.7, and summarized with observational findings in paragraph 4 of Section 3.2.8. In general, the data suggest that users are able to control devices of all sizes reasonably well as long as repeated interactions occur within close proximity to one another.

Although this guideline allows for considerable flexibility in interface design, I should clarify that it is more important for the rule be applied within the set of operations that define a logical task, than across the entire user experience with a device. That is, while it might be ideal if a user could access every feature a device has to offer without needing to adjust her hand grip, there are several pragmatic hurdles to offering users such a system, especially if it has a touchscreen. First, for this goal to be achievable for a touchscreen, a portion of the screen would need to be designated as the "interaction region" across all applications. But since one of the advantages of touchscreen systems is that their software interfaces are more flexible in terms of interaction object design and layout than non-touchscreen systems, it is unlikely that the designation of a single interaction region would serve all applications and tasks equally well. Second, by limiting all interaction to a particular region of the screen, that area may become overcrowded; this may lead to smaller targets that are harder to hit, or targets that are no longer visually or spatially distinct, thus requiring users to pay more visual attention during interaction. Both of these problems would reduce overall system usability. A final issue is that third-

party software providers would need to follow these global design restrictions on target placement, which is both impractical and unenforceable. Given these challenges, it is more reasonable to assume application independence, and to suggest only that within an application, interface objects are positioned to allow users to complete a logical task or subtask while maintaining a single grip.

Because the thumb movement study design focused on movement time, rather than interaction with specific hardware, such as buttons or touchscreens, this recommendation applies equally well to hardware and software layouts.

Bottom Line. Grip adjustment has a high usability cost because it increases the risk of dropping a device, and requires a user to devote both physical and attention resources to an activity that is not directly related to the task at hand. The placement of input hardware on non-touchscreen devices in many cases already supports a single grip for full device interaction. Since touchscreen devices are less consistent in their support of a single one-handed grip, device designers should consider offering a generalized input strategy that can be used across all applications, independent of their design. While ThumbSpace is an example of a software-based solution, another option is to support access to all interaction objects via indirect input hardware such as a scrollwheel or directional pad.

(C2.2) Allow users to configure tasks for either left or right-handed operation especially when they involve diagonal movements.

This design recommendation is derived from the movement direction results from my thumb movement study, reported in Chapter 3, Section 3.2.7, and summarized in paragraph 1 of Section 3.2.8. The data show that for all devices, movement in the ↙

direction was significantly faster than movement in the ↘ direction. Since all participants were right handed, there is reason to believe that this systematic result is due to the fact that ↙ movements were biomechanically easier to perform than ↘ movements.

The natural arc movement that results from rotating the thumb around the base joint is tempting to take advantage of in interface design. As an example, Windows Mobile 5 Pocket PC Phone Edition, whether intentionally or not, leverages this movement for unlocking the device by requiring users to hit the left menu button, and then a software button positioned halfway up the right side of the screen. Based on personal experience, this design can inspire warm feelings that the designers "got it right". But what about the user who has her right-hand occupied? Or the left-handed user that feels more in control when using a device in her left hand? This same design would require the left-handed user to perform the unlock action using a movement style that, for mechanical reasons, is likely to be slower and more awkward than most other options. Unlocking a device may at first appear to be an isolated and relatively infrequent action, but many users do this task, literally, dozens of times a day.

One possible remedy to avoid mismatches between an interface design and the particular hand used to perform the task is to design towards the ↕ or ↔ movement directions, which are more mechanically agnostic to the hand that is used than are diagonal movements. For example, a device unlocking scheme that required users to press a soft key at the bottom center of the screen, followed by one at the middle center of the screen (e.g., the ↑ direction), should be equally usable for both right and left handed users. An alternative approach might have users designate their hand preference as a global setting, and then allow interfaces to adapt dynamically to accommodate the

hand preference of the user. An example of this approach would be to remap the ACTIVATE and CANCEL operations of the gesture language introduced in Chapter 3, Section 3.4.1, from the ⤢ directions to the ⤡ directions, to make these frequent commands easier to perform for left-handed users. As devices become more sophisticated, one can imagine that devices will be able to sense which hand is holding the device, and automatically adapt interfaces accordingly. Of course, it may be impossible for a system to know for certain whether the user intends to operate the device with the thumb of the hand in which it is held, in which case the interface should adapt, or whether the user intends to use the opposite hand (e.g., with an index finger or stylus), in which case adaptation would be unnecessary. Furthermore, Shneiderman [135] and others have advocated that stable interfaces promote positive feelings of being in control, and so are generally considered more usable designs.

Bottom Line: Interfaces that use diagonal thumb movement can feel very comfortable and natural, but only when used with a specific hand. Interfaces that cause users to move their thumbs along a diagonal run the risk of annoying users who use the "wrong" hand out of preference or necessity. A safe recommendation is for designers to strive for hand-neutral designs that focus on the ↕ or ↔ movement directions. Otherwise, designers should allow users to specify hand preference as a setting, and adapt interfaces consistently to accommodate the users' stated hand preference.

(C2.3) Strive to place interaction targets toward the vertical center of the device to support a stable, balanced grip, as well as toward the middle (horizontal center) of the device to make them easy to reach for both left and right handed users.

This design recommendation is supported by both subjective preference and target location data from my thumb movement study, reported in Chapter 3, Section 3.2.7, and summarized in paragraphs 2 ,3 and 4 of Section 3.2.8. Not only were users more comfortable interacting with the center regions of each device (both vertically and horizontally), but they were also able to move their thumbs more quickly in those regions, suggesting users were more physically capable of interacting with central regions than peripheral regions.

While the recommendation to place targets toward the center of a device follows quite naturally from the empirical data, the guideline may seem impractical for touchscreen-based devices whose screens often take up the majority of the surface real estate (see, for example, the Cingular 8525 in Figure 50a); for these designs, placing interaction targets toward the center of the device (e.g., vertically and horizontally) would mean placing them in the middle of the display, but of course, this is where *content* is typically displayed! Thus for touchscreen devices, it is helpful to interpret this design guideline in terms of *interaction*, rather than target placement. For example, gesture-based interaction, such as dragging a list of items with the thumb in LaunchTile (Chapter 4, Section 4.5.6), or the highly touted finger flick for scrolling lists on the iPhone, can allow users to hold a device with a stable grip and to interact with the area of the screen that is most comfortable for them to reach. Furthermore, in both cases, users have control

over the final position of the scrolled content, and so can ensure a specific list item is within thumb reach for direct selection.

The gesture-based command language for controlling the AppLens cursor in Chapter 4, Section 4.4.4 is an example of how interfaces that were not designed with single-handed thumb use in mind might be adapted so that users can access all interface objects, regardless of their sizes or positions, by interacting only with the region of the screen that is comfortable and easy to reach. Of course, ThumbSpace (Chapter 6) offers users the same convenience by allowing each to define the area of the screen they are comfortable using, and then mapping all interaction targets into that space. Based on the successes of the above examples I can offer two design strategies for tailoring my recommendation for centralized interaction to touchscreen devices: 1) offer flexible interfaces that allow users to move content (e.g., by dragging) quickly and fluidly so that it may be reached easily for direct interaction, or 2) offer indirect interaction methods (e.g., gestures) that allow users to access and action upon objects that are not within immediate reach of the thumb, and which still allow users to hold the device with a stable, centralized grip.

For non-touchscreen devices, the recommendation to centralize interaction targets is more reasonable, as many of these devices devote at least half of the display to hardware buttons. For example, Figure 50c,d shows two popular non-touchscreen phones, which both place the most frequently used interaction feature—the directional navigation hardware—at the center of the device (both vertically and horizontally), together with the hardware to support other common actions, including the menu, back, home and call buttons. Contrasting these designs to the touchscreen-based Cingular 8525 (Figure 50a),

whose directional hardware is placed at the bottom of the device, it is clear that the first two support a more centralized, stable one-handed grip. It is worth noting, however, that not all touchscreen-based devices position their navigation hardware on the periphery. The iPAQ hw6500 shown in Figure 50b is an example of a device that balances the benefits of a relatively large touchscreen, with the stability offered by centralizing frequently used hardware buttons.

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

Figure 50. Examples of different hardware design choices for smartphones. The Cingular 8525 (a), the iPAQ hw6500 (b), the BlackBerry Pearl (c), and the Samsung Blackjack (d).

Bottom Line: It is generally unnecessary to restrict the placement of interaction objects on devices that are small enough that most surface regions can be accessed using a single grip. For larger devices, two design goals should be considered: 1) to support users in operating the device with a stable grip; and 2) to accommodate left and right-handed users equally. One way to address these goals is to position interaction objects on the surface of the device centrally, in both the vertical and horizontal directions. For touch-sensitive screens, indirect gesture commands can allow users to issue navigation and control commands wherever their grip is the most stable and comfortable. Alternatively designers might consider allowing users to drag interaction objects to

within direct reach of users' most stable grip, or use software techniques like ThumbSpace to limit the total interaction area required. Many of today's non-touchscreen devices devote the only half the surface area to the screen so that indirect interaction hardware can be offered in the lower half. However, the general trend toward larger screens will likely squeeze such hardware out of the central regions; while keypads are likely to be hidden beneath the screen and pulled out only when necessary, directional navigation hardware might be relocated to the sides of the device to support stable one-handed interaction in the general case (e.g, the Blackberry scrollwheel). This strategy can also be beneficial for increasing the one-handed use options on touchscreen devices, with the added benefit that fingers then do not interfere with the screen contents during interaction.

(C2.4) Favor small devices in order to support overall device control and satisfaction.

This design recommendation is backed by both the subjective preference and target location results from my thumb movement study, reported in Chapter 3, Section 3.2.7, and which are summarized in paragraph 5 of Section 3.2.8. Overall, users preferred smaller devices to larger devices when interacting with only the thumb of one hand. Furthermore, the quality of user interaction, as measured by task speed, was more consistent across the surface of the smaller devices than it was for the larger devices.

This recommendation has some interesting implications for both touchscreen and non-touchscreen devices. The trend over last several years has been that of devices getting smaller and slimmer, no doubt to make them more portable and more likely to be carried at all times. Touchscreen-based devices have remained somewhat larger than their

non-touchscreen counterparts, perhaps because most have software interfaces that are designed to resemble those of the desktop, and this added complexity simply demands more pixels. Yet if we focus only on one-handed control, perhaps it is the non-touchscreen devices that should be made larger, and the touchscreen devices that should be made smaller. The key is that for any type of device, users have a finite range of thumb motion, which limits how large the interaction area of a device can be while still being manageable with one hand. For touchscreen devices, the entire touchscreen area should fall within this range to support full control of the interface. Otherwise, users should use solutions like ThumbSpace, gesture-controlled cursors that allow them to confine interaction to a reachable sub-portion of the screen, or even fall back to using indirect input hardware.

However, for non-touchscreen devices, only the buttons and navigation hardware are required to be within easy thumb range for users to control the device with one hand; since users don't interact with the screen, its size is irrelevant and so could be quite large in theory. In practice, non-touchscreen devices are small because they are designed to be as portable as possible. A nice solution that allows phones to have larger screens while maintaining a small overall package for portability is the clamshell or "flip" design.

Looking forward, convertible devices—those whose keyboards or keypads slide out from under the screen—have perhaps the greatest potential for dominating in the mobile device design space. By hiding keyboards, the screen can occupy the majority of the device surface; the overall device size is then constrained only by user requirements for device portability. When users then need to perform tasks for which physical keys are the most effective means, such text entry, they can slide out a Qwerty keypad. This

combination offers users the best of both worlds: a large screen that places high value on pixels for the display of information, and a full Qwerty keypad for efficient text entry.

Bottom Line: When a device's interaction model requires users to interact with the surface of the screen, ideally the size of the device would be limited to one users can manage comfortably with one hand. Otherwise designers should follow the guidelines suggested for C2.1 and C2.3 above.

(C2.5) Make touchscreen targets 1cm^2 to support fast, accurate one-handed selection.

This design recommendation is derived directly from the task time and error results of my target size study, reported in Chapter 3, Sections 3.3.6 and 3.3.7. The raw data indicate that of the targets tested (3.8, 5.8, 7.1, 9.5, 11.5, and 13.4 mm), targets 9.6 mm^2 strike an effective balance between speed, error rate, and user preference for both single (discrete) and multiple (serial) target selection. Data collected on the precise locations of thumb contact when users aimed for a target, together with user satisfaction feedback, indicate that targets as small as 9.1 mm^2 may be equally effective. Yet because 9.1 mm^2 targets were not specifically featured in the study design, a safe and realistic guideline for practitioners is to strive for 1cm^2 targets. For interfaces that do not adhere to this recommendation but which still want to support one-handed thumb interaction, additional tools are advised. For example, Shift [147] supports high-accuracy, direct selection of small targets. For targets that are out of thumb reach, indirect methods are more appropriate, such as controlling a cursor using on-screen gestures or directional navigation hardware, or using ThumbSpace.

It is worth noting that my recommendation for 1cm^2 thumb targets applies strictly to interaction with resistive touchscreens—the technology used for the vast majority of today's touchscreen-based devices. The iPhone and Prada LG phones, on the other hand, are the first mainstream devices to have highly sensitive capacitive touchscreens, which register more information about the nature of the finger touch than do resistive touchscreens. While resistive touchscreens report only a single pixel that represents the average of the finger contact area, capacitive touchscreens can report the entire contact area of the finger. The gentle finger flicks that allow lists to be scrolled on the iPhone, as well as research on interaction techniques for multi-touch displays (e.g., [19]) promise that mobile devices will become more sophisticated in their ability to interpret user intention from thumb "signatures", or rather, the characteristics of their contact areas. The additional data captured by capacitive screens together with new algorithms should allow users to select small targets more accurately than they can with today's resistive touchscreens, and this fact may lead to new recommendations for target sizes that are specific to capacitive screens. A problem that capacitive screens will still share with resistive touchscreens, however, is that users cannot visually verify selections when targets are smaller than the finger. Thus even if capacitive screens can better predict user intentions during selection, the problem of finger occlusion is likely to place practical limits on how small targets can be made for capacitive screens while still satisfying users.

Other potential benefits of capacitive screens could be in predicting right vs. left-handed use based on the shape of the thumb's contact area, interpreting a user's target intentions by inferring the user's angle of approach, and even distinguishing between one-handed thumb use and two-handed index finger use. The advanced capabilities

offered by capacitive touchscreens will give users much more power in controlling touchscreen devices with one hand, but will require the development of a new set of technology-specific design recommendations for one-handed thumb interaction.

Bottom Line: If fast, accurate one-handed operation is the design goal for a resistive touchscreen device, software interfaces should be composed of 1cm^2 targets. If one-handed operation is not the primary design goal, software techniques like Shift and ThumbSpace allow designers more flexibility in screen design, while still supporting relatively fast and accurate one-handed interaction. If stable device grip and selection accuracy are valued above average interaction time, indirect input hardware options such as direction pads offers a third option for supporting one-handed use.

(C2.6) Gestures should be offered to complement, rather than replace, direct interaction.

This design recommendation is derived from observational data gathered during the comparative study of LaunchTile and AppLens, reported in Chapter 4 Section 4.7. While we know from the results of the gesture study of Chapter 3, Section 3.4.7 that simple sets of command-mapped gestures can be learned and executed with relatively little training, observations of users interacting with LaunchTile and AppLens indicated that users were much more likely to try to hit small or far targets directly than use the gesture options at their disposal. In both cases, increased interaction time may have been the common reason why users avoided gestures. In AppLens, incremental movement of the selection cursor was certainly slower than hitting targets directly, and sluggish animation times may have made object dragging unattractive for user of the LaunchTile interface.

Yet cursor control and object dragging aren't the only possible uses for gestures. Symbolic gestures have the possibility of actually reducing the number of steps required for users to accomplish their goals. For example, making on-screen gestures such as ✓ or × might be faster than using a menu system for accepting or rejecting movie recommendations [110]. Another example is the ability for users to delete contacts on the iPhone with either a three-tap sequence, or a natural (and discoverable) cross-thru gesture (→) plus a tap. I believe these two examples represent an ideal for gesture-based interaction: 1) users are initially provided with a straightforward method to perform a task, which is visually obvious as well as easy to learn and use; and 2) as users become more experienced, they can learn, be shown, or discover a gesture sequence that allows them to do the task even faster than before. This role for gestures can be likened to keyboard shortcuts on the desktop, which may take some time and expertise to learn, but ultimately allows users to be more effective in performing frequent tasks.

Bottom Line: Avoid using gestures as the primary interaction model for touchscreen devices. However, discoverable and intuitive gestures that improve the speed of a task can increase the enjoyment and effectiveness of an interface.

7.1.2 Applications: Interaction Techniques for One-Handed Data Access

Contribution (C3): Inspired novel application management methods for small screens based on fluid context switching, rich adaptive application representations, one-handed access, and device independence.

This contribution is based on novel design goals of the AppLens and LaunchTile interfaces, described in Chapter 4, Sections 4.3 and 4.5, and the positive feedback they received during their evaluation, which is described in Section 4.7.6.

The most valuable aspects of the AppLens and LaunchTile designs may have had little to do with their support for one-handed interaction, since study participants seemed to be quite amenable to both styles of design and interaction. More innovative, perhaps, were their shared design strategy of displaying dynamic notification information for multiple programs at once, and the ability to rapidly navigate among items to investigate more detailed information. Although at the time AppLens and LaunchTile were developed, few phones or PDAs were Internet enabled, today a significant percentage of users browse the Web from their mobile devices. Most devices offer browser experiences similar to those of the desktop, but unfortunately the browser model is ill-suited to the input and output constraints of mobile devices; slow mobile text entry makes URL input tedious, and web pages adapted from desktop formats to mobile platforms do not always lead to the most usable results, either because they require substantial scrolling, or the expected content is hard to find due to page restructuring. So even though more phones are equipped to access the Internet than ever before, the mismatch between the traditional browser-based approach and the limited capabilities of mobile devices prevents users from taking full advantage of the phone's Web connectivity for effective content browsing.

If we instead consider the approaches of AppLens or LaunchTile, which both offer a landscape of application tiles that display content at varying levels of granularity, and which update automatically, the result is a framework that brings content to the user, rather than requiring the user to hunt it down. This framework removes the effort and tedium that makes phone-based web-browsing so unattractive today, and instead allows users to focus their interactions on the browsing experience and the information content.

This design is consistent with the goals of minimal attention user interfaces (MAUIs) [113] because content is updated without the need for any user intervention; user interaction is distilled to high-level browsing, and making navigation choices about the subset of information that is of interest, rather that wasting effort reiterating what content is interesting (e.g., selecting from a favorites list) or being involved with the low-level mechanics of telling the system where to find content (e.g., entering URLs). As the increasing numbers of web services and content are made available to small devices, the approach of low-effort, parallel delivery, assistive browsing has the potential to greatly decease the amount of energy users are required to expend to access web content, and in so doing, greatly improve the utility of devices, as well as user satisfaction and enjoyment in accessing networked content. We will see how well these predictions hold up once systems like Zumobi (www.zumobi.com, a commercialization of LaunchTile) are made available for public use.

Bottom Line: Minimize the amount of required interaction for users to operate a device. Benefits include increased task flow and user enjoyment, and reduced mental demand, attention demand, and user frustration. One strategy is to minimize the number of steps required to perform common tasks, such as placing a call on a phone. Another might be to preemptively fetch content that is known to be of interest to the user, which insulates the user from the slow connection speeds of most devices.

Contribution (C4): Offered a one-handed approach to searching the large volumes of data from a mobile phone without the need for text entry.

This contribution is based on the speed advantage that attribute navigation offers over mulit-tap text entry for performing browsing tasks using the FaThumb interface, described in Chapter 5, Section 5.5, and evaluated in Section 5.6.

Phones equipped with full Qwerty keyboards allow users to enter text substantially faster than when using the standard methods of multi-tap or predictive text entry on a numeric keypad. The drawback for mobility, however, is that Qwerty text entry requires substantial physical, visual and mental resources: 1) users generally use two thumbs to improve the parallelism, and hence speed, of entry; 2) the large number of keys requires users to visually verify key presses; and 3) the act of composing characters into words, and words into sentences demands mental attention. Attribute navigation, on the other hand, may be *less* physically, visually, and mentally demanding for the following greason: 1) since attribute navigation only ever requires one hand, it demands fewer physical resources, 2) the smaller number of potential targets (nine) may allow users to devote less visual attention to the task, especially if parts of the attribute hierarchy have been memorized; and 3) since users navigate data attributes via recognition (rather than recall), the process may be less mentally demanding than generating one's own keywords. Of course, the two search methods are complementary in practice; for example, convertible device designs that feature a touchscreen interface and a hidden slide-out keyboard can offer one-handed attribute navigation when the device is closed (e.g., while walking), and then offer the advantages of fast text entry when users are able to open the device and devote more resources to the task.

Although attribute navigation is a promising method for increasing mobile accessibility for tasks that have traditionally required text entry, practical technical challenges may stand in the way of its success. The value of attribute navigation is in no small part due to the rapid updates to the result set that occur as users navigate through the attribute hierarchy, or enter alphabetic characters for incremental text-based filtering. If the data set does not reside on the device, each of these user actions will require a round-trip query to a remote server, which will no doubt slow down response time substantially, and so jeopardize any advantages that attribute navigation may have over keyword text entry for searching data sets off the device. Luckily, there may be valuable uses for facet navigation that don't suffer this drawback. New devices already support several gigabytes of local data, and even that is likely to increase. Because small screens can only display a handful of items at a time, the traditional method of navigating local data sets via folder hierarchies will become increasingly time consuming and tedious. Much as the facet-based Phlat [39] search interface offers users benefits in finding local data on the desktop, a FaThumb approach that uses data attributes such as data type, date of creation, file size, and so on, may offer a much needed improvement over folder hierarchy navigation for finding data stored on a device. And because the target data set resides locally, users will not need to contend with the latencies inherent in performing multiple round-trip sub-queries to the server to satisfy a single data search need.

Bottom Line: Search is an increasingly indispensable tool across all computing systems. Optimizing search interaction and result sets are especially important for mobile devices because the impoverished input and output channels can quickly frustrate users when trying to perform complex tasks such as data search. A wide range of solutions

might ease user burden in specifying data needs, including faceted attribute navigation, cached queries, location-based context, recommended or collective queries, and camera-based search term input. The value of search output might also be improved, for example, by taking into account location-based search context, previously seen items, and even activities that users have performed at their desktop PCs (e.g., "work context").

Contribution (C5): Offered a generalized one-handed approach for interacting with touchscreen applications designed for rich presentation and stylus-based interaction.

This contribution is based on the experimental results supporting the benefit that the ThumbSpace interaction widget can offer users in accessing objects that are too small or too far away to be hit reliably with the thumb on touchscreen interfaces. The ThumbSpace interface is describe in Chapter 6, Section 6.2, the experimental results are in Section 6.7 and the discussion of results is in Section 6.8.

Through the iterative design and evaluation of ThumbSpace, I was able to identify both the design characteristics and use scenarios for which ThumbSpace can provide real benefits to users for one-handed operation of stylus-oriented designs. Yet based on the fact that I only began to observe benefits in using ThumbSpace over Shift [147] when the touchscreen size increased from 2.8" to 3.5", it may be that techniques like Shift, which focus on the problem of thumb occlusion, actually go a long way to solving the most immediate problems of thumb interaction on touchscreens.

It is also appropriate to reconsider the role that peripheral hardware interaction serves in controlling a touchscreen device with one hand. In my study of ThumbSpace, I only investigated the relative value of touchscreen-based interaction (e.g., DirectTouch,

Shift, and ThumbSpace) when compared to today's most popular hardware navigation features (e.g., ScrollWheel and D-Pad). It is important to reiterate that the D-Pad had a better success rate than either Shift or ThumbSpace, and supported comparable target access times to Shift and ThumbSpace for about half of the targets. The D-Pad was more competitive than the ScollWheel in both speed and subjective preference presumably because it allowed users to move the input cursor in 2D rather than just 1D. Perhaps this advantage is what has lead RIM to phase out the side-mounted thumb wheel on Blackberry devices, in favor of front-mounted trackball (Figure 50c). The increased speed that trackballs and isometric joysticks can offer over traditional D-Pads means they have the potential to beat ThumbSpace and Shift in both accuracy *and* speed, while at the same time supporting localized interaction for increased stability during mobile use. These factors may signify the very real possibility that indirect hardware interaction can pull out ahead of touchscreen-based interaction for one-handed touchscreen device operation. Nevertheless, large touchscreen devices such as the iPhone give up on directional hardware entirely in favor of touch only, in which case the need for touchscreen-based solutions like ThumbSpace may also *increase*.

Bottom Line: Through the development of ThumbSpace, I have confirmed that users are willing and able to learn novel interaction techniques. Furthermore, they are able to intuit the relative values of different techniques in terms of access times and errors and make optimal decisions about the input method to use for performing abstract selection tasks. In practice, however, ThumbSpace showed only modest performance benefits for a minority of interface items. Furthermore, the interface I studied can be considered pessimistic in the sense that interface objects were very small, and distributed

uniformly over the surface. In practice, the majority of interfaces feature a range of target sizes, and the small, inconveniently placed objects are not typically the ones users access most often. Finally, invoking ThumbSpace requires more mental demand than aiming at items directly with the thumb, given that forethought is required to launch ThumbSpace and then users must perform several mental steps to translate and track the goal target from the original display, to ThumbSpace, and then back to the original display. Although users were willing to make these extra calculations when the activity was abstract and unrelated to a real-world user task, it is less likely users will be able or willing to interrupt goal-oriented activities with on-the-fly cost/benefit assessments of launching ThumbSpace. When these issues are considered along with the extra time required to launch ThumbSpace, it is difficult to imagine a practical scenario in which ThumbSpace should be recommended over Shift. The momentary device instability users may experience when accessing far targets, and the physical awkwardness of accessing near targets, seem to be relatively minor costs by comparison to launching and using ThumbSpace, especially because those tasks may constitute only a small percentage of total interaction time, and even so, may pertain to only a subset of the population. That said, since users choose for themselves whether to use ThumbSpace, there is little downside to making ThumbSpace available on devices, especially since it may offer a disproportionately large value to users with small hands, and those who enjoy it for the sake of novelty.

7.2 Future Work

Given the anticipated growth potential for touchscreen-based phones, and advances in touchscreen technology, there exist significant opportunities for innovation as it pertains to the one-handed usability of touchscreens.

7.2.1 ThumbSpace

Following directly from my own research of ThumbSpace, several outstanding questions remain to be answered. The first is whether ThumbSpace can be extended to support the full range of interactions that today's touchscreens interpret regularly, including tap+hold (e.g., pop-up menus) and object dragging (e.g., scroll bars); as it stands, ThumbSpace only supports the equivalent of stylus-tap functionality. Without mechanisms for users to drag objects or open context-sensitive menus, ThumbSpace falls short of its goal to provide generalized one-handed access to arbitrary touchscreen interface designs.

A second research direction for ThumbSpace is to investigate an appropriate and low-cost method for triggering ThumbSpace. My final evaluation of ThumbSpace aimed to eliminate the details of triggering ThumbSpace so that we could understand the value of the interaction method independent of the unique costs associated with a particular trigger mechanism. Because of this, the final evaluation of ThumbSpace should be considered a best-case comparison of ThumbSpace and Shift. Further research is required to determine whether trigger methods can be found which reduce as much as possible the physical cost of triggering ThumbSpace, and to understand whether users still chose to use ThumbSpace when using a realistic trigger method.

Finally, ThumbSpace was only evaluated for artificial target-selection tasks that featured very small targets. In reality, interfaces can and do feature larger targets, complex widgets (e.g., multi-selection combo boxes), and non-uniform object layouts—any of which have the potential to reduce the perceived utility of ThumbSpace. Thus the true viability of ThumbSpace remains in question until: 1) tap+hold and object dragging are supported via ThumbSpace interaction; 2) ThumbSpace is implemented for use with real-world interfaces—that is, integrated at the operating system level; 3) ThumbSpace is coupled with a realistic mechanism for on-demand launching; 4) ThumbSpace is evaluated for performing real-world tasks with common touchscreen interfaces, including email, contacts, appointments, and web browsing; and finally 5) ThumbSpace is again compared to the most competitive hardware alternatives available (e.g., trackballs, isometric joysticks, and touch pads).

7.2.2 Mobility

One limitation of my thesis work is the lack of situational realism in my evaluations of interfaces and interaction techniques. While controlled experimentation is important for understanding comparative benefits of competing interface designs and techniques, it offers little proof that these benefits will hold up during actual use scenarios. In particular, except for the field study and web survey, all of my evaluations were conducted indoors, and with users typically either standing or seated. In only one experiment did I ask users to walk while performing tasks. Even then, users had little need to pay attention to their surroundings since they were following a figure 8 outline on the floor. In practice, environmental factors such as lighting, noise, physical obstructions,

holding other objects, among many others, can negatively impact the usability of a system. As an example, users must visually monitor the display for feedback during touchscreen interaction because the uniformly flat surface doesn't offer tactile feedback. Unfortunately, high light levels can wash out screen details when touchscreens are used outside, making tasks difficult or impossible to perform. Furthermore, true mobile scenarios may pose more challenges for device stability than the laboratory setting.

Many researchers have strongly advocated the evaluation of mobile systems under ecologically valid conditions [10, 24, 46, 115]. Given the somewhat sterile laboratory conditions of my past studies, it is clear that an important next step in my research is to evaluate some of these applications, such as FaThumb and ThumbSpace, in more naturalistic settings and as part of longitudinal studies.

7.2.3 High-Sensitivity Capacitive screens

A final area for further exploration is to understand how the lessons learned over the course of this dissertation research relate to new touchscreen technologies. Until a few months ago, resistive touchscreens were the de facto hardware used in touchscreen-based devices. Yet the fervor surrounding the iPhone's capacitive multi-touch screen as well as various anticipated releases by other companies indicates that we will see a dramatic shift toward mobile devices equipped with more sensitive, responsive, capable touchscreens. Not only might these new devices foster innovations in interaction that were simply not possible before, but it will also be valuable to try to quantify whether the findings from studies of resistive screens are upheld when transferred to capacitive screens. For example, it is likely that my recommendation for $1cm^2$ targets for thumb-oriented resistive touchscreen interface designs will be invalid for capacitive touchscreens. But

other relationships are harder to predict. For example, will the advantages and disadvantages of ThumbSpace vs. Shift hold even for capacitive touchscreens? One option is to revisit each of my studies using more sensitive screens and see if the results are the same. But another approach is to study the fundamental characteristics of the two technologies and strive to develop basic rules to predict the comparative outcomes without needing to rerun past studies. But if one thing is certain, it is that the additional data provided by capacitive touchscreens hold great potential for highly responsive, effective, finger-oriented mobile interfaces.

7.2.4 Understanding Software/Hardware Tradeoffs

In Chapter 6, Section 6.3 I discussed the potential advantages that indirect input hardware, for example an indirect navigation pad, has over direct interaction with a touchscreen: 1) indirect input can lead to more stable device use assuming it allows for a constant, comfortable grip; 2) indirect input avoids the problem of fingers occluding the display; and 3) indirect input allows for an interaction model that separates selection into two phases, aim/focus and selection, which lowers the visual and possibly mental demands of tasks so that user may pay more attention to the mobile environment. Yet when I compared two indirect input methods (a directional navigation pad and a scrollwheel) to the touchscreen input methods ThumbSpace and Shift (Chapter 6, Section 6.5) the indirect methods were less preferred than was Shift, despite having comparable or higher accuracy. Thus the additional benefits (1-3 above) that the indirect methods should have offered users over the touchscreen methods either went unnoticed were not appreciated enough to have offset the negative influence of the significantly higher average task times when using the indirect methods. One possible reason that the indirect

methods were undervalued by users is that the walking mobility condition studied had surprisingly little influence on user performance. This may have been because a) the device was easy to control in one hand; b) the target selection task was not mentally challenging; or c) walking was self-paced and did not demand much visual attention. A more difficult task or mobility scenario may have increased the perceived utility of the indirect input methods. Furthermore, the fact that users were able to access a subset of the interface items as fast or faster with the indirect methods as with the touchscreen methods suggests that the large number (48) and layout (2D) of the interface objects may have biased users toward the touchscreen methods which, in contrast to the indirect hardware methods, did not require users to traverse any intermediate objects when selecting a target. A different interface design, such as one featuring only a small number of distant objects, or a 1D text list, may have yielded considerably different performance and preference results.

In considering the results of the ThumbSpace studies more broadly, it is clear I have learned only that touchscreen methods *can* benefit users, especially when accessing small items arranged in a dense 2D layout, and not that touchscreen methods outperform indirect methods in general. In fact, there is little doubt that the efficacy of an input technique is strongly influenced by the characteristics of the interface design and user tasks. For example, while my own study found the scrollwheel to be the least preferred and slowest input method on average, RIM's Blackberry has been a leader in the corporate mobile device market for years, even though, until recently, Blackberries have supported object navigation and selection exclusively via a scrollwheel (more recent models have replaced the scrollwheel with a trackball). The success of the Blackberry's

scrollwheel-based interaction can be attributed in part to the design of the associated interface software, which has been optimized for scrollwheel interaction—often favoring visually simple, linear layouts, and intuitive contextual menus. In addition, physical properties of the scrollwheel support stable one-handed use and eyes-free interaction.

Unfortunately few public guidelines exist that might help industry players (e.g., hardware manufacturers, interface designers, software architects) to make informed decisions about how hardware and software choices influence mobile device usability. Extensive further research is required to understand the complex interactions among the myriad characteristics of the interface design (e.g., size, layout, and placement of interaction objects), task (e.g., level of mental and interaction demands), input technique (e.g., direct vs. indirect input, pixel vs. object pointing, amount of physical exertion required), information structure (e.g., broad and shallow vs. narrow and deep), mobile scenario (e.g., lighting, attention requirements, stability), and how the costs and benefits change as users move between one and two-handed use scenarios. Only through the careful study of these factors can we begin to understand the design tradeoffs that contribute to safer, more efficient mobile system designs.

Due to the cost constraints of offering mobile devices on a massive distribution scale at affordable prices, most devices to date have included relatively inexpensive, unsophisticated input hardware options by comparison to desktop and laptop computers. But as technology advances increase the feasibility of bringing more capable hardware to users at lower cost, it will be useful to revisit the role hardware can play in addressing the persistent issues of interaction efficiency, user comfort and safety during one-handed mobile computing. One question of particular relevance to my thesis is whether an

indirect, non-touchscreen, hardware input solution will emerge that successfully delivers on the theoretical benefits I have suggested such input might offer over touchscreen-based interaction. If so, the rising trend in touchscreen popularity may slow dramatically as manufacturers instead turn their favor toward richly designed, non-touchscreen displays with interaction models reminiscent of desktop and laptop interfaces. To illustrate both the potential and uncertainty of hardware-only solutions, I'll briefly discuss some of the usability tradeoffs that the touchpad and isometric joystick (two highly successful input technologies used in laptop computing) might pose for users in the mobile domain.

Touchpads

As one of the most common methods for entering 2D positional input on laptop computers today, the touchpad offers users the capabilities of a mouse within a finite and fixed surface area. Because a touchpad can be used in either absolute or relative input mode, it could be used quite similarly to ThumbSpace, but with the added benefits of being more sensitive and specialized to finger interaction (presumably leading to higher selection precision), and avoiding the problem of fingers occluding display content. Additionally, a touchpad version of ThumbSpace could be modified so that object selection is divided into two distinct steps: 1) positioning the object cursor with the touchpad, and 2) pressing a button to activate the object under the cursor. The benefit of a two-step selection approach is that cursor state is maintained between touchpad interactions, which gives the user more freedom to interleave the visual requirements of positioning the cursor and those of monitoring the dynamic mobile environment. Of course the touchpad could also be used in standard *relative* input mode to position a

mouse pointer on the device screen, whereby a separate button press would perform a mouse "click" and activate the object under the pointer.

One drawback of the touchpad is that it occupies surface real estate that cannot then be used for displaying information content. It has been suggested that a touchpad could overlay a numeric keypad on handheld devices such as mobile phones and thus double the utility of the non-display surface area [143]. Others have proposed the back of the device as a viable surface for a touchpad [153], so that the front of the device might be reserved for display space. This design also allows pointing tasks to be performed with the index finger, freeing the thumb for other tasks.

Whether the touchpad is placed on the front or the back of the mobile device, its size and placement must be considered carefully. Since mobile devices are used in both the left and right hands, the touchpad should be equally accessible in both use cases, and so centered horizontally on the device. One possible complication, however, is that as the size of a device grows, the touchpad will need to shrink in order to accommodate the reach limitations of users. For example, consider Figure 51 which shows that a touchpad would need to shrink when moving from a smaller device to a larger device to accommodate a user's maximum horizontal reach of length x.

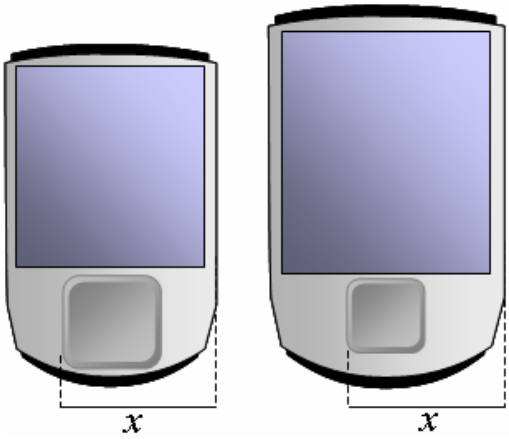


Figure 51. A depiction of how the size of a touchpad might change to accommodate left and right handed use for two different screen sizes. Assuming a user's maximum horizontal reach is of length *x*, touchpads will need to shrink as devices grow larger in order to accommodate one-handed operation.

Accessing a larger screen with a smaller input region will tend to decrease user precision during object selection for input techniques like ThumbSpace. Otherwise, when a small touchpad is used in relative mode to position a mouse pointer, "clutching" may be necessary to access all screen positions—similar to running out of desk space (or arm reach) when using a desktop mouse with a large-screen display. However, these *potential* limitations of the touchpad need to be evaluated within the context of practical device dimensions to understand whether they pose practical challenges for mobile Alternatively, ergonomic shaping or weighting of the device might allow users to hold the device safely from the bottom center, and thus offer designers more flexibility in setting the size and shape of the touchpad.

Joysticks

An alternative 2D input option that avoids the physical dimension issues associated with the touchpad is any hardware that allows users to issue (up, down, left, right) directional commands to the operating system. The most common such option found on devices today is the four-way directional navigation pad, and less common are joysticks that issue

the same four commands. Since each user action (button press, joystick movement) is interpreted as a single direction command, users can preplan a cursor movement path and then focus their visual attention on the environment while tapping out the associated command sequence. However, both directional navigation pads and their joystick equivalents also support “press and hold”, which issues a command repeatedly at a constant rate when a button or joystick direction is held beyond a time threshold, such as one second. In contrast to the multiple-press approach, “press and hold” requires some visual monitoring because people are generally not very good at estimating the number of items that will be traversed from the length of time a button is held.

While directional pads and joysticks are acceptable means for interacting with simple interfaces that have a relatively small number of accessible objects, they become increasingly tedious to use as the number of selectable objects grows. The problem is exacerbated by the fact that directional pads and joysticks can move a cursor in only four distinct directions, so additional steps are required to move a cursor along a diagonal. Such limitations have inspired selection strategies that, for example, allow users to jump a cursor to a specific region of the display using a numeric keypad before fine-tuning the selection [57]. Of course, it cannot be assumed in general that alternative hardware will be available for increasing the input efficiency of directional pads and joysticks, and so they will ultimately fail to provide the power necessary to interact with and control rich mobile interfaces.

The isometric joystick is a sophisticated alternative to the directional navigation pad which has no moving parts but instead interprets directional force for controlling the direction and speed of cursor movement. The fact that users can modulate cursor speed

by adjusting the force applied allows users to move the cursor faster for accessing distant objects, which largely solves the problem of slow cursor movement users can experience with directional pads. Another advantage the isometric joystick enjoys over the directional navigation pad is 360° directional input, which allows users to move the cursor along efficient diagonal trajectories. The commercial success of the isometric joystick as the red TrackPoint "eraser" on IBM's line of ThinkPad laptops has proven it to be an efficient alternative to a mouse for 2D pointing tasks on laptop-sized screens, and it's reasonable to assume the benefits would also transfer to the smaller screens handheld devices.

Lab studies have already shown that users can effectively control an isometric joystick with one hand while standing [90], but there are reasons to question whether isometric joysticks are suitable for mobile scenarios. First, isometric joysticks offers users a great degree of power in a tiny footprint, but using one comfortably and effectively requires both practice and finesse; the sensitivity that makes isometric joysticks so useful in stationary conditions may make them difficult to control with unstable hands and devices while users are mobile, resulting in higher errors or longer performance times. A second reason that isometric joysticks may be inappropriate for mobile use is that they do not support eyes-free use well. This is due to the fact that isometric joystick interaction involves a complex process of composing two continuous parameters (direction and force); since the joystick is not displaced physically during use, it offers no haptic feedback with which users predict the associated movement effect on the cursor. The requirement that users visually monitor the cursor movement during the

majority of the joystick use can impair both the mobile activity (e.g., pausing while walking) and the device task (e.g., increased task errors and time).

How competitive indirect input hardware will be in supporting mobile one-handed interaction in future devices will depend on careful attention to ergonomic, software, and information design. Given the potential challenges that the frontrunners from laptop computing may face when adapted to the mobile domain, it remains an open question whether touchpads, isometric joysticks, or a yet-to-be-developed indirect method will be determined to definitively outperform touchscreen-based interaction. One issue that will be import to understand further in the pursuit of "best mobile design practices" is whether the traditional style of pixel-level pointing from PC and laptop computing is an appropriate interaction model for mobile computing. In particular, the characteristics of device instability, high visual demand, and mental distraction inherent to mobile computing may make pixel level positioning impractically difficult, time-consuming, or error prone in practice. In this case, object-pointing [54] might prove to be a more effective interaction model, since it supports pointing at a coarser granularity, and may be less sensitive to the various sources of noise present in mobile computing. Even so, object pointing has its own unique drawbacks, such as the fact that the speed of object selection depends on the number and layout of objects displayed, and its efficiency will vary from one interface to the next. Additionally, both pixel- and object-level pointing share the characteristic that object selection time depends on the (potentially non-deterministic) starting position of the cursor, so that even within the same interface, performance may vary from one task to the next. Touchscreen interaction methods like Shift and ThumbSpace, by comparison, have no concept of a persistent cursor, and so may offer

users more consistent and predictable performance. In addition, both touchscreen methods support a primary ballistic thumb movement, which may prove faster on average than indirect methods. It is apparent, through these reflections, that this thesis constitutes just one component of the body of knowledge that can help guide software and hardware developers toward more effective, more enjoyable, and safer mobile interfaces. In particular, it will only be through further innovation and rigorous comparative study that the relative competencies of indirect hardware and touchscreen interaction methods will be understood well enough to suggest one be used to the exclusion of the other.

Bibliography

1. Ahonen, T.T. *Putting 2.7 billion in context: mobile phone users*. Blog entry of the book Communities Dominate Brands, http://communities-dominate.blogs.com/brands/2007/01/putting_27_bill.html, Accessed on January 8, 2007.

2. Aliakseyeu, D., Subramanian, S., Gutwin, C. and Nacenta, M. Bubble radar: efficient pen-based interaction. *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '06)*, pp. 19-26, ACM Press, 2006.

3. Anonymous. *Going mobile: an international study of content use and advertising on the mobile web*. Online Publishers Association, http://www.online-publishers.org/, 2007.

4. Apitz, G. and Guimbretiere, F. CrossY: A crossing based drawing application. *Proceedings of the Symposium on User Interface Software and Technology (UIST '04)*, pp. 3-12, ACM Press, 2004.

5. Apple *iPod*, http://www.apple.com/ipod/, 2006.

6. APTA. *"Blackberry Thumb" Causing Digital Distress in and out of the Workplace*. American Physical Therapy Association, http://www.apta.org/. Alexandria, 2006.

7. Asano, T., Sharlin, E., Kitamura, Y., Takashima, K. and Kishino, F. Predictive interaction using the delphian desktop. *Proceedings of the Symposium on User Interface Software and Technology (UIST '05)*, pp. 133-141, ACM Press, 2005.

8. ASHT, A.S.o.H.T. *Heavy use of handheld electronics such as Blackberry, iPod can lead to hand ailments*. American Society of Hand Therapists, www.asht.org, Chicago, 2005.

9. Barmakian, J.T. Anatomy of the joints of the thumb. *Hand Clinics*, Vol. 8, No. 4, pp. 681-691, 1992.

10. Barnard, L., Yi, J.S., Jacko, J.A. and Sears, A. An empirical comparison of use-in-motion evaluation scenarios for mobile computing. *International Journal of Human-Computer Studies*, Vol. 62, pp. 487-520, 2005.

11. Bartlett, J.F. Rock 'n' Scroll Is Here to Stay. *IEEE Computer Graphics and Applications*, Vol. 20, No. 3, pp. 40-45, 2000.

12. Baudisch, P., Cutrell, E., Robbins, D.C., Czerwinski, M., Tandler, P., Bederson, B.B. and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems. *Proceedings of the International Conference on Human-computer Interaction (INTERACT '03)*, pp. 57-64, 2003.

13. Baudisch, P. and Rosenholtz, R. Halo: a technique for visualizing off-screen objects. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*, pp. 481-488, ACM Press, 2003.

14. Baudisch, P., Xie, X., Wang, C. and Ma, W.-Y. Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. *Proceedings of the Symposium on User Interface Software and Technology (UIST '04)*, pp. 91-94, ACM Press, 2004.

15. Bederson, B.B. Interfaces for staying in the flow. *Ubiquity*, Vol. 5, No. 27, pp. 1-1, 2004.

16. Bederson, B.B., Clamage, A., Czerwinski, M. and Robertson, G. DateLens: A Fisheye Calendar Interface for PDAs. *ACM Transactions on Computer-Human Interaction*, Vol. 10, No. 4, pp. 90-119, 2003.

17. Bederson, B.B., Grosjean, J. and Meyer, J. Toolkit Design for Interactive Structured Graphics. *IEEE Transactions on Software Engineering*, Vol. 30, No. 8, pp. 535-546, 2004.

18. Bederson, B.B., Meyer, J. and Good, L. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *Proceedings of the Symposium on User Interface Software and Technology (UIST '00)*, pp. 171-180, ACM Press, 2000.

19. Benko, H., Wilson, A.D. and Baudisch, P. Precise selection techniques for multi-touch screens. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, pp. 1263–1272, ACM Press, 2006.

20. Bezerianos, A. and Balakrishnan, R. The vacuum: facilitating the manipulation of distant objects. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 361-370, ACM Press, 2005.

21. Blanch, R., Guiard, Y. and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '04)*, pp. 519-526, ACM Press, 2004.

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

23. Bourbonnais, D., Forget, R., Carrier, L. and Lepage, Y. Multidirectional analysis of maximal voluntary contractions of the thumb. *Journal of Hand Therapy*, Vol. 6, No. 4, pp. 313-318, 1993.

24. Brewster, S.A. Overcoming the lack of screen space on mobile computers. *Personal and Ubiquitous Computing*, Vol. 6, No. 3, pp. 188-205, 2002.

25. Brewster, S.A., Lumsden, J., Bell, M., Hall, M. and Tasker, S. Multimodal 'Eyes-Free' interaction techniques for mobile devices. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*, pp. 473-480, ACM Press, 2003.

26. Broder, A. A taxonomy of web search. *SIGIR Forum*, Vol. 36, No. 2, pp. 3-10, 2002.

27. Büring, T., Gerken, J. and Reiterer, H. Usability of overview-supported zooming on small screens with regard to individual differences in spatial ability. *Proceedings of the working conference on Advanced Visual Interfaces (AVI '06)*, pp. 233-240, ACM Press, 2006.

28. Büring, T. and Reiterer, H. ZuiScat - querying and visualizing information spaces on personal digital assistants. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '05)*, pp. 129-136, ACM Press, 2005.

29. Buyukkokten, O., Garcia-Molina, H. and Paepcke, A. Accordion Summarization for end-game browsing on PDAs and cellular phones. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '01)*, pp. 213-220, ACM Press, 2001.

30. Buyukkokten, O., Garcia-Molina, H., Paepcke, A. and Winograd, T. Power browser: efficient Web browsing for PDAs. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '00)*, pp. 430-437, ACM Press, 2000.

31. Card, S.K., Moran, T.P. and Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, Vol. 23, pp. 396-410, 1980.

32. Chen, Y., Ma, W.-Y. and Zhang, H.-J. Detecting web page structure for adaptive viewing on small form factor devices. *Proceedings of the International World Wide Web Conference (WWW '03)*, pp. 225-233, ACM Press, 2003.

33. Cho, S.-J., Choi, C., Sung, Y., Lee, K., Kim, Y.-B. and Murray-Smith, R. Dynamics of tilt-based browsing on mobile devices. *Extended Abstracts of the Proceedings of the Conference on Human Factors in Computing Systems (CHI '07)*, pp. 1947-1952, ACM Press, 2007.

34. Christian, K., Kules, B., Shneiderman, B. and Youssef, A. A comparison of voice controlled and mouse controlled web browsing. *Proceedings of the Conference on Assistive Technologies (ASSETS '00)*, pp. 72-79, ACM Press, 2000.

35. Cockburn, A., Karlson, A.K. and Bederson, B.B. *A review of focus and context interfaces*. HCIL Tech Report HCIL-2006-09, 2006.

36. Colle, H.A. and Hiszem, K.J. Standing at a kiosk: effects of key size and spacing on touch screen numeric keypad performance. *Ergonomics*, Vol. 47, No. 13, pp. 1406-1423, 2004.

37. Crossan, A., Murray-Smith, R., Brewster, S., Kelly, J. and Musizza, B. Gait Phase Effects in Mobile Interaction. *Extended Abstracts of the Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 1312-1315, 2005.

38. CSP *Physios urge students to improve their text life*. CSP - Chartered Society of Physiotherapy, http://www.csp.org.uk/director/newsandevents/mediareleases.cfm., 2005.

39. Cutrell, E., Robbins, D., Dumais, S. and Sarin, R. Fast, flexible filtering with phlat. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, pp. 261-270, ACM Press, 2006.

40. Dai, L., Goldman, R., Sears, A. and Lozier, J. Speech-based cursor control: a study of grid-based solutions. *Proceedings of the Conference on Assistive Technologies (ASSETS '04)*, pp. 94-101, ACM Press, 2004.

41. DeLuca, E.W. and Nurnberger, A. Supporting information retrieval on mobile devices. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '05)*, pp. 347-348, 2005.

42. Department of Health and Human Services. 45 CFR Part 46, Protection of Human Subjects, 2005.

43. Dijkstra, T.M.H., Giese, M.A. and Schoner, G. *RelPhase.box*. T.M.H Dijkstra, www2.psy.ohio-state.edu/visionlab/dijkstra/Software/, 1997.

44. Doan, K., Plaisant, C. and Shneiderman, B. Query previews in networked information systems. *Proceedings of the Third Forum on Research and Technology Advances in Digital Libraries (ADL '96)*, pp. 120-129, IEEE Press, 1996.

45. Dong, L., Watters, C. and Duffy, J. Comparing two one-handed access methods on a PDA. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '05)*, pp. 235-238, ACM Press, 2005.

46. Duh, H.B.-L., Tan, G.C.B. and Chen, V.H.-h. Usability evaluation for mobile device: a comparison of laboratory and field tests. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '06)*, pp. 181-186, ACM Press, 2006.

47. Dumais, S., Cutrell, E., Cadiz, J.J., Jancke, G., Sarin, R. and Robbins, D.C. Stuff I've seen: a system for personal information retrieval and re-use. *Proceedings of the Conference on Research and Development in Informaion Retrieval (SIGIR '03)*, pp. 72-79, ACM Press, 2003.

48. Dumais, S., Cutrell, E. and Hao, C. Optimizing search by showing results in context. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '01)*, pp. 227-284, ACM Press, 2001.

49. Eslambolchilar, P. and Murray-Smith, R. Tilt-based automatic zooming and scaling in mobile devices - a state-space implementation. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '04)*, pp. 120-131, Springer, 2004.

50. Fitts, P.M. The information capacity of the human motor system in controlling amplitude of movement. *Journal of Experimental Psychology*, Vol. 47, No. 6, pp. 381-391, 1954.

51. Furnas, G.W. Generalized fisheye views. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '86)*, pp. 16-23, ACM Press, 1986.

52. Gong, J. and Tarasewich, P. Alphabetically constrained keypad designs for text entry on mobile devices. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 211-220, ACM Press, 2005.

53. Grossman, T. and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 281-290, ACM Press, 2005.

54. Guiard, Y., Blanch, R. and Beaudouin-Lafon, M. Object pointing: a complement to bitmap pointing in GUIs. *Proceedings of Graphics Interface (GI '04)*, pp. 9-16, Canadian Human-Computer Communications Society, 2004.

55. Guimbretiere, F. and Winograd, T. FlowMenu: combining command, text, and data entry. *Proceedings of the Symposium on User Interface Software and Technology (UIST '00)*, pp. 213-216, ACM Press, 2000.

56. Gutwin, C. and Fedak, C. Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. *Proceedings of Graphics*

Interface (GI '04), pp. 145-152, Canadian Human-Computer Communications Society, 2004.

57. Hachet, M., Pouderoux, J., Tyndiuk, F. and Guitton, P. "Jump and refine" for rapid pointing on mobile phones. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '07)*, pp. 167-170, ACM Press, 2007.

58. Hall, A.D., Cunningham, J.B., Roache, R.P. and Cox, J.W. Factors affecting performance using touch-entry systems: tactual recognition fields and system accuracy. *Journal of Applied Psychology*, Vol. 73, No. 4, pp. 711-720, 1988.

59. Harada, S., Landay, J.A., Malkin, J., Li, X. and Bilmes, J.A. The vocal joystick:: evaluation of voice-based cursor control techniques. *Proceedings of the Conference on Assistive Technologies (ASSETS '06)*, pp. 197-204, ACM Press, 2006.

60. Hascoët, M. Throwing models for large displays. *Proceedings of the Human-Computer Interaction Conference (HCI'2003)*, pp. 73-77, British HCI Group, 2003.

61. Hearst, M., Elliot, A., English, J., Sinha, R., Swearingen, K. and Yee, K.-P. Finding the flow in web site search. *Communications of the ACM*, Vol. 45, No. 9, pp. 42-49, 2002.

62. Himberg, J., Häkkilä, J., Kangas, P. and Mäntyjärvi, J. On-line personalization of a touchscreen based keyboard. *Proceedings of the International Conference on Intelligent User Interfaces (IUI '03)*, pp. 77-84, ACM Press, 2003.

63. Hinckley, K., Baudisch, P., Ramos, G. and Guimbretiere, F. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 451-460, ACM Press, 2005.

64. Hinckley, K., Pierce, J., Sinclair, M. and Horvitz, E. Sensing techniques for mobile interaction. *Proceedings of the Symposium on User Interface Software and Technology (UIST '00)*, pp. 91-100, ACM Press, 2000.

65. Hirotaka, N. Reassessing current cell phone designs: using thumb input effectively. *Extended Abstracts of the Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*, pp. 938-939, ACM Press, 2003.

66. Hornbæk, K. and Frøkjær, E. Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '01)*, pp. 293-300, ACM Press, 2001.

67. Huot, S. and Lecolinet, E. SpiraList: a compact visualization technique for one-handed interaction with large lists on mobile devices. *Proceedings of the 4th Nordic*

conference on Human-computer interaction: changing roles, pp. 445-448, ACM Press, 2006.

68. Hutchinson, H., Bederson, B.B. and Druin, A. Supporting elementary-age children's searching and browsing: design and evaluation using the International Children's Digital Library. *Journal of the American Society for Information Science and Technology*, Vol. 58, No. 11, pp. 1618-1630, 2007.

69. Informa Telecoms & Media. *WCIS Insight: Multiple SIMs, Quantifying the phenomenon taking mobile penetration beyond 100%*, http://www.informatm.com/, 2007.

70. Jones, M., Buchanan, G. and Thimbleby, H. Sorting out searching on small screen devices. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '02)*, pp. 81-94, Springer-Verlag, 2002.

71. Jones, S., Jones, M. and Deo, S. Using keyphrases as search result surrogates on small screen devices. *Personal and Ubiquitous Computing*, Vol. 8, No. 1, pp. 55-68, 2004.

72. Kabbash, P. and Buxton, W. The "Prince" technique: Fitts' Law and selection using area cursors. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, pp. 273-279, ACM Press, 1995.

73. Karimullah, A.S. and Sears, A. Speech-based cursor control. *Proceedings of the Conference on Assistive Technologies (ASSETS '02)*, pp. 178-185, ACM Press, 2002.

74. Karlson, A.K. and Bederson, B.B. ThumbSpace: generalized one-handed input for touchscreen-based mobile devices. *Proceedings of the International Conference on Human-computer Interaction (INTERACT '07), (in press)*, 2007.

75. Karlson, A.K., Bederson, B.B. and Contreras-Vidal, J.L. Understanding One Handed Use of Mobile Devices. in Lumsden, J. ed. *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*, Idea Group Reference, 2008 (expected).

76. Karlson, A.K., Bederson, B.B. and SanGiovanni, J. AppLens and LaunchTile: two designs for one-handed thumb use on small devices. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 201-210, ACM Press, 2005.

77. Karlson, A.K., Robertson, G., Robbins, D.C., Czerwinski, M. and Smith, G. FaThumb: A facet-based interface for mobile search. *Proceedings of the*

Conference on Human Factors in Computing Systems (CHI '06), pp. 711-720, ACM Press, 2006.

78. Kristensson, P.-O. and Zhai, S. Relaxing stylus typing precision by geometric pattern matching. *Proceedings of the International Conference on Intelligent User Interfaces (IUI '05)*, pp. 151-158, ACM Press, 2005.

79. Kristoffersen, S. and Ljungberg, F. "Making place" to make IT work: empirical explorations of HCI for mobile CSCW. *Proceedings of the International Conference on Supporting Group Work (SIGGROUP '99)*, pp. 276-285, ACM Press, 1999.

80. Kuo, L., Cooley, W., Kaufman, K., Su, F. and An, K. A kinematic method to calculate the workspace of the TMC joint. *Proceedings of the Institution of Mechanical Engineers Part H*, Vol. 218, No. 2, pp. 143-149, 2004.

81. Lam, H. and Baudisch, P. Summary thumbnails: readable overviews for small screen web browsers. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 681-690, ACM Press, 2005.

82. Laramee, R.S. and Ware, C. Rivalry and interference with a head-mounted display. *ACM Transactions on Computer Human Interaction*, Vol. 9, No. 3, pp. 238-251, 2001.

83. Larson, K. and Czerwinski, M. Web page design: implications of memory, structure and scent for info. retrieval. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '98)*, pp. 25-32, ACM Press, 1998.

84. Lenhart, A., Hitlin, P. and Madden, M. *Teens and technology: youth are leading the transition to a fully wired and mobile nation*. Pew Internet and American Life Project, http://www.pewinternet.org/PPF/r/162/report_display.asp, 2005.

85. Li, Z.-M. and Harkness, D.A. Circumferential force production of the thumb. *Medical Engineering & Physics*, Vol. 26, pp. 663-670, 2004.

86. Long, A.C., Landay, J.A., Lawrence, R.A. and Michiels, J. Visual similarity of pen gestures. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '00)*, pp. 360-367, ACM Press, 2000.

87. Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A. and Looney, E.W. Twiddler typing: one-handed chording text entry for mobile phones. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '04)*, pp. 671-678, ACM Press, 2004.

88. MacKenzie, I.S. A note on the information-theoretic basis for Fitts' law. *Journal of Motor Behavior*, Vol. 21, No. 1, pp. 323-330, 1989.

89. MacKenzie, I.S. and Buxton, W. Extending Fitts' Law to two-dimensional tasks, pp. 219-226, ACM Press, 1992.

90. MacKenzie, I.S. and Kauppinen, T. An isometric joystick as a pointing device for handheld information terminals. *Proceedings of Graphics Interface (GI '01)*, pp. 119-126, Canadian Information Processing Society, 2001.

91. MacKenzie, I.S., Kober, H., Smith, D., Jones, T. and Skepner, E. LetterWise: prefix-based disambiguation for mobile text input. in *Proceedings of the Symposium on User Interface Software and Technology (UIST '01)*, ACM Press, pp. 111-120, 2001.

92. MacKenzie, I.S. and Zhang, S.X. An empirical investigation of the novice experience with soft keyboards. *Behaviour & Information Technology*, Vol. 20, No. 6, pp. 411-418, 2001.

93. Mankoff, J. and Abowd, G.D. Cirrin: a word-level unistroke keyboard for pen input. *Proceedings of the Symposium on User Interface Software and Technology (UIST '98)*, pp. 213-214, ACM Press, 1998.

94. McAtamney, G. and Parker, C. An examination of the effects of a wearable display on informal face-to-face communication. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, pp. 45-54, ACM Press, 2006.

95. McEvoy, S.P., Stevenson, M.R., McCartt, A.T., Woodward, M., Haworth, C., Palamara, P. and Cercarelli, R. Role of mobile phones in motor vehicle crashes resulting in hospital attendance: a case-crossover study. *British Medical Journal*, Vol. 331, pp. 428-430, 2005.

96. McGookin, D.K. and Brewster, S.A. Understanding concurrent earcons: Applying auditory scene analysis principles to concurrent earcon recognition. *ACM Transactions on Applied Perception*, Vol. 1, No. 2, pp. 130-155, 2004.

97. Microsoft Inc. *Better Living through Software: Microsoft Advances for the Home Highlighted at Consumer Electronics Show 2002*, http://www.microsoft.com/presspass/features/2002/Jan02/01-08msces.mspx, Accessed on August 28, 2007, 2002.

98. Mihara, Y., Shibayama, E. and Takahashi, S. The migratory cursor: accurate speech-based cursor movement by moving multiple ghost cursors using non-verbal vocalizations. *Proceedings of the Conference on Assistive Technologies (ASSETS '05)*, pp. 76-83, ACM Press, 2005.

99. Milanesi, C., Desai, K., et al. *Market Share: Mobile Terminals, Worldwide, 4Q05 and 2005*. Gartner, Inc., 2006.

100. Milanesi, C., Liang, A., Vergne, H.J.D.L., Mitsuyama, N. and Nguyen, T.H. *Data Insight: Market Share for Mobile Devices, 1Q07*. Gartner, Inc., 2007.

101. Milanesi, C., Liang, A., Vergne, H.J.D.L., Mitsuyama, N. and Nguyen, T.H. *Market Share: Mobile Devices, Worldwide, 4Q06 and 2006*. Gartner, Inc., 2007.

102. Milic-Frayling, N., Sommerer, R., Rodden, K. and Blackwell, A. SearchMobil: web viewing and search for mobile devices. *Posters of the International World Wide Web Conference (WWW '03)*, pp. 320, ACM Press, 2003.

103. Ministry of Defense. Defense Standard 00-25, Human Factors for Designers of Systems, Part 17: Personnel Domain, 2004.

104. Minney, J. *M:Metrics: Increasing Cameraphone Ownership Forces a New Focus for Graphics Publishers*. M:Metrics. Seattle and London, http://www.mmetrics.com/press/PressRelease.aspx?article=20070417-graphics, 2007.

105. Mizobuchi, S., Chignell, M. and Newton, D. Mobile text entry: relationship between walking speed and text input task difficulty. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '05)*, pp. 122-128, ACM press, 2005.

106. Mizobuchi, S., Mori, K., Ren, X. and Yasumura, M. An empirical study of the minimum required size and the number of targets for pen on the small display. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '02)*, pp. 184-194, Springer-Verlag, 2002.

107. Mustonen, T., Olkkonen, M. and Hakkinen, J. Examining mobile phone text legibility while walking. *Extended abstracts of the Conference on Human Factors in Computing Systems (CHI '04)*, pp. 1243-1246, 2004.

108. Nacenta, M.A., Aliakseyeu, D., Subramanian, S. and Gutwin, C. A comparison of techniques for multi-display reaching. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 371-380, ACM Press, 2005.

109. Nesbat, S.B. A system for fast, full-text entry for small electronic devices. *Proceedings of the International Cnference on Multimodal Interfaces (ICMI '03)*, pp. 4-11, ACM Press, 2003.

110. Nicholson, M. and Vickers, P. Pen-based gestures: an approach to reducing screen clutter in mobile computing. *Proceedings of the International Symposium on*

Mobile Human-Computer Interaction (Mobile HCI '04), pp. 320-324, Springer-Verlag, 2004.

111. Oulasvirta, A., Tamminen, S., Roto, V. and Kuorelahti, J. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '05)*, pp. 919-928, ACM Press, 2005.

112. Parhi, P., Karlson, A.K. and Bederson, B.B. Target Size Study for One-Handed Thumb Use on Small Touchscreen Devices. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '06)*, pp. 203-210, ACM Press, 2006.

113. Pascoe, J., Ryan, N. and Mores, D. Using while moving: HCI issues in fieldwork environment. *ACM Transactions on Computer Human Interaction*, Vol. 7, No. 3, pp. 417-437, 2000.

114. Perlman, J.L., Roach, S.S. and Valero-Cuevas, J. The fundamental thumb-tip force vectors produced by muscles of the thumb. *Journal of Orthopaedic Research*, Vol. 22, pp. 306-312, 2004.

115. Pirhonen, P., Brewster, S.A. and Holguin, C. Gestural and audio metaphors as a means of control in mobile devices. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '02)*, pp. 291-298, ACM Press, 2002.

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

117. Ramos, G. and Balakrishnan, R. Fluid interaction techniques for the control and annotation of digital video. *Proceedings of the Symposium on User Interface Software and Technology (UIST '03)*, pp. 105-114, ACM Press, 2003.

118. Rao, R. and Card, S.K. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pp. 318-322, ACM Press, 1994.

119. Rath, M. and Rohs, M. Explorations in sound for tilting-based interfaces. *Proceedings of the International Conference on Multimodal interfaces*, pp. 295-301, ACM Press, 2006.

120. Rekimoto, J. Tilting operations for small screen interfaces. *Proceedings of the Symposium on User Interface Software and Technology (UIST '96)*, pp. 167-168, ACM Press, 1996.

121. Robbins, D.C., Cutrell, E., Sarin, R. and Horvitz, E. ZoneZoom: map navigation for smartphones with recursive view segmentation. *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '04)*, pp. 231-234, ACM Press, 2004.

122. Robertson, G.G. and Mackinlay, J.D. The Document Lens. *Proceedings of the Symposium on User Interface Software and Technology (UIST '93)*, pp. 101-108, ACM Press, 1993.

123. Rose, D.E. and Levinson, D. Understanding user goals in web search. *Proceedings of the International World Wide Web Conference (WWW '04)*, pp. 17-22, ACM Press, 2004.

124. Roto, V., Popescu, A., Koivisto, A. and Vartiainen, E. Minimap: a web page visualization method for mobile phones. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, pp. 35-44, ACM Press, 2006.

125. Sarkar, M. and Brown, M.H. Graphical Fisheye Views of Graphs. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '92)*, pp. 83-91, ACM Press, 1992.

126. Sawhney, N. and Schmandt, C. Nomadic radio: speech and audio interaction for contextualizing messaging in nomadic environments. *ACM Transactions on Computer Human Interaction*, Vol. 7, No. 3, pp. 353-383, 2000.

127. Sazawal, V., Want, R. and Borriello, G. The unigesture approach. *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, pp. 256-270, Springer-Verlag, 2002.

128. Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S. and Roseman, M. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction*, Vol. 3, No. 2, pp. 162-188, 1996.

129. Sears, A. Improving touchscreen keyboards: design issues and a comparison with other devices. *Interacting with Computers*, Vol. 3, No. 3, pp. 253-269, 1991.

130. Sears, A., Revis, D., Swatski, J., Crittenden, R. and Schneiderman, B. Investigating touchscreen typing: the effect of keyboard size on typing speed. *Behaviour & Information Technology*, Vol. 12, pp. 17-22, 1993.

131. Sears, A. and Shneiderman, B. High-precision touchscreens: design strategies and comparisons with a mouse. *International Journal of Man-Machine Studies*, Vol. 34, No. 4, pp. 593-613, 1991.

132. Sears, A. and Zha, Y. Data entry for mobile devices using soft keyboards: understanding the effects of keyboard size and user tasks. *International Journal of Human-Computer Interaction*, Vol. 16, No. 2, pp. 163-184, 2003.

133. Shiffler, G. *Forecast: PC Installed Base Worldwide, 2003-2011*. Gartner, Inc., 2007.

134. Shiffler, G., Kitagawa, M., Atwal, R. and Vasquez, R. *Dataquest Insight: 2Q07 Update, Global PC Scenarios, 2006-2008*. Gartner, Inc., 2007.

135. Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd ed.* Addison-Wesley, Reading, Mass, 1997.

136. Shneiderman, B. Dynamic queries for visual information seeking. *IEEE Software*, Vol. 11, No. 6, pp. 70-77, 1994.

137. Spence, R. and Apperley, M.D. An Office Environment for the Professional. *Behaviour & Information Technology*, Vol. 1, No. 1, pp. 43-54, 1982.

138. Sporka, A.J., Kurniawan, S.H., Mahmud, M. and Slav, P. Non-speech input and speech recognition for real-time control of computer games. *Proceedings of the Conference on Assistive Technologies (ASSETS '06)*, pp. 213-220, ACM Press, 2006.

139. Sporka, J., Kurniawan, H. and Slavik, P. Acoustic control of mouse pointer. *Univers. Access Inf. Soc.*, Vol. 4, No. 3, pp. 237-245, 2006.

140. Stifelman, L.J., Arons, B., Schmandt, C. and Hulteen, E.A. VoiceNotes: a speech interface for a hand-held voice notetaker. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '93)*, pp. 179-186, ACM Press, 1993.

141. Strayer, D.L., Drews, F.A. and Crouch, D.J. A comparison of the cell phone driver and the drunk driver. *Human Factors*, Vol. 48, No. 2, pp. 381-391, 2006.

142. Strayer, D.L., Drews, F.A. and Johnston, W.A. Cell phone induced failures of visual attention during simulated driving. *Journal of Experimental Psychology*, Vol. 9, No. 1, pp. 23-32, 2003.

143. System for disposing a proximity sensitive touchpad behind a mobile phone keypad. USA: Cirque Corporation, Salt Lake City, UT, US Patent 7151528, 2006.

144. Tegic Inc. *T9*, www.tegic.com, 2006.

145. Tilly, A.R. *The Measure of Man and Woman: Human Factors in Design*. Wiley, 2001.

146. Vadas, K., Patel, N., Lyons, K., Starner, t. and Jacko, J. Reading on-the-go: a comparison of audio and held-held displays. *Proceedings of the International Symposium on Mobile Human-Computer Interaction (Mobile HCI '06)*, pp. 219-226, ACM Press, 2006.

147. Vogel, D. and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '07)*, pp. 657-666, 2007.

148. Watters, C., Zhang, R. and Duffy, J. Comparing table views for small devices. *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 975-980, ACM Press, 2005.

149. Weberg, L., Torbj, Brange, r. and Hansson, s.W. A piece of butter on the PDA display. in *Extended Abstracts of the Proceedings of the Conference on Human Factors in Computing Systems (CHI '01)*, ACM Press, pp. 435-436, 2001.

150. Wigdor, D. and Balakrishnan, R. A comparison of consecutive and concurrent input entry techniques for mobile phones. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '04)*, pp. 81-88, ACM Press, 2004.

151. Wigdor, D. and Balakrishnan, R. TiltText: using tilt for text input to mobile phones. *Proceedings of the Symposium on User Interface Software and Technology (UIST '03)*, pp. 81-90, ACM Press, 2003.

152. Wigley, A., Wheelwright, S., Burbidge, R., MacLeod, R. and Sutton, M. *Microsoft .NET Compact Framework*. Microsoft Press, 2002.

153. Wobbrock, J.O., Chau, D.H. and Myers, B.A. An alternative to push, press, and tap-tap-tap: gesturing on an isometric joystick for mobile phone text entry. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '07)*, pp. 667-676, ACM Press, 2007.

154. Wobbrock, J.O., Forlizzi, J., Hudson, S.E. and Myers, B.A. WebThumb: interaction techniques for small-screen browsers. *Proceedings of the Symposium on User Interface Software and Technology (UIST '02)*, ACM Press, 2002.

155. Wobbrock, J.O., Myers, B.A. and Kembell, J.A. EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. *Proceedings of the Symposium on User Interface Software and Technology (UIST '03)*, pp. 61-70, ACM Press, 2003.

156. Wood, B., Milanesi, C., et al. *Mobile Terminal Market Share: Worldwide, 4Q03 and 2003*. Gartner, Inc., 2004.

157. Wood, B., Milanesi, C., et al. *Market Share: Mobile Terminals, Worldwide, 4Q04 and 2004*. Gartner, Inc., 2005.

158. Xie, X., Miao, G., Song, R., Wen, J.-R. and Ma, W.-Y. Efficient browsing of web search results on mobile devices based on block importance model. *Proceedings of the International Conference on Pervasive Computing*, pp. 17-26, IEEE Computer Society, 2005.

159. Yun, M.H., Han, S.H., Hong, S.W. and Kim, J. Incorporating user satisfaction into the look-and-feel of mobile phone design. *Ergonomics*, Vol. 46, No. 13/14, pp. 1423-1440, 2003.

160. Zhang, J. and Marchionini, G. Evaluation and Evlolution of a Browse and Search Interface: Relation Browser++. *Proceedings of the National Conference on Digital Government Research*, pp. 179-188, Digital Government Research Center, 2005.

161. Zhao, S., Dragicevic, P., Chignell, M., Balakrishnan, R. and Baudisch, P. Earpod: eyes-free menu selection using touch input and reactive audio feedback. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '07)*, pp. 1395-1404, ACM Press, 2007.

162. http://www.4info.net/.

163. http://www.aol.com/.

164. http://www.citysearch.com/.

165. http://www.google.com/.

166. http://www.msn.com/.

167. http://www.nokia.com/.

168. http://www.upsnap.com/.

169. http://www.vivisimo.com/.

170. http://www.yahoo.com/.

171. http://www.yp.com/.