

COSC460 Honours Report

Improved Mobile Scrolling Interfaces

November 3, 2009

Stephen Fitchett

saf75@cosc.canterbury.ac.nz

Department of Computer Science and Software Engineering
University of Canterbury, Christchurch, New Zealand

Supervisor: Professor Andy Cockburn
andy@cosc.canterbury.ac.nz

Abstract

Mobile devices are everywhere, and their capabilities are impressive: they have gigabytes of data storage, abundant applications, graphics processors, and high bandwidth wireless communications. In many ways, they are yesterday's desktop computer in your palm. And there lies the problem: their input and output devices are tiny, and interaction methods have not kept pace with the miniaturisation.

This project examines the mobile scrolling design space and proposes two new touch based scrolling interfaces. The first is a tiered alphabet scroller, which is a specialised interface designed specifically for navigating alphabetically sorted lists. An evaluation did not show a performance improvement over a standard non-tiered scroller, but users generally preferred it and we gained insights into how it is used. The second is a general purpose hybrid scrolling technique that allows for a combination of rate based scrolling for navigating short and medium distances and zero-order scrolling for navigating large distances. Evaluations showed that each component of this hybrid technique performed well for certain complementary types of tasks and that they would work together well as a hybrid technique.

Acknowledgements

The author would like to thank his supervisor, Andy Cockburn, for his valuable feedback and support throughout the project. He would also like to thank his fellow honours students for going through the experience with him, and all the participants in the experiments for their time and feedback.

Contents

1	Introduction	5
2	Taxonomy and Review of Mobile Scrolling	6
2.1	Mobility	6
2.2	Purpose of scrolling	7
2.3	Landmarks	8
2.4	Dimensions	8
2.5	Size	8
2.6	Project Scope	9
3	Design of a Tiered Alphabet Scroller	10
3.1	Initial Design	10
3.2	Transition Delays	10
3.3	Sticky Selection	12
3.4	Highlights	12
3.5	Lens View	13
3.6	Alternative Interfaces	14
4	Evaluation of the Alphabet Scroller	15
4.1	Participants and Apparatus	15
4.2	Procedure and Design	15
4.3	Results	16
4.3.1	Task Times	16
4.3.2	Tiered Scroller Usage	17
4.3.3	Errors	17
4.3.4	Preferences and Comments	19
4.4	Discussion and Future Work	20
5	Design of a General Purpose Hybrid Scrolling Technique	22
5.1	Rate Based Scrolling	22
5.1.1	Drift Zones	23
5.1.2	Edge Proximity Warnings	24
5.2	Zero-Order Scrolling	25
6	Evaluation of the General Purpose Hybrid Scrolling Technique	26
6.1	Reading Evaluation for Rate Based Scrolling	26
6.1.1	Participants and Apparatus	26
6.1.2	Procedure and Design	26
6.1.3	Results	27
6.2	Recall Evaluation for Zero-Order Scrolling	30
6.2.1	Participants and Apparatus	30
6.2.2	Task	30
6.2.3	Procedure and Design	31
6.2.4	Results	31
6.3	Discussion and Future Work	34
7	Conclusions	36
	Bibliography	36
A	iPhone Development Issues	40
A.1	View Size and Memory	40
A.2	Frame Rate Control	40
A.3	Faulty Touch Sensors	40
B	Fishing Through Data	41

List of Figures

3.1	Using an alphabet scroller to navigate a long list of words	11
3.2	Additional features of the alphabet scroller	13
4.1	The task interface for the alphabet scroller evaluation	16
4.2	Task times for the alphabet scroller evaluation	17
4.3	Number of tiers used per task	18
4.4	Errors made when using the tiered scroller in the alphabet scroller evaluation	19
4.5	Mean NASA-TLX responses for the alphabet scroller evaluation	20
5.1	Rate based scrolling while active, with drift zones displayed at the top and bottom	23
5.2	Edge proximity warnings in the drift zones	24
6.1	The task interface for the rate based scrolling evaluation	27
6.2	Task times and error rates for the reading evaluation	28
6.3	Percentage preferred walking speed (PPWS) for the reading evaluation	28
6.4	Mean NASA-TLX responses for the reading evaluation	29
6.5	The task interface for the zero-order scrolling evaluation	31
6.6	Task times by distance and target size for the recall evaluation	32
6.7	Task times by phase for the recall evaluation	33
6.8	Error rates and types for the recall evaluation	34
6.9	Mean NASA-TLX responses for the recall evaluation	35
B.1	The effect of use of sticky selection in the tiered alphabet scroller on task performance for short lists	41

1

Introduction

Document navigation in graphical user interfaces on desktop operating systems has evolved over 25 years. Scrollbars remain the dominant scrolling method, however numerous refinements enhance or complement them. Countless other scrolling and navigation techniques have also been developed.

Mobile devices present a new challenge in that some of the traditional assumptions about input and output devices are not valid. Rather than mouse and keyboard inputs, mobile devices have commonly used keypad or stylus inputs, while the Apple iPod uses a click wheel as input and more recent devices use a diverse range of different inputs. These include touch input as well as accelerometer, magnetometer, GPS, camera and microphone inputs for certain tasks or in novel interfaces. Rather than a large screen as the primary output method, mobile devices feature small screens with limited real estate. Audio feedback is also often present, as well as vibrotactile feedback on some devices. These changes have a great effect on the practicality and efficiency of scrolling methods that have become so mainstream on desktop machines. While some interfaces can be adapted for mobile devices and still work in much the same way (for example adapting scrollbars, zoom controls and overview & detail interfaces [7]), other interfaces need to be completely redesigned (for example avoiding scrolling altogether [3]).

This report establishes a foundation for scrolling analysis by presenting a taxonomy of mobile scrolling and reviewing previous work in Chapter 2. Chapter 3 describes a tiered alphabet scroller, an improvement that we designed to the alphabet scroller used to navigate sorted lists in the iPhone interface, such as contacts or music. Chapter 4 details an evaluation that compares this tiered scroller to the original. Chapter 5 describes new scrolling techniques for general purpose scrolling, which allows for both rate based scrolling and absolute positioning using zero-order scrolling. We also describe two evaluations of this technique in Chapter 6, which compare it to flick scrolling in the context of search and reading tasks. Finally, Chapter 7 summarises the findings of this project.

2 Taxonomy and Review of Mobile Scrolling

The mobile scrolling design space is large and includes a variety of factors that affect the suitability of a particular scrolling technique to a specific scenario. To identify the factors, we start by looking at a very narrow view of scrolling and then move outwards for a wider view. At the basic level, we can look at the view that is being scrolled. This gives us two factors: firstly, the *size* of the view and secondly the number of *dimensions*, that is whether scrolling will occur in one or two dimensions. If we then include the data space we gain context about the data that is being scrolled. This gives us a third factor, *landmarks*. Some contexts include landmarks which can be leveraged by scrolling techniques to help navigation. Moving outwards still, we can include the user, who has a specific *purpose of scrolling*, such as reading a passage of text or searching for a specific part of the document. Finally, if we include the surrounding environment, we can identify a final factor, *mobility*, which tells us whether the user is stationary or mobile while using their device.

The following table identifies different classifications within each factor. Interaction techniques are often suited to particular configurations for some or all of these factors. We then describe each factor in more detail and discuss some of the related work relevant to each.

Mobility	Stationary	↔	Mobile
Purpose of scrolling	Visual search	Revisitation	Analysis
Landmarks	No		Yes
Dimensions	1D		2D
Size	Small	↔	Large

2.1 Mobility

Mobile devices can be used while stationary, while moving, or somewhere in between (for example, a stationary user sitting in a moving bus). Each of these categories has factors that affect the optimal scrolling method. For example, if a user is operating the device while walking, one handed input is usually preferred. Gestures which require multiple fingers or precise manipulation, such as pinching to zoom, are often difficult to achieve with one hand since the hand must also be used to hold the device. Some techniques have been designed to facilitate one handed input, such as GraspZoom [25], which uses a pressure sensitive display to allow both scrolling and zooming actions with a single finger.

Additionally, accelerometer input is subject to greater noise while moving, affecting scrolling techniques that make use of this. This noise could potentially be reduced when movement patterns are predictable (for example, using gait analysis), however methods that use accelerometer input will always be most reliable while stationary. Indeed, we previously found that tilt scrolling is significantly more efficient than flick scrolling for analysis tasks while stationary but not while walking [16]. Interestingly, while there has been much research on tilt scrolling, it has been rarely evaluated and even more rarely evaluated while moving. Research has tended to focus on particular applications of scrolling, for example photos [5, 9], lists [18], menus [33] and maps [33]. Similarly, Mooser et al. [26] describe an interface for scrolling and controlling the zoom level of a map using a camera to detect movement rather than an accelerometer, although no mention is made of how the interface performs while walking. Finally, Eslambolchilar et al. implemented a more general purpose tilt scrolling technique by adapting speed-dependent automatic zooming for

mobile devices [14, 15]. Unfortunately, while there have been several evaluations comparing variants of tilt input (eg [32, 27]), just one of the above papers included a formal quantitative evaluation comparing it to another technique [9] and none performed an evaluation while moving.

Several other techniques have been evaluated for different levels of mobility. MacKay et al. compared scrollbar, tap-and-drag and Touch-n-Go navigation techniques for three different levels of mobility and found that they all had greater performance when stationary [24]. Brewster evaluated number entry tasks on a PDA when conducted both inside while stationary and while walking outside [6]. In both cases, no interaction was found between level of mobility and interface, however they both evaluate only interfaces that use stylus interfaces. Studies of vastly different input techniques (such as comparing touch and tilt input, as above) are more likely to result in interactions.

Other researchers have investigated how best to conduct evaluations for mobile devices that are similar to real world usage. Kjeldskov and Stage, for example, compared different techniques for evaluating the usability of mobile systems with varying degrees of mobility and distractions [22]. They found that, surprisingly, more usability problems were identified when sitting at a table than with any other technique.

2.2 Purpose of scrolling

In some contexts scrolling is usually performed for visual search tasks, that is, to search for something in an unknown location somewhere in the document. In other contexts the user simply wishes to process the whole document and will typically slowly move downwards. Interfaces are often more suited towards one of these tasks than the other. For example, interfaces that excel at making slow stable movements may be better for the latter type of task, while interfaces that excel at larger but possible jerky movements may be better for the former. A scrollbar provides examples of both scenarios. The arrow buttons at the ends can be used for small consistent scrolls suitable for reading a passage of text. Dragging the scroll thumb, however, is used for moving large distances and is less precise, making it more suitable for search tasks.

There are numerous examples of scrolling techniques aimed at search tasks. Speed-dependent automatic zooming (SDAZ) [21], for example, automatically zooms the document based on the user's scrolling velocity to keep the perceptual scrolling speed constant which in turn removes disorientation caused by motion blur. Oakley et al. developed a specialised technique for searching for contacts in an address book using tilt input [28], which, unlike most other implementations of tilt input, uses zero-order (position based) navigation as opposed to rate based scrolling. The user begins by holding down the number key on the device's keypad which corresponds to the first letter of the contact they are searching for, in order to narrow down the search space. Clearly, this method is only suitable for search tasks. Even within search tasks there are certain subsets of tasks. The Footprints scrollbar [1], for example, is aimed at revisitation tasks, which involve searching for previously acquired targets.

On mobile devices, we hypothesise that visual search tasks are less common than on the desktop when scrolling through documents, since, for example, searching through a large PDF document would be uncommon, while reading an email or website would be more typical. Unfortunately, most previous research has focused on visual search tasks and less research has been done on scrolling for analysis or reading tasks. While search tasks are typically much more time consuming than analysis tasks and therefore a more tempting research area, there is still considerable scope for research in scrolling for analysis, since most scrolling techniques interrupt the flow of reading when in use. Indeed, previous research has found that paging can be more efficient than line by line scrolling on mobile phones [29], since there are fewer interruptions to the reading task. Another study compared five different dynamic text presentation methods and found that each was suited to a particular type of screen on a mobile device [23].

Another consideration independent of the motivations of scrolling relates to interfaces which aim to support both navigation and other tasks. For example, supporting the ability to scroll, click links and select text on a web page is a difficult design challenge for most input modes. Innovative approaches to this problem include BezelSwipe, which supports text selection by using swiping gestures initiated on the device's bezel [34] and a paperweight metaphor to support editing only when the device is held securely in the palm [35].

2.3 Landmarks

Some contexts have clear landmarks which can be used to aid scrolling. One example is the alphabet scroller in the Contacts application on the iPhone and iPod touch, which contains a list of letters that can be tapped to jump to the corresponding section of a sorted list. Another is Space-Filling Thumbnails [11], which uses the pages of a document as landmarks and displays their thumbnails in a grid. Other contexts, such as emails and some webpages, do not have clear landmarks. Contexts with landmarks can often use specialised navigation techniques, such as the alphabet scroller, while those without them usually require more general techniques.

2.4 Dimensions

Using the same interface on a mobile device as on a desktop computer will almost always require two dimensional scrolling, since screen resolutions on mobile devices are much lower than on desktop computers. Two dimensional scrolling can be time consuming and frustrating with some techniques, so the smaller screen resolutions have major implications when designing interfaces for mobile devices. Some interfaces can be redesigned to accommodate a narrower view (for example, many lists). For other interfaces, such as in a web browser, a view that scrolls in only a single dimension is less feasible and both horizontal and vertical scrolling are usually required.

For various reasons, some navigation interfaces are suited for one dimensional scrolling but not two dimensional scrolling. The scroll wheel on many mice, for example, works well for one dimensional scrolling, but two dimensional versions are more difficult to use. The Footprints scrollbar [1], which aids revisitation by marking previously visited areas on the scrollbar, would not work at all in two dimensions, as the markings on a scrollbar in one dimension would not provide information about the corresponding location in the other dimension. The OrthoZoom Scroller [4] uses one dimension of input for panning and another for zooming, so would also not scale up to two dimensional scrolling. Conversely, other techniques such as rate based scrolling [38] and speed-dependent automatic zooming [21] work well in both one or two dimensions. Traditional scrollbars are commonly used for scrolling in two dimensions, although the absence of a way to scroll simultaneously in both using the scrollbar can be frustrating. On the other hand, flick scrolling, as implemented on the iPhone, provides a simple and intuitive method of scrolling simultaneously in two dimensions.

2.5 Size

Different scrolling techniques are suited to differently sized documents. In some ways, this is related to the *purpose of scrolling* in that different types of scrolling task typically involve scrolling different distances. Reading and analysis tasks involve scrolling short distances while search tasks more often involve scrolling longer distances, for example. On the other hand, document size can also have an effect independent of the distance being scrolled. For example, scrolling techniques that do not scroll relative to the current position, such as Apple's alphabet scroller and Oakley et al.'s zero-order tilt technique [28], require users to perform a sequence of gestures to locate a position which is independent of the current location. These sequences will typically be longer for larger documents since more filtering needs to be performed. Other times these techniques will just have less precision when used for long documents. Some techniques aim to reduce the cost of navigating large documents, such as by using hierarchical lists.

Size also relates to the physical input and output size. Interfaces with a lot of small components are less feasible on small screens as it becomes harder to see them and make out minor details. If a touchscreen is being used, this is especially important because it is difficult to interact with small targets.

2.6 Project Scope

Due to the limited timeframe of this project, not every part of the design space could be examined in detail. A summary of our decisions about where to focus is as follows:

Mobility	We focused on touch based scrolling techniques which did not make use of an accelerometer. However, it is still interesting to evaluate the effect of movement even between touch based techniques, so we evaluated both stationary and moving tasks in one experiment (see Section 6.1).
Purpose of scrolling	We explored analysis, visual search and revisitation tasks. Our different techniques are each suited to different tasks, and as such each was evaluated for a unique type of task.
Landmarks	We developed a tiered alphabet scroller for sorted lists containing landmarks, and generic techniques for documents without landmarks.
Dimensions	We focused on one dimensional scrolling, although our rate based and zero-order techniques would trivially scale up to two dimensions.
Size	We designed interfaces suited for long documents as well as short ones. Our evaluation of the tiered alphabet scroller features list length as a factor.

3

Design of a Tiered Alphabet Scroller

Apple use a non-tiered alphabet scroller in applications such as Contacts and Music on the iPod touch and iPhone. It lists the letters of the alphabet on the right hand side of the display and allows users to tap the letters to jump to the part of the list with items starting with that letter. This project extends the alphabet scroller concept and introduces multiple tiers of letters. The aim for this tiered interface is to improve navigation efficiency for alphabetically sorted lists by allowing users to select additional letters beyond the first to further narrow down the location. This is potentially beneficial for long lists where, for example, there might be hundreds of items beginning with 'S'. Navigating to the word 'String' in such a list would be time consuming using the non-tiered alphabet scroller, as the user must first go to the 'S' section and then scroll through a large number of items using flick scrolling before the target is visible. The tiered scroller improves performance by allowing the user to scroll directly to 'STR', for example. From this point little, if any, flick scrolling is required to reach the target.

3.1 Initial Design

The tiered interface works in a similar way to the non-tiered version. When the alphabet scroller is inactive, it looks the same in both versions. An inactive alphabet scroller is shown in Figure 3.1a. When a letter is touched, however, the program inspects the words beginning with that letter (or sequence of letters) and determines the set of letters that could follow based on items in the list. If there is more than one such letter, a new tier is displayed to the left of the current tier with the possible letters equally spaced within it. The maximum number of tiers is restricted to four. An example of an active tiered alphabet scroller is shown in Figure 3.1b. In this example, the user first touched the 'S' in the rightmost tier. On touching it, the view scrolled to the first item beginning with an 'S' and a new tier appeared to the left of the original. In the second tier, the user touched the 'T'. This displayed a third tier, with a significantly smaller set of letters, and scrolled the view to the first word beginning with 'ST'. The five letters in the third tier are the five letters that could follow 'ST' in the given list of words. This interface allows users to scroll with a precision of their choosing. They can lift their finger after making a selection in the first tier, which is functionally equivalent to using the non-tiered alphabet scroller, or they can be more precise and use multiple tiers. Users can then make decisions about how precise to be based on their target word; 'String' is more likely to benefit from multiple tiers of precision than 'Jab'.

The tiered scroller also introduces several visual changes. It features the concept of an active tier, that is, the tier that the finger is currently over. The active tier is displayed in a darker colour than the inactive tiers as an indicator of where the user is currently working. Selected letters in each tier are displayed in white to indicate the currently selected letters, which was not needed in the non-tiered version. Finally, the entire letter context is displayed in a context view above the alphabet scroller while it is active. For example, in Figure 3.1b, the letters 'ST' are displayed in the top right corner, indicating that this is the current context. This is important because the finger will often occlude currently selected letters and it provides a stable location for users to refer to.

3.2 Transition Delays

One difficulty in the development of the tiered alphabet scroller was the transition between tiers. Consider, for example, a user who wants to navigate to 'TH'. The 'T' is near the bottom of the first tier, while the 'H' is near the top of the second. The line of movement between these locations will pass over a number of other letters in the first tier, potentially changing the first letter and displaying a second tier for a different context. This is similar to

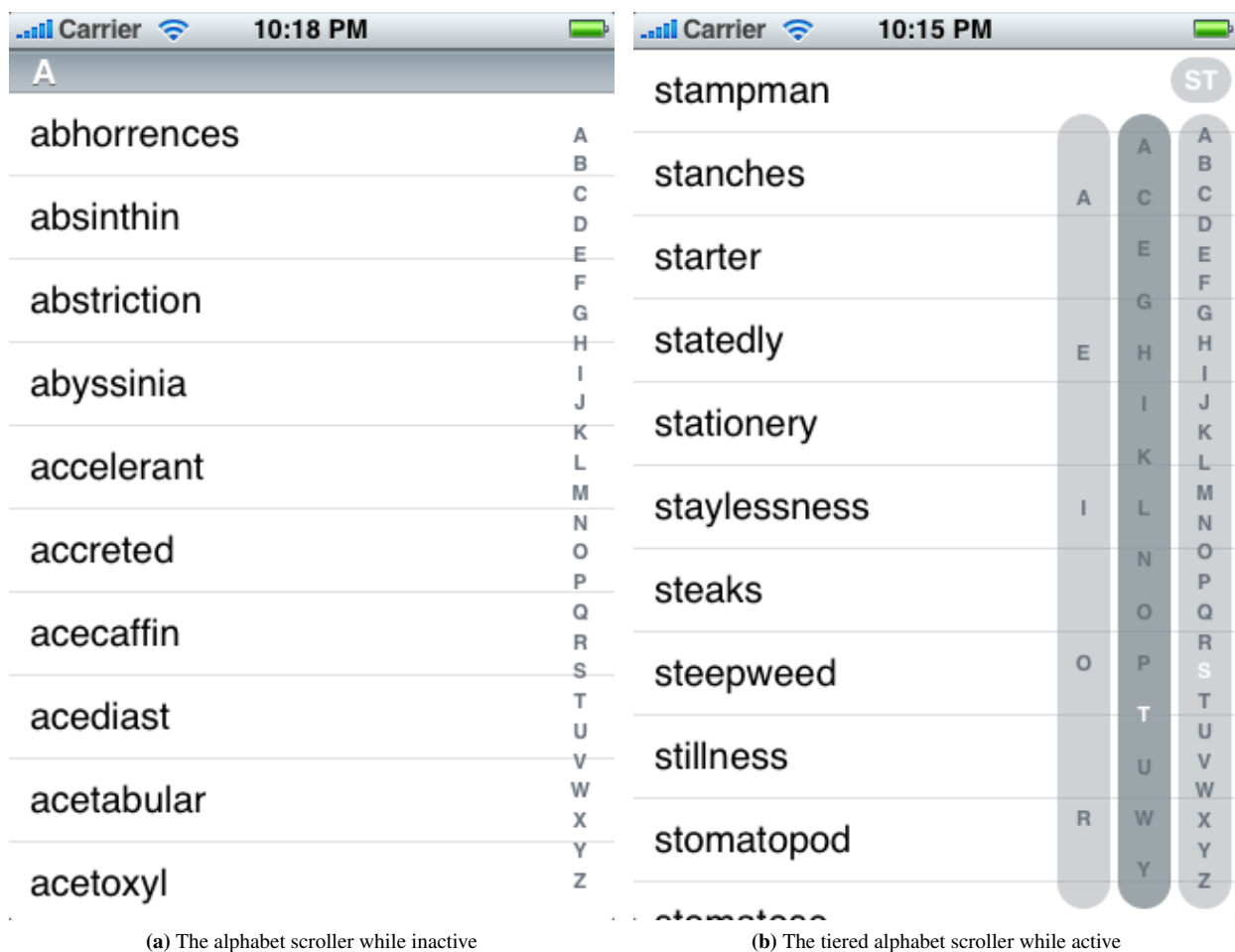


Figure 3.1: Using an alphabet scroller to navigate a long list of words

the cascading menu problem, where careful steering is required to navigate to submenu items, however the problem is exacerbated by the narrow tiers used in the alphabet scroller. Several approaches have been used to solve this problem with menus. Enlarged activation area menus [10] use activation areas for submenus in the parent menu that continue down to the next submenu or to the end of the submenu. Unfortunately this approach would not work with the alphabet scroller since every letter produces the equivalent of a submenu. Another approach is Adaptive Activation Area Menus [36], which provide enlarged triangle shaped activation areas to provide broader steering paths. Again, this would be impractical for the alphabet scroller since the tier sizes would result in very elongated triangles and it would be difficult to determine whether users were moving within a tier or moving to a child tier when moving long distances.

A common solution is to impose a short temporal delay between leaving the parent's activation area and hiding its child menu. While not ideal, since the ideal delay length will differ between user and even at different times for the same user, we used this approach as the basis for ours since it can be applied to the alphabet scroller. Rather than using a constant delay length, however, we also included ideas from Adaptive Activation Area Menus as well as aspects of Fitts' law [17] as part of a heuristic for determining a suitable delay length. The length of the delay is designed so that users can navigate to letters in child tiers without changing the selection in the parent tier, while ensuring that the interface remains relatively responsive. This delay length t , measured in seconds, is calculated using Equation (3.1):

$$t = 0.02 + \frac{0.02 \log_2 d}{n} + 0.30 \quad (3.1)$$

This equation includes three components. The first is a constant value which acts as a baseline, while the second is based partially on Fitts' law. Fitts' law tells us that the time to acquire a certain letter is roughly proportional to the log of the amplitude of the movement, given a constant speed-accuracy trade off. This means that the motion between the origin and the target has a lower velocity for close targets. Note that while the target height may vary, the width will not and will always be either similar to or smaller than the height, so we assume a constant target width in the Fitts' law model. We next assume that the movement from the origin to the target has a linear velocity (although in practice it is not), which allows us to approximate the amount of time spent travelling across a particular portion of the path. The second component in Equation (3.1) is therefore proportional to the log of the distance d moved from the previously selected context, in pixels and rounded up to a multiple of the height of the activation region for each letter, and divided by the number of letters n moved from the previously selected letter to the current point. This gives a longer delay length when close to the previously selected context, for situations when the user is selecting a nearby letter in a child tier and will be moving more slowly. The delay length is also longer when a letter is allocated a larger portion of the scroller (since n will be smaller for a given value of d), which is necessary as it takes longer to travel over larger distances.

The third component of Equation (3.1) considers the angle, θ , between the y-axis and the straight line from the point where the previous section was selected to the currently touched point. θ is set to zero if the finger has moved to the right, but is otherwise in the range $[0, \frac{\pi}{2}]$. This component uses ideas from Adaptive Activation Area Menus, but rather than using a solid triangle boundary, it assigns a higher weight to larger angles; large values of θ are a likely indicator that the user is moving to a child tier, and therefore they have a large influence on the delay length. A completely vertical movement will typically have a delay length of about 0.03 seconds for large distances to 0.1 seconds for a movement of one letter, while movements with large angles will typically have a delay length of up to 0.3 seconds.

3.3 Sticky Selection

A feature introduced in the tiered alphabet scroller is sticky selection. The non-tiered version is only ever active while a finger is pressed over the scroller. Sticky selection allows users to tap a letter without the scroller becoming inactive, where tapping is the action of quickly touching a letter and then lifting the finger from the display. They can then proceed to tap letters in other tiers. Touching the alphabet scroller for more than half a second causes it to become inactive after releasing the finger, as in the original. Alternatively, tapping on the screen away from the scroller also makes it inactive. When using sticky selection, the active tier is the tier that was last tapped.

The sticky selection behaviour is roughly analogous to menu selection in most desktop operating systems, which allow both hold-and-drag and click-and-move behaviours. Sticky selection can reduce both the physical and mental demand required to use the scroller as the user does not need to keep his or her finger pressed on the display and they do not need to guess as to what is under their finger as that portion of the display is no longer occluded.

3.4 Highlights

Once a context is selected in the alphabet scroller, the user is potentially interested in whether there are only a small number of items matching that context (in which case further narrowing down the location is not necessary) or a large number of items (in which case the user might want to make use of more tiers to narrow down the location further). One aid to making this judgement is to temporarily highlight rows which match the selected context. This technique would also improve visual search by drawing users towards those list items that they are actually interested in.

We implemented this technique so that any time the alphabet scroller context is updated, the highlighted rows change. The rows are highlighted regardless of whether the alphabet scroller is active or not, but they contain an inactivity timeout so that the rows are highlighted only when the highlights are likely to be relevant and useful. After two seconds of no scrolling events, whether they are from the alphabet scroller or using flick scrolling, the highlights begin to fade. The fading process lasts one second, after which no rows are highlighted until the next context selection using the alphabet scroller. If a flick scrolling event occurs during the fading process, the rows are fully highlighted again and the two second timer is reset. Figure 3.2a shows fully highlighted rows after selecting the context 'STR' in the alphabet scroller. Note that in the figure, the alphabet scroller is inactive, but it would have been active when the rows were originally highlighted.

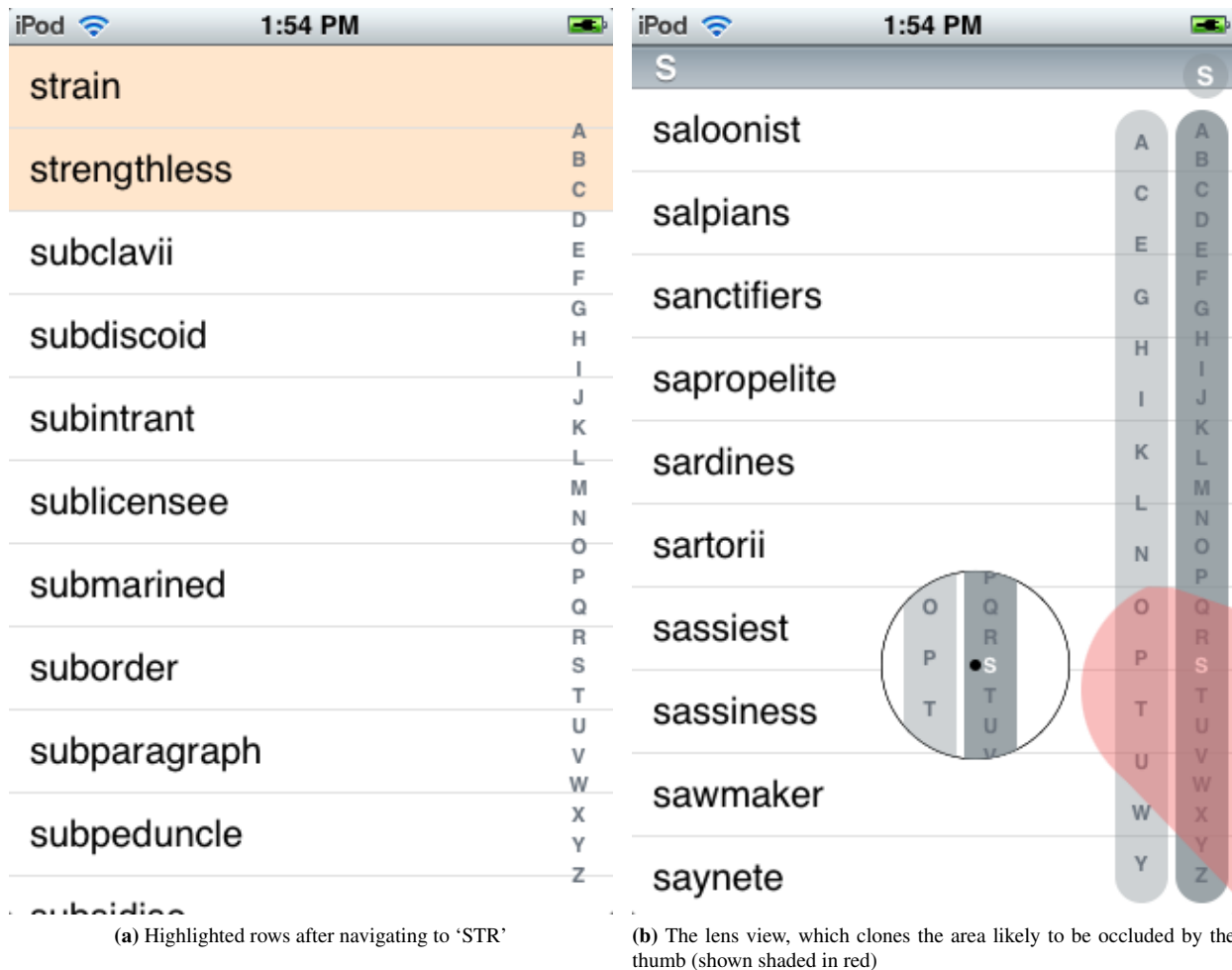


Figure 3.2: Additional features of the alphabet scroller

3.5 Lens View

While the context view and sticky selection partially solve the occlusion problem caused by placing a finger or thumb over the alphabet scroller, it remains an issue when searching for occluded characters in the alphabet scroller when not using sticky selection. For example in Figure 3.1b, if the thumb is directly over the 'S' in the first tier then surrounding letters will likely be occluded, including the 'P' and 'T' in the second tier. This makes it difficult for the user to navigate to the 'T' in the second tier, for example, as they can not be certain exactly where it is.

To solve this problem we draw a circular lens view [13] offset to the left of the thumb position which appears whenever a finger or thumb is held on the alphabet scroller. There is a 0.15 second delay before the view is displayed to prevent it from quickly appearing and disappearing when using sticky selection. The lens view clones the area under the thumb without magnification and draws it in a location that will not be occluded. It also draws a small black dot in its centre to indicate the exact point that is recognised as the touch location; this is useful for making precise selections. The lens view is shown in Figure 3.2b.

3.6 Alternative Interfaces

An alternative to the alphabet scroller is to use direct keyboard entry. This is done, for example, in the search field of Apple's Contacts application. Previous research has shown that keyword-based search is underused, while an "orienting" strategy is more common [37]. A major advantage of the alphabet scroller is that it does not reduce the screen real estate by displaying an onscreen keyboard; a list using the alphabet scroller has almost three times the vertical space than one with a search field and an active keyboard viewer. This makes it easier to understand the current context and refine the selection. Additionally, the alphabet scroller filters selection of letters to only the possible letters for the current context, while keyboard entry allows entry of invalid combinations of letters which do not appear in the list. Finally, the alphabet scroller can be used to navigate to a rough area of a list, while the search feature in the Contacts application just filters the list to the relevant items. These differences make the alphabet scroller easier to use for "orienting" strategies, which have been found to be more commonly used.

4 Evaluation of the Alphabet Scroller

We ran an evaluation comparing the standard alphabet scroller to the tiered alphabet scroller for search tasks.

4.1 Participants and Apparatus

16 computer science students (four female) with a mean age of 25 participated in the evaluation. Six had had previous experience with an iPod touch or iPhone. Participants were given a \$10 shopping voucher for participating in this and one or more of the other two evaluations.

The evaluation was performed on a second generation iPod touch running iPhone OS 2.2.1. The display's resolution was 480×320 pixels and it was always oriented in portrait.

4.2 Procedure and Design

The evaluation was analysed as a 2×2 repeated measures analyses of variance (ANOVA) for factors *interface* (standard and tiered alphabet scroller) and *list length* (short and long), with task time as a dependent variable. Short lists contained 100 items and long lists contained 1000 items. All factors were counterbalanced.

At the beginning of the experiment participants were asked to provide basic demographic information. They were then given a demonstration of the features of one of the alphabet scroller interfaces. Next, they were asked to complete five practice tasks for the interface, using lists of length 320. This length was chosen as it is close to the geometric mean of the list lengths of the short and long lists used in the real tasks.

Tasks consisted of a search word permanently displayed at the top of the screen in the location the status bar normally appears. The list took up the remainder of the screen (456 pixels). The search word was randomly selected from the list of words, with no duplicates across all trials for a participant. Additionally, words that required no scrolling were excluded from the candidate words. The list was scrolled to the top at the start of each trial. The task interface is shown in Figure 4.1.

Task time was the time from the appearance of the task to the time that the correct item was tapped in the list. While evaluations for desktop interfaces often use the time of the first mouse movement for the task start time, this would correspond to a participant moving their finger with an iPod touch, which does not trigger an event until they touch the display. We felt it was more important to include the time spent moving the finger than excluding the time spent preparing to start a task.

After completing practice tasks for an interface, participants began tasks for short and long lists, which had the same appearance and behaviour as the practice tasks with the exception of list length. Each condition had ten trials, the first two of which were not included in statistical analysis, although participants were not aware of this exclusion. On completion of the tasks for the first interface, participants completed NASA Task Load Index (TLX) worksheets [19]. This sequence was then repeated for the second interface. In total, each participant completed 25 tasks for each interface (16 recorded for statistical analysis).

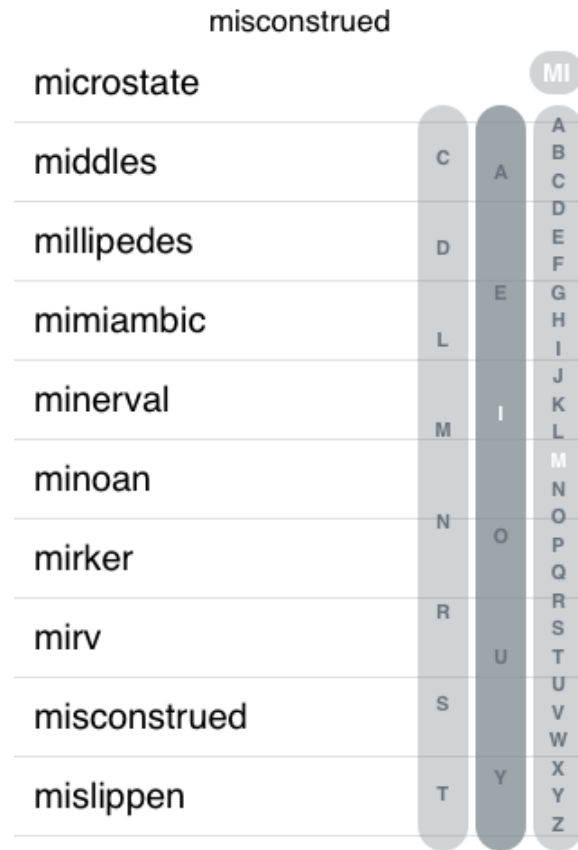


Figure 4.1: The task interface for the alphabet scroller evaluation

4.3 Results

The following sections discuss the results for task times as well as insights gained about how the tiered scroller is used, analysis of the types of errors made by participants and participants' preferences and comments.

4.3.1 Task Times

Surprisingly, task times were slightly faster for the non-tiered scroller for both short and long lists, although the difference was not significant ($F_{1,15} = 2.204, p = 0.158$). There was substantial variation between participants, with participants ranging from a 30% improvement when using the tiered scroller to a 40% decrease in performance. It was clear that the main issue with both versions of the scroller was that it was difficult to accurately make selections since the targets were small, but for the tiered scroller the cost of making an error was larger. For example, participants would often accidentally make a selection on the tier to the right of the one they intended to use, resulting in having to reselect the correct letters on two tiers. Other times, if they were using sticky selection, they would accidentally tap to the left of the leftmost tier, resulting in the scroller becoming inactive. Although in theory the tiered scroller could be used in the same way as the non-tiered scroller, participants were inclined to try using multiple tiers when available; errors made when attempting to select items on the additional tiers is what caused many participants to perform worse than when using the non-tiered scroller.

There was a significant main effect for *list length* ($F_{1,15} = 117.643, p < 0.001$), with tasks for short lists (mean: 4.40 seconds) taking less time than for long lists (mean: 9.15 seconds). There were no significant interactions. Task times are summarised in Figure 4.2.

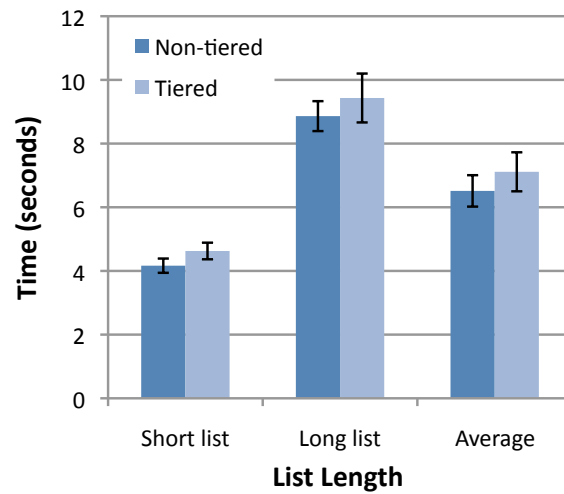


Figure 4.2: Task times for the alphabet scroller evaluation. Error bars show standard error.

4.3.2 Tiered Scroller Usage

When demonstrated the tiered scroller to participants, both sticky selection and dragging were explained, but participants were not prompted to use a particular method. The experiment logs included data about how much participants used each. The initial touch on the scroller was regarded as neither type of selection, while subsequent context changes before the scroller was hidden were classified as one or the other based on whether the participant had released their finger since the last context change. Eight participants used sticky selection exclusively, while one dragged across the scroller exclusively. Amongst the remainder, four used sticky selection more than dragging and three dragged more than using sticky selection. There was no clear relationship between usage of sticky selection and task performance.

Additionally, we recorded the number of tiers of the tiered scroller that were used for each task. This was defined as the total number of tiers which were actually interacted with for each task, that is, selecting a letter in the tier rather than the tier just being visible. There was a significant main effect for *list length* ($F_{1,15} = 68.713, p < 0.001$), with an average of 1.5 tiers used for short lists and 2.4 tiers used for long lists. This is shown in Figure 4.3.

4.3.3 Errors

While it was obvious during the experiment that many errors were being made with the tiered alphabet scroller, the types of these errors was less clear. We conducted a post hoc analysis to investigate this. Firstly, we devised classifications for three different types of errors that occurred when using the tiered scroller, each of which had a slightly different cause depending on the mode of interaction. When using sticky selection, the errors were classified as follows:

Scroller dismissal These occurred when participants hit to the left of a target and accidentally dismissed the scroller (horizontal imprecision). These were the costliest errors as participants had to start their selection from the beginning. They were detected in the logs when the scroller was dismissed and then later reactivated for the same task.

Post-selection errors These occurred when participants selected a correct letter in a tier then later selected an incorrect letter in the same tier. This would typically be a result of aiming for a target in a child tier and hitting too far to the right (horizontal imprecision), and were reasonably costly errors.

Pre-selection errors These occurred when participants selected an incorrect letter in a tier when the tier did not already have the correct letter selected. This would typically be a result of aiming for the correct letter but hitting too far up or down (vertical imprecision), and was the least costly error for sticky selection.

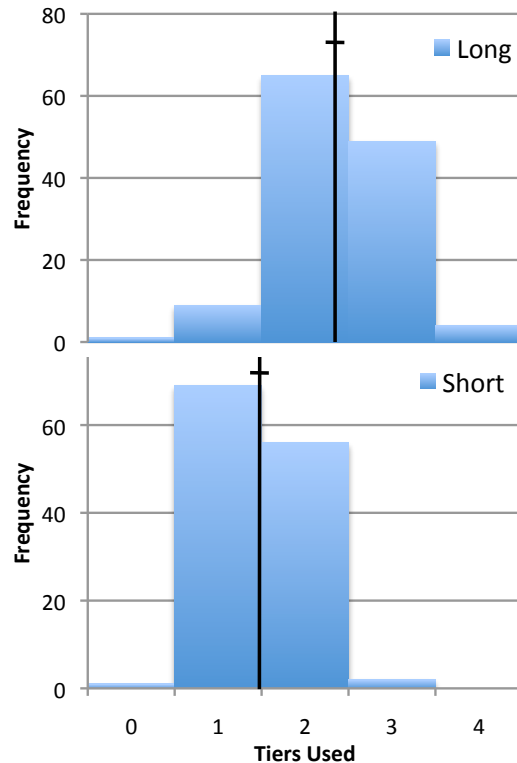


Figure 4.3: Number of tiers used per task when using the tiered scroller for short (bottom) and long lists (top). The vertical line on each shows the mean; the horizontal lines crossing them show the standard error of that mean.

When dragging, the errors were classified as follows:

Scroller dismissal These occurred when participants released their finger then later reactivated the scroller. Possible reasons for this include not realising that releasing the finger would hide the scroller or being in such an erroneous state due to other errors that the participant would decide to start again.

Post-selection errors These occurred when participants selected a correct letter in a tier then later selected an incorrect letter in the same tier. This was typically a result of either not moving sufficiently to the left when moving to a child tier (horizontal imprecision) or unintended movement after making an initial correct selection (vertical imprecision).

Pre-selection errors These occurred when participants selected an incorrect letter in a tier when the tier did not already have the correct letter selected. When dragging, this would typically be a result of letters being selected while the finger passed between the previously selected letter in a parent tier and the target letter in a child tier, and as such are not technically errors. Other times it would be legitimate errors when the user did not acquire the target correctly (vertical imprecision).

Figure 4.4a shows the number of occurrences of each type of error. By far the most common type of error was *pre-selection* errors when dragging, although as explained about most of these are not technically errors, and 78% of them were made by just two participants. *Post-selection* errors were more common than *scroller dismissal* errors when using sticky selection. This indicates that participants had a bias towards hitting slightly to the right of their intended target, since both types of errors corresponding to horizontal imprecision but in different directions. This is logical since their fingers all approached the display from the right, and perhaps with practice users would learn to compensate for this and reduce their error rate.

If we consider *scroller dismissal* and *post-selection* errors as major errors, since they both result in having to redo previous work, there were an average of 5.6 major errors per participant, or 0.35 per task (there were 16 tasks using the tiered scroller included in the analysis for each participant). This is a large error rate which shows why the tiered scroller did not perform as well as expected. In contrast, almost all errors using the non-tiered scroller would be classified as *pre-selection* errors, which are not very costly. While *post-selection* errors would be possible in a non-tiered scroller, they would be rare and less costly than in the tiered scroller.

Because usage patterns also differed between participants, we also analysed how many participants encountered one or more of each type of error. These counts are shown in Figure 4.4b. Almost every participant who used sticky selection made *post-selection* errors, but all three types were made by a majority of such participants. Of the seven participants who used dragging, just three made *scroller dismissal* errors, while *post-selection* and *pre-selection* errors were both made by most such participants. Participants varied wildly in their error rates, with the number of all errors made varying from a low of one for two participants to a high of 53 for another. Excluding *pre-selection* errors, error counts ranged from zero to 16.

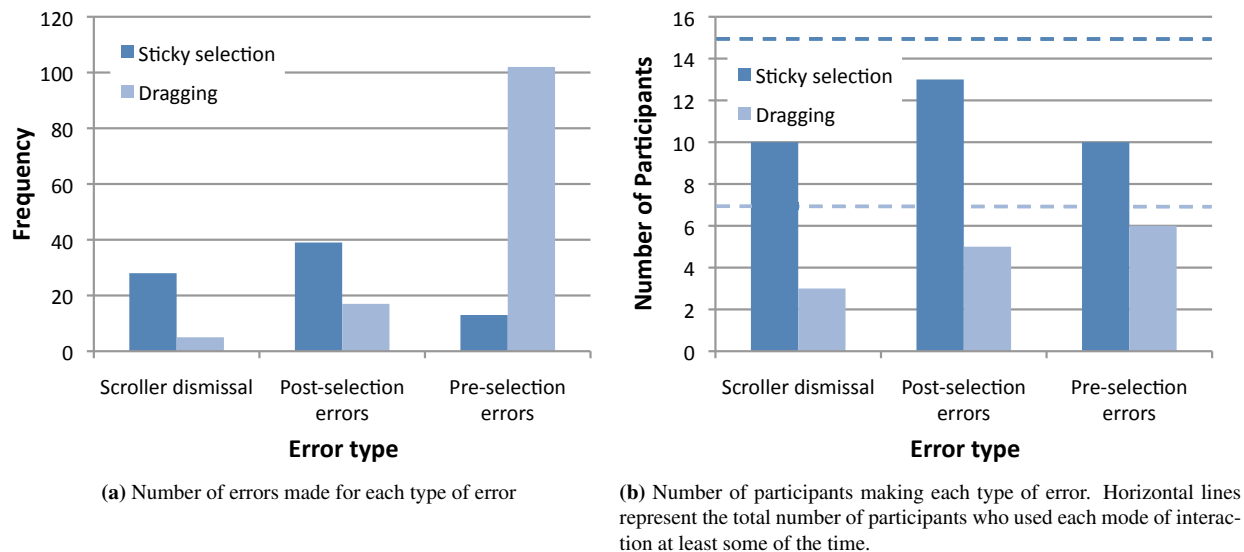


Figure 4.4: Errors made when using the tiered scroller in the alphabet scroller evaluation

4.3.4 Preferences and Comments

For short lists, five of the 16 participants preferred the tiered scroller while 11 preferred the non-tiered scroller. This rose to 13 participants preferring the tiered scroller for long lists and two preferring the non-tiered scroller. Overall, 13 participants preferred the tiered scroller and three preferred the non-tiered scroller. Participants were also asked which was more enjoyable to use; enjoyability is often cited as the main reason for the success of flick scrolling, for example. On this measure, 10 chose the tiered scroller and five chose the non-tiered scroller. Finally, 11 participants agreed that they would like the tiered scroller to be used for sorted lists on the iPod touch and iPhone, while the remaining five were neutral about the idea.

Participants were also asked which features of the tiered scroller they found useful. 10 found row highlighting useful, including one who was especially enthusiastic about it, commenting that they would love for it to be added as a feature for the non-tiered scroller as well. Six reported finding the lens view useful, although one of these participants used sticky selection exclusively during timed tasks so likely would not have encountered the lens view except on practice tasks. One participant also commented that the lens view would be useful for the non-tiered scroller. 14 reported finding sticky selection useful. This corroborated the previously described usage data, with just the two participants who used sticky selection the least not describing it as useful. Finally, just two participants found the context view at the top of the scroller useful, some commenting that they didn't even notice it.

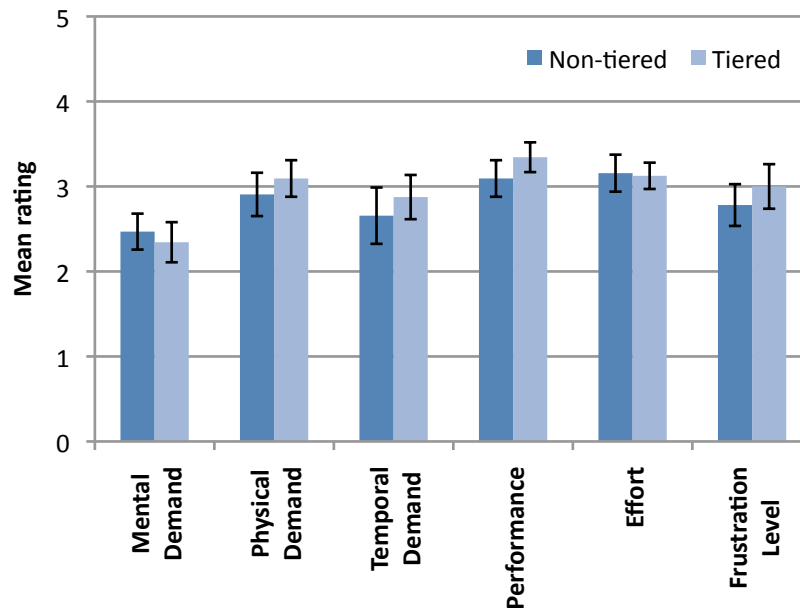


Figure 4.5: Mean NASA-TLX responses for the alphabet scroller evaluation. Lower numbers are better except for performance. Errors bars show standard error.

Almost every participant commented on difficulties with correctly selecting letters. Many had problems with both scrollers, although the cost of errors was greater for the tiered scroller as it would often select letters in the wrong tier. Several participants also commented that they would like it if the scroller would not hide itself after making a selection using dragging, so that they could use a combination of dragging and sticky selection.

Several participants made comments about the general design of the tiered scroller. One commented that rather than having letters equally spaced in each tier, they would like it if they were lined up with the letters in the other tiers, for example having ‘e’ at the same vertical location in every tier. Another commented that they were sometimes confused about which tier they were working with when a new tier appeared on the left, despite the distinct appearance of the active tier. In terms of using the tiered scroller, one participant commented that it was difficult to choose whether to use an additional tier or to start using flick scrolling.

The mean NASA-TLX responses were similar for the non-tiered scroller and the tiered scroller on all measures, with no significant differences. A summary of responses is shown in Figure 4.5.

4.4 Discussion and Future Work

The tiered alphabet scroller was preferred by a majority of participants but did not perform any better than the non-tiered scroller. While the tiered scroller has the potential to improve target acquisition, especially in long lists, it is clear from the evaluation the the current implementation suffers from flaws that limit its performance. The small target sizes result in a large number of errors and the number of these errors varies greatly between participants, with some performing better with the tiered scroller and some performing much worse. Changes need to be made to both reduce the number of errors and the cost of such errors when they do occur.

For the former problem, one possibility to reduce errors includes variable width tiers, for example making the active tier and its child tier wider to increase the target size and reduce the frequency of horizontal errors (which have the highest cost). Additionally, the context view at the top of the scroller was rarely used by participants and many did not even notice it. It could be removed to give more vertical space to the scroller, slightly increasing vertical target sizes. Increasing target sizes is especially important when a thumb is used rather than a finger, since it has a larger surface

error and is less precise on a touch screen. A thumb is ideal, however, when controlling the device one handed.

For the latter problem, there are several changes that could be made to assist in error correction. The alphabet scroller could be kept visible after drag selections and only hidden when tapping away from the scroller. This would allow users to alter their original selection when dragging rather than starting again from the first tier. Additionally, functionality could be added to revert to a previous context.

In theory, the tiered scroller can be used in the same way as a non-tiered scroller, so if users follow an optimal usage pattern they should perform at worst the same as with the non-tiered scroller. In practice, this was not the case, and provides a clear example of the advantages of simplicity. Offering multiple ways of achieving a task adds extra mental demand for users and they will often not choose the most efficient way. Future iterations of the alphabet scroller could take advantage of this insight, for example by providing visual cues indicating whether it is likely to be more efficient for users to use an additional tier. If only five rows are matched when making a selection, the next tier could be semi-transparent, for example, since it would be inefficient to use it.

While it was not a dominant issue in the evaluation, future work could also investigate the best way to space letters rather than distributing them evenly. One possibility is to allocate space for each letter in the scroller based on the frequency of the context. For example, if there is one word beginning with 'THR' and two words beginning with 'THE', the 'E' in the third tier would be allocated more space than the 'R'. This would increase the target size for more common targets and provide a visual indicator as to which contexts are more common. Additionally, if the number of items in a context is proportional to its target size, the position of each letter in the first tier would provide an indicator as to where in the list each letter is placed. A proportional mapping would not likely be optimal for target acquisition, however, since Fitts' law states that task time has a logarithmic rather than linear relationship to target size [17].

This approach is similar to morphing menus [12] which assign a larger target size to menu items that are more commonly selected. Like morphing menus, target sizes could be adjusted over time based on selection history. Additionally, the count of items in each context could be used to calculate the initial sizes rather than using the same initial size for all items as in morphing menus.

Unfortunately such a feature would be difficult to accomplish in practice except when there are relatively few letters in a tier. When the letter density of a tier is high, a large proportion of its vertical space is used to display the letters and there is little white space between them. The font size could not be decreased much as the display is small and doing so would make the letters difficult to read. There would therefore not be much flexibility to position letters. Another practical issue if the letters are assigned different proportions of a tier is visually distinguishing the activation areas of each letter. Currently letters can potentially have large vertical activation errors but will only have a visual indication in their centre where the letter is written, with an implied barrier at the centre of the gap between two consecutive letters. Variable letter proportions would result in barriers that are not at the centre of these gaps, making it less clear where the barriers are without visual indications. It was also apparent during the evaluation that several participants did not realise the activation areas extended significantly beyond the letters themselves in cases where there were few letters in a tier.

An alternative approach to letter spacing would be to do as one evaluation participant suggested and position letters in a child tier at the same location as in their parent tiers. This would have the disadvantage of occasionally giving smaller than average targets for common letters, however would provide a level of spacial consistency that may aid users in quickly locating targets within the alphabet scroller.

Even if the tiered scroller is not adapted outside this project, either in its current form or in a modified form based on the changes suggested above, some of its features would be worthy of adding to the non-tiered scroller. Notably, both row highlighting and the lens view were found to be useful by some participants and were not intrusive for the others.

5

Design of a General Purpose Hybrid Scrolling Technique

While the alphabet scroller provides a specialised scrolling interface for alphabetically sorted lists, flick scrolling is a general scrolling technique used throughout the iPhone and iPod touch interface. It can be used in one of two ways. Firstly, touching the display and moving the finger or thumb *without* releasing it will scroll the view by the distance moved in the opposite direction. In other words, touching a particular part of the view and moving the finger will scroll the view such that the same part of the view stays under the finger. Secondly, *flicking* a finger across the display, that is, touching the display only very briefly while moving the finger, causes the view to continue scrolling once the finger is released with an initial velocity calculated based on the speed of the flick movement. This scrolling velocity then decreases over time until coming to rest after about one to three seconds.

While the above implementation is the most widely used, there are other versions of flick scrolling. Aliakseyeu et al. describe and evaluate four different versions, including Apple's implementation [2]. Notably, the three other implementations they describe do not feature a friction factor which decreases the scrolling velocity over time for flick actions. A separate action, tapping the screen, is required to stop scrolling.

Unfortunately these implementations are all slow or cumbersome to use when navigating medium to long distances. With Apple's implementation, navigating a long distance requires repeated flicking actions and can be time consuming and strenuous. Other implementations described by Aliakseyeu et al. improve this by requiring only an initial flick action, after which the user only needs to wait and then tap the screen when reaching the target. While these implementations reduce the physical demand involved to scroll, it can still be very time consuming to scroll large distances. Additionally, it is difficult to control the scrolling speed using these methods; for example, users may wish to reduce the scrolling speed as they approach their destination so as not to overshoot their target. It is also slightly counterintuitive that an action must be performed to stop scrolling, rather than *ending* an action to stop scrolling.

We describe below a new scrolling technique which reduces physical demand as in Aliakseyeu et al.'s techniques, but also allows for fine control of scrolling speed throughout a scroll operation using rate based (first-order) scrolling. We also take advantage of multi-touch to support a zero-order scrolling mechanism for rapidly scrolling large distances. Combined, these enhancements allow for quick navigation with low physical demand across both short and long distances.

5.1 Rate Based Scrolling

In Apple's flick scrolling implementation, dragging a finger on the screen and then keeping it pressed will result in an initial movement only. We describe a rate based (first-order) scrolling technique to continue scrolling as long as the finger touches the display. Rate based scrolling has previously been implemented on the desktop (eg [38]) and is supported in many applications such as some web browsers. Our implementation works by first recording the location of the initial touch. We then continually update the scrolling speed based on the distance between the current touch location and the initial touch location. Equation (5.1) gives the scrolling speed calculation.

$$s = (|p_1 - p_0|)^{1.3} \quad (5.1)$$

Here, s is the scrolling speed in pixels per second, p_1 is the current touch location and p_0 is the initial touch location. The view is scrolled in the opposite direction to Apple's flick scrolling implementation; moving the finger down scrolls the view down. An exponent of 1.3 was devised empirically to allow for both fine control with small distances and



Figure 5.1: Rate based scrolling while active, with drift zones displayed at the top and bottom

high scrolling speeds with large distances. In our prototype we supported only one dimensional scrolling, in effect ignoring the x components of p_0 and p_1 , however this interface would easily scale to two dimensions.

5.1.1 Drift Zones

A difficulty with this implementation is that if the initial touch location is close to the edge of the view, the scrolling speed is limited to a low speed in one dimension. For example if the user touches the display near the top of the view and wishes to scroll upwards, they can only move their finger up a small amount whilst keeping it on the display. To solve this, we introduce a new concept called drift zones. Drift zones appear at the top and bottom of the view and are displayed as subtle translucent rectangles, shown in Figure 5.1. When the current touch location is within a drift zone, the initial touch location is gradually moved away from the touched end of the view, up to a maximum of 25% of the view's height away from the opposite end of the view. The speed at which the initial touch location moves is based on how long the finger has been in the drift zone and increases over time. It is calculated using Equation (5.2), where t is the time in seconds since entering the drift zone and s is the speed at which the initial touch location is moving, in pixels per second.

$$s = 25 + 10t \quad (5.2)$$

With a full screen view (with the exception of the a 20 pixel high status bar at the top of the screen), this technique allows for a distance between the initial and current touch locations of up to 345 pixels regardless of where the display was initially touched. This distance corresponds to a scrolling speed of approximately 2000 pixels per second using Equation (5.1).

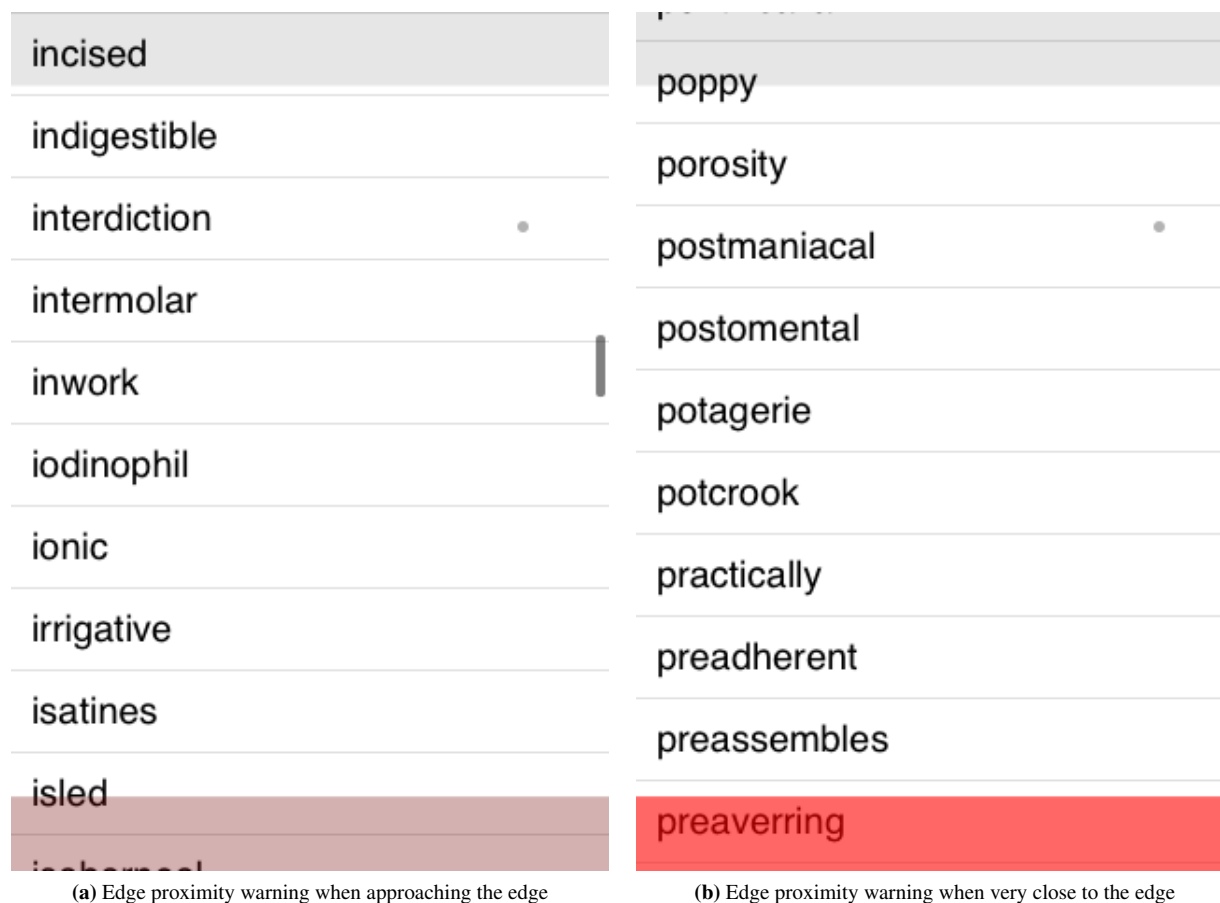


Figure 5.2: Edge proximity warnings in the drift zones

5.1.2 Edge Proximity Warnings

The iPhone OS does not distinguish being a finger being lifted off the device’s display and the finger sliding off the edge of the display onto the bezel; both are interpreted as *touch ended* events. Similarly, sliding the finger from the bezel onto the display is interpreted as a *touch began* event. This is problematic due to the use of drift zones at the edges of the display: sliding the finger too far during a scroll operation will result in a *touch ended* event that stops the scrolling. Sliding the finger back onto the display does not rectify the problem as the initial touch location will not be in its original location. While other researchers have attempted to detect swipes onto the bezel (eg Bezel Swipe [34]), these techniques will never be completely accurate without additional hardware.

We attempted to partially prevent this problem by implementing proximity warnings when the touch location is close to the edge of the display during a scroll operation. When the finger is in the outer half of a drift zone (that is, the 20 pixels adjacent to the top and bottom of the display), the drift zones are shaded red to indicate imminent ‘danger’. The shading is graduated so that the closer to the edge the finger is, the redder the shading. A value p representing the proximity to the edge is calculated using Equation 5.3, where y is the distance, in pixels, from the edge of the display. The proximity value, which is between zero (20 or more pixels from the edge) and one (right on the edge), is then used in Equation 5.4 to calculate the drift zone colour. Note that we are representing colours in the form {red, green, blue, alpha}. The edge proximity warning can be seen in Figure 5.2.

$$p = \left(1 - \min\left(\frac{y}{20}, 1\right)\right)^2 \quad (5.3)$$

$$c = \left\{p, 0, 0, 0.05 + \frac{p}{2}\right\} \quad (5.4)$$

5.2 Zero-Order Scrolling

While the rate based scrolling design described above is useful for scrolling short to medium distances, it is slow for navigating long distances in large views. Zero-order control maps the finger position directly to the view position; as the finger position moves, the view position changes proportionally. While Apple's flick scrolling implementation uses zero-order control when dragging a finger, the control-display gain is such that a single gesture can scroll the view by a maximum of the height of the view. We made four key changes. Firstly, while flick scrolling considers only finger movements while the display is touched, our approach also considers movements when the display is not touched. Secondly, we used a different control-display gain such that the user can scroll to any point in the document with a single gesture. Taken together, these first two changes mean that touching $y\%$ down the display scrolls $y\%$ down the view. Thirdly, this mode of scrolling is activated only when touching the display with two fingers, so as not to interfere with rate based scrolling. Finally, we made the scroll indicator always visible rather than just visible while scrolling as with flick scrolling. When scrolling gestures do not cause scrolling relative to the current location, it becomes more important to have a visual indicator for the current scroll position when initiating a scrolling operation.

Typical usage of zero-order scrolling involves touching the display in the approximate location of the target, then dragging to refine the location. Once activated, zero-order scrolling continues until all fingers are released; this makes it easier to scroll to the end of a view, for example, since two fingers will not generally be at exactly the same y positions and one will leave the display and trigger a *touch ended* event before the other. An undesirable consequence of this is that when releasing both fingers at a similar time, the view may scroll to the location of the last finger to be released. To avoid this problem, we implemented a 0.1 second delay before scrolling. If all fingers are released in this time, the scrolling is cancelled. This change means that the view will not generally scroll while releasing fingers.

6

Evaluation of the General Purpose Hybrid Scrolling Technique

Our generic touch scrolling method described in Chapter 5 contains two techniques within it: rate based scrolling, primarily aimed at navigating short to medium distances, and zero-order scrolling, aimed at navigating long distances. To evaluate our method we teased these two techniques apart to examine them independently and gain more insightful results. We ensured that their usage was similar to if they were used together, however zero-order scrolling was activated with a single finger for the evaluation to reduce confusion since one finger input would have otherwise done nothing and because of some minor device dependent precision issues when using multitouch. The evaluations for each technique used tasks that would require appropriate scrolling distances for the interface.

6.1 Reading Evaluation for Rate Based Scrolling

Reading tasks are a major type of task when scrolling. An example would be reading an email, where the entire text is read and the view is slowly scrolled down. Scrolling techniques which do not impede reading have an advantage, while techniques that allow for rapidly navigating large distances are less useful. Reading tasks may be particularly common on mobile devices relative to search tasks since long documents are not as common as on the desktop. Additionally, as reading tasks involve scrolling short distances, they are well suited for evaluating rate based scrolling.

We compared our rate based scrolling technique to flick scrolling for reading tasks using the same methodology that we previously developed to evaluate tilt scrolling [16]. The task interface is shown in Figure 6.1.

6.1.1 Participants and Apparatus

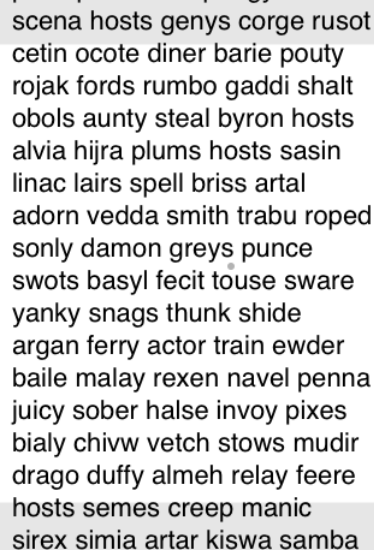
20 computer science students (four female) with a mean age of 25 participated in the evaluation. Eight had had previous experience with an iPod touch or iPhone. Participants were given a \$10 shopping voucher for participating in this and zero or more of the other two evaluations.

The evaluation was performed on a second generation iPod touch running iPhone OS 2.2.1. The display's resolution was 480×320 pixels and it was always oriented in portrait.

6.1.2 Procedure and Design

The evaluation was analysed as a 2×2 repeated measures analyses of variance (ANOVA) for factors *interface* (rate based and flick scrolling) and *movement type* (stationary and moving). Both factors were counterbalanced. Dependent variables were task times and error rates. Percentage preferred walking speed (PPWS) [30, 31] was also analysed for moving tasks.

Our original methodology used two task types: one consisted of text tasks where participants were asked to count the number of occurrences in a word, while the other consisted of a grid of differently coloured dots where participants were asked to count the number of rows with an even number of black dots. In the original experiment, text task times and error rates both had larger variation than for the counting task, while the difference between task times and error rates for each interface was about the same for each task type. This indicates that little information would be lost by focusing on a single task type, and it provides an opportunity to increase the number of trials per condition given the same time constraints, resulting in greater statistical power. Since the text task more accurately represents real world



scena hosts genys corge rusot
 cetin ocote diner barie pouty
 rojak fords rumbo gaddi shalt
 obols aunty steal byron hosts
 alvia hijra plums hosts sasin
 linac lairs spell briss artal
 adorn vedda smith trabu roped
 sonly damon greys punce
 swots basyl fecit touse sware
 yanky snags thunk shide
 argan ferry actor train ewder
 baile malay rexen navel penna
 juicy sober halse invoy pixes
 bialy chivw vetch stows mudir
 drago duffy almeh relay feere
 hosts semes creep manic
 sirex simia artar kiswa samba

Figure 6.1: The task interface for the rate based scrolling evaluation

tasks, we decided to evaluate only those tasks. We increased the number of trials for each condition from three in the original experiment (including one practice task) to five in this experiment (again, with one practice task).

Otherwise, the methodology used for this evaluation was mostly unchanged. Participants were given a brief introduction to the experiment before they carried out a preferred walking speed calibration to determine their normal walking speed. Next, they were shown an example text task. They were then given a brief demonstration of one interface and were then given 30 seconds to practice using it on a 2000 pixel high view. After this they began the actual tasks, filling out NASA Task Load Index (TLX) worksheets [19] after completing all tasks for the interface, both stationary and moving. The process was then repeated for the other interface starting with the demonstration. For details, we refer readers to our earlier work.

6.1.3 Results

During the experiment, eight moving tasks were repeated due to participants not correctly following instructions or due to outside distractions. This corresponds to 2.5% of all tasks included in analysis. No tasks had a trial time exceeding three standard deviations of the mean for the respective *interface* × *movement type* combination of factors, so no tasks were discarded as outliers.

Task Times

There was a significant main effect for *interface* ($F_{1,19} = 6.284, p < 0.05$), with faster task times for rate based scrolling (mean: 16.2 seconds) than flick scrolling (mean: 17.3 seconds). This difference occurred for both stationary and moving tasks, with no significant *interface* × *movement type* interaction. There was also no significant difference between movement types. Task times are summarised in Figure 6.2a.

Errors

When analysing the average error per task (that is, the difference between the participant's count and the actual count), there was a significant main effect for *interface* ($F_{1,19} = 6.263, p < 0.05$), with higher errors for rate based scrolling (mean: 2.1) than for flick scrolling (mean: 1.7). This, along with the significant difference in task times, indicates that participants chose a different speed-accuracy tradeoff when using rate based scrolling and suggests that rate based scrolling may be well suited to skim reading tasks where speed is valued over accuracy. There was no main effect for movement type or any significant interactions. Errors are shown in Figure 6.2b.

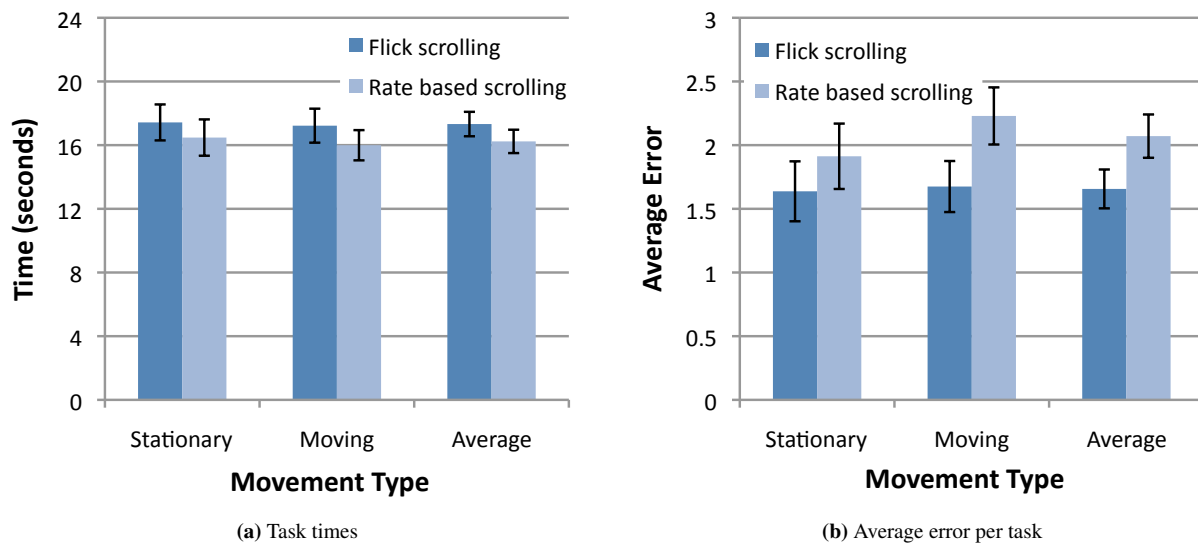


Figure 6.2: Task times and error rates for the reading evaluation. Error bars show standard error.

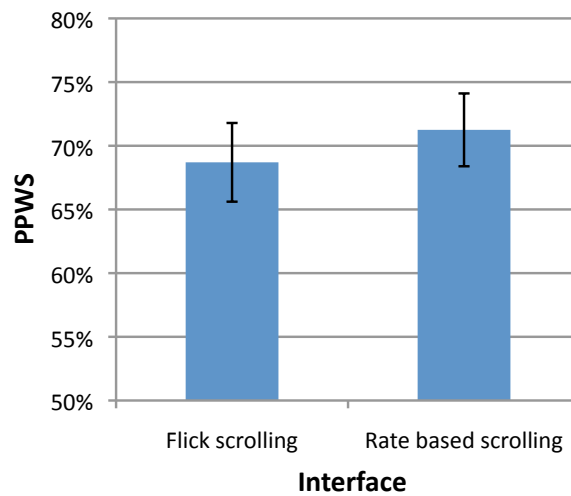


Figure 6.3: Percentage preferred walking speed (PPWS) for the reading evaluation. Note the non-zero y origin. Errors bars show standard error.

Percentage Preferred Walking Speeds (PPWS)

There was a marginally significant main effect for interface ($F_{1,19} = 3.232, p = 0.088$), with participants walking slightly faster for rate based scrolling tasks (71.2% of preferred walking speed) than for flick scrolling tasks (68.7%). This is shown in Figure 6.3.

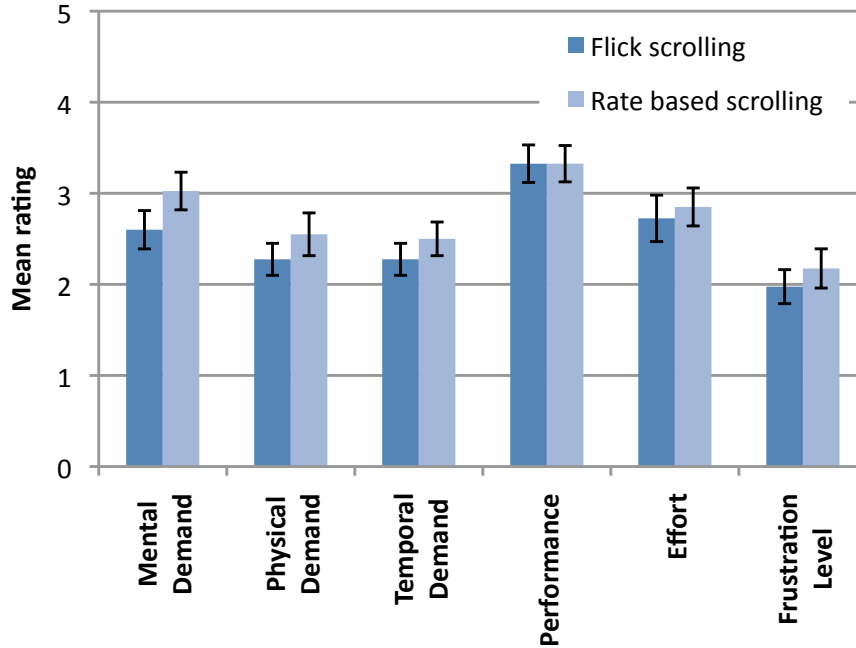


Figure 6.4: Mean NASA-TLX responses for the reading evaluation. Lower numbers are better except for performance. Errors bars show standard error.

Preferences

Participants were split over which interface they preferred. Overall, eight preferred rate based scrolling and nine preferred flick scrolling, while the other three were neutral. Five preferred rate based scrolling for both moving and stationary tasks, while seven thought the same about flick scrolling. Four preferred rate based scrolling for moving tasks and flick scrolling for stationary tasks, while another four thought the exact opposite. Finally, ten participants agreed that they would like support for rate based scrolling as the main scrolling technique in web browser and other applications on the iPod touch and iPhone, while six disagreed and four were neutral.

There was also no clear consensus in participants' comments. On one hand, a number of participants praised flick scrolling, describing it as intuitive, easier to control and having better precision than rate based scrolling. The main criticisms of rate based scrolling included difficulty in controlling the rate of scrolling and occlusion problems, although several participants stated that they thought they would improve with practice. On the other hand, rate based scrolling was also described positively by many participants, often for the same reasons. Comments included that it was intuitive, easier to control, provided a consistent scrolling speed and required less thinking. Others commented that it was good for long documents and when just skimming, corroborating the above conclusions about the different speed-accuracy tradeoff for rate based scrolling.

The mean NASA-TLX responses were similar for both interfaces on all measures, with no significant differences. A summary of responses is shown in Figure 6.4.

6.2 Recall Evaluation for Zero-Order Scrolling

Hinckley et al. [20] proposed a quantitative methodology for evaluating scrolling techniques that involves repeatedly navigating between two points in a document, varying the scrolling distance and the tolerance of scrolling. This methodology allows us to analysis which scrolling distances zero-order scrolling works well for as well as how precisely it can be used. We adapted it to suit the iPod touch interface and to reduce the time requirements and performed an evaluation comparing zero-order scrolling with flick scrolling. We refer the interested reader to the original paper for the background behind the original methodology, and describe the adapted methodology that we used below.

6.2.1 Participants and Apparatus

12 computer science students (three female) with a mean age of 25 participated in the evaluation. Five had had previous experience with an iPod touch or iPhone. Participants were given a \$10 shopping voucher for participating in this and the previous two evaluations.

The evaluation was performed on a second generation iPod touch running iPhone OS 2.2.1. The display's resolution was 480×320 pixels and it was always oriented in portrait.

6.2.2 Task

Tasks consist of a vertically scrolling view containing 600 lines of English text from a public domain book. The text within the view is rendered in 16 point Helvetica, allowing approximately 23 lines of text to be visible at any time. Lines are numbered to simulate scrolling in a familiar document, as in the original methodology and first conceived by Buxton and Myers [8]. Tasks require participants to move back and forth between two target lines, which are highlighted in either red or blue for the first and second targets respectively. Only the next target is highlighted at any one time; on acquisition of a target the next target is highlighted and the highlighting for the current target is removed.

The scrolling view is 300 pixels wide and 456 pixels high, with the status bar hidden. A "frame" is shown on the left of the view, sized based on the tolerance of the particular task. The frame remains stationary while scrolling, with participants aiming to scroll until the target line is completely within the range indicated by the frame. The top 24 pixels of the display shows task information, including the line number of the current target line and the participant's progress through the experiment. The task interface is shown in Figure 6.5.

For timing purposes, task times begin as soon as the previous target is acquired, rather than when the first scrolling action takes place (although we record both times). This is done since there may be a difference in cognitive preparation time for scrolling between the two interfaces, for example zero-order scrolling could conceivably have longer preparation times if participants consider what position in the document they wish to scroll to before touching the display.

We differed from the original methodology to determine when a target was acquired; in the original methodology the participant hit the caps lock key to confirm their selection. Typical usage for zero-order scrolling involves touching the display, moving the fingers until the target is attained, then releasing the fingers. For flick scrolling, flick motions are typically used when the target is not close, which results in the view continuing to scroll for several seconds after the finger is released from the screen, during which the display is likely to be touched again to scroll further. Once the target is close, users typically drag a finger across the screen and then release it once reaching the target, which results in no further scrolling when the finger is released. In both cases, if the display is untouched, the view is not scrolling and the target line is within the range indicated by the frame, it indicates that the user is satisfied with their selection and we deem the task completed. If these criteria are satisfied but the target line is not within the range indicated by the frame, we record it as an error provided the target line is sufficiently close to the frame (within 128 pixels - approximately the maximum distance for the target line to remain on screen with the maximum frame size). Participants must successfully complete a phase before progressing to the next one; an error does not automatically cause progression. This method reduces the accuracy of the error numbers, but removes the variable time it would take participants to communicate to the program that they have finished acquiring the target and prevents the need for clutter that adding such a function to the interface would create.

1/36 Line 191

175: she had often great enjoyment
 176: out of doors. Her favourite walk,
 177: and where she frequently went
 178: while the others were calling on
 179: Lady Catherine, was along the
 180: open grove which edged that
 181: side of the park, where there was
 182: a nice sheltered path, which no
 183: one seemed to value but herself,
 184: and where she felt beyond the
 185: reach of Lady Catherine's
 186: curiosity.
 187:
 188: In this quiet way, the first
 189: fortnight of her visit soon passed
 190: away. Easter was approaching,
 191: and the week preceding it was to
 192: bring an addition to the family at
 193: Rosings, which in so small a
 194: circle must be important.
 195: Elizabeth had heard soon after
 196: her arrival that Mr. Darcy was
 197: expected there in the course of a

Figure 6.5: The task interface for the zero-order scrolling evaluation. The user must try to align the target line (line 191) inside the frame (shown on left).

6.2.3 Procedure and Design

The evaluation was analysed as a $2 \times 3 \times 2$ repeated measures analyses of variance (ANOVA) for factors *interface* (flick scrolling and zero-order scrolling), *distance* (D, the distance between targets) and *target width* (W) with task time and error rate as dependent variables. To reduce the evaluation length we used fewer combinations of scrolling distance and target widths than Hinckley et al.'s evaluation, using scrolling distances of 20, 80 and 320 lines and target widths of 5 and 10 lines; the indices of difficulty of the six distance-width combinations have a roughly even spread. The *interface* factor was counterbalanced, while the other factors were randomised.

For each interface, participants performed a block of practice trials followed by two blocks of timed trials. Each block consisted of a trial of each of the six combinations of scrolling distance and target width in random order. Trials consisted of six phases of reciprocal movement between two targets in practice blocks and 10 phases in timed blocks, with the first phase starting after scrolling to the first target and ending after scrolling to the second target. The first two phases of each trial were excluded from task time analysis. There were therefore 16 recorded phases for each distance-width-interface combination.

Participants began the experiment by providing basic demographic information and reading a brief overview of the evaluation. They were then given a demonstration of one interface. Next, they completed the one practice block and two timed blocks for the interface, as described above, before filling out NASA TLX worksheets. This was then repeated for the second interface. Participants were then asked several questions about comparisons between the two interfaces.

6.2.4 Results

Data for phases which had task times greater than three standard deviations away from the mean for the relevant *interface* \times *distance* \times *target width* combination were discarded as outliers. This amounted to 71 phases, or approximately 3.1% of all non-practice phases. The outliers were spread evenly across the factors.

Task Times

There was a main effect for interface ($F_{1,11} = 6.127, p < 0.05$), with tasks for zero-order scrolling (mean: 2.92 seconds) faster than flick scrolling (mean: 3.15 seconds), however this was mainly a result of the distances between targets chosen for the evaluation. There were also main effects for distance ($F_{2,22} = 205.302, p < 0.001$) and target size ($F_{1,11} = 135.301, p < 0.001$), with longer distances and smaller target sizes respectively resulting in longer task times.

There was a significant *interface* × *distance* interaction ($F_{2,22} = 135.301, p < 0.001$), with faster task times for flick scrolling for short distances (20 lines) and zero-order scrolling for long distances (320 lines). Notably, times for flick scrolling increased greatly for larger distances, while they only increased slightly when using zero-order scrolling. This matched expectations since flick scrolling, in practice, has a limited speed so it takes much longer to scroll a long distance even when it is known that the target is a long distance away. With zero-order scrolling, on the other hand, participants could take advantage of their knowledge about the location of the target and longer times for longer distances were just a result of less accuracy about the exact location of the target when it was further away. For short distances, it was relatively simple for participants to scroll a short distance with flick scrolling, while for zero-order scrolling they had to remember where they were in the document, which was a more taxing task. Task times for different interfaces and distances are shown in Figure 6.6a.

There was also a significant *interface* × *target size* interaction ($F_{1,11} = 17.511, p < 0.005$). When the target size was 5 lines, the two interfaces performed similarly on average. When the target size was 10 lines, zero-order scrolling performed better than flick scrolling. This result indicates that zero-order scrolling has a weakness for making precise selections and is more suited to scrolling to approximate locations. Task times for different target sizes are shown in Figure 6.6b.

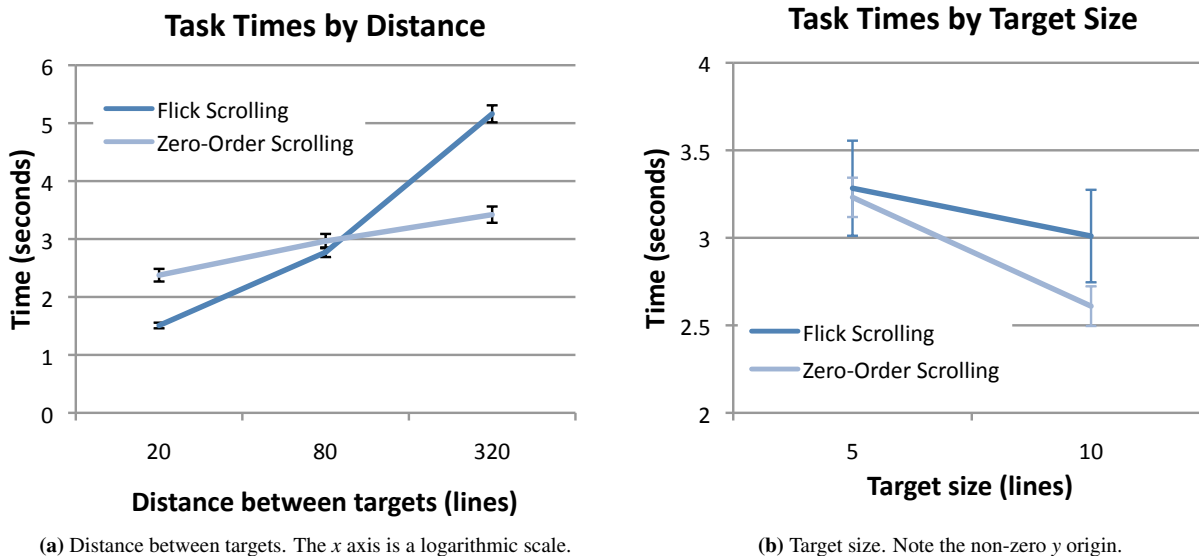


Figure 6.6: Task times by distance and target size for the recall evaluation. Error bars show standard error.

Learning Effects

Average task times for each phase are shown in Figure 6.7. Note that because we included the first two phases, which were not included in the task time analysis above and have much larger times than the other phases, outliers were not excluded when analysing learning effects. For zero-order scrolling, there was a clear learning effect with task times closely following a power law curve ($R^2 = 0.9239$). Task times reduce the quickest in the early phases, but continue to slowly decrease in later phases. For flick scrolling, task times also quickly decrease in the early phases, but there is

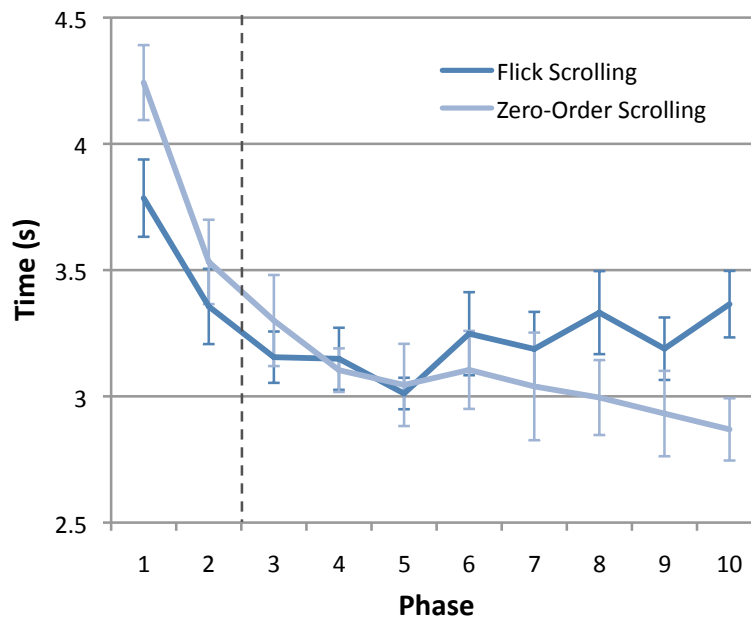


Figure 6.7: Task times by phase for the recall evaluation. Phase 1 corresponds to scrolling to the second target after acquiring the first target for the first time. Phases on the left of the dotted line were not included in analyses. Note the non-zero y origin. Error bars show standard error.

no clear trend in the later phases and the task times across phases do not follow a power law well ($R^2 = 0.3151$). This can be explained with similar reasoning to the *interface* \times *distance* interaction above; for flick scrolling, it is easy to reach the practical speed limit of scrolling, so even when the participant knew where the target was they could not get there any faster with practice. For zero-order scrolling, on the other hand, additional practice led to greater precision in estimating the target's location in the document, allowing for faster target acquisition.

Although the *interface* \times *phase* interaction is clear by the crossover effect in Figure 6.7, we also confirmed it by analysing the data as a 2×10 repeated measures analyses of variance for factors *interface* and *phase*. This shows a significant *interface* \times *phase* interaction ($F_{10,110} = 3.534, p < 0.001$).

Error Rates

There was a main effect for interface ($F_{1,11} = 32.415, p < 0.001$), with zero-order scrolling resulting in fewer errors per task (mean: 0.053) than rate based scrolling (mean: 0.159). Caution should be taken interpreting this result, however, since there may have been false positives for flick scrolling. For example, if the view stopped scrolling near the target after a participant released their finger but before they placed it again, it would have been counted as an error, however it may just have been the participant being slow to start the next scroll action. On the other hand, it could be an indication of a real difference. This could potentially be explained by imprecision caused by uncertainty about how much scrolling will occur after a flick motion ends, or by greater care taken to correctly acquire targets with zero-order scrolling since the cost of an error is much greater than with flick scrolling.

There was also a main effect for target size ($F_{1,11} = 11.279, p < 0.01$), with 5 line targets having a greater number of errors (mean: 0.133 per task) than 10 line targets (mean: 0.079). There was no main effect for distance and no significant interactions. Error rates for different interfaces and target sizes are shown in Figure 6.8a.

The type of each error, that is whether it was undershooting or overshooting the target, was also recorded. Figure 6.8b shows the frequency of each type of error for each interface. Undershoots were more common than overshoots for both interfaces. There was no apparent interaction between error type and interface, with undershoots being 63% of errors for flick scrolling and 64% of errors for zero-order scrolling.

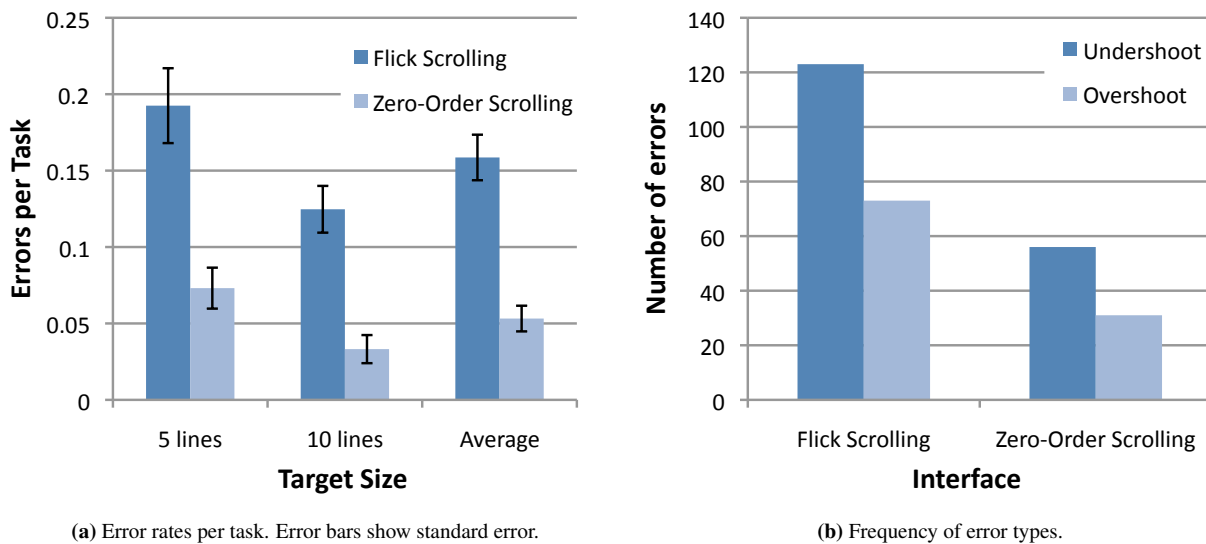


Figure 6.8: Error rates and types for the recall evaluation

Preferences

Of the 12 participants, nine thought that zero-order scrolling was quicker for getting to the approximate area of a target, while three thought the same for flick scrolling. All but one participant thought that flick scrolling was better for making precise selections. These preferences confirm the previously discussed *interface* × *target size* interaction, in that zero-order scrolling can be used to quickly get near a target but is not as good as flick scrolling for precise selections. Overall, nine participants preferred flick scrolling and three preferred zero-order scrolling.

Participants were also asked if they would like zero-order scrolling to be the main scrolling technique in web browser and other applications on the iPhone. Eight participants disagreed, most strongly, while two agreed. Participants were then asked if they would like zero-order scrolling to be available in combination with a relative technique in these applications, with our multitouch approach used as an example. For this question, eight agreed and just one disagreed.

The most common comment from participants was that they could not refine their position after releasing their finger when using zero-order scrolling. It was described as “very fiddly to control” by one participant and another commented that “flick scrolling felt a lot more natural”. Many of the participants’ issues with zero-order scrolling would be rectified if used in combination with a rate based approach. At the other end of the spectrum, several participants noted that zero-order scrolling was useful for scrolling long distances, with one even describing it as “fun” in such cases.

The mean NASA-TLX responses were similar for flick scrolling and zero-order scrolling for most measures, although flick scrolling was significantly better than zero-order scrolling for mental demand (Wilcoxon $z = 1.66, p < 0.05$). A summary of responses is shown in Figure 6.9.

6.3 Discussion and Future Work

We have shown that rate based scrolling achieves better task performance than flick scrolling for reading tasks, at the cost of a slight loss in accuracy. When rated subjectively, rate based scrolling is very competitive with flick scrolling. These results are promising considering that most participants had been previously exposed to flick scrolling, either in earlier evaluations or from past experience using an iPod touch or iPhone. Additionally, results indicate that rate based scrolling is well suited to skim reading and to scrolling short to medium length distances.

The recall evaluation for zero-order scrolling confirmed that the technique improves task performance when acquiring

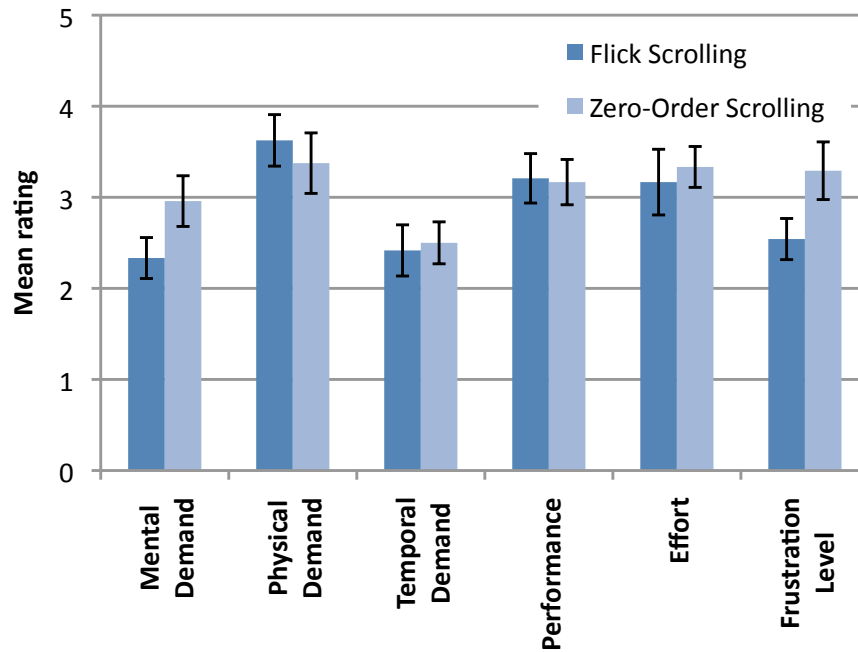


Figure 6.9: Mean NASA-TLX responses for the recall evaluation. Lower numbers are better except for performance. Errors bars show standard error.

targets in known locations that are far away. However, it also highlighted its weaknesses for scrolling short distances and precisely selecting targets.

These results confirm our hypothesis that rate based scrolling and zero-order scrolling suit different, complementary tasks and a hybrid approach is worth pursuing. Adding rate based scrolling to a zero-order scrolling interface, for example, solves the problems zero-order scrolling has for scrolling short distances and for refining the current location. Alternatively, zero-order scrolling could be combined with flick scrolling, which is perhaps better suited than rate based scrolling for making precise selections while rate based scrolling is best suited to slightly longer distances than flick scrolling. Further evaluations would be needed to compare hybrid approaches to standalone techniques and to find which hybrid approach works best.

Additional further work may involve creating a two dimensional version of both rate based scrolling and zero-order scrolling. Since they currently use only one dimension for input, this should be relatively simple to accomplish. We can extend zero-order scrolling to map both the horizontal and vertical touch locations onto horizontal and vertical positions in the document. Additionally, for rate based scrolling we can take both the x and y offsets from the touch origin to determine the scrolling speed. Drift zones could be added to the left and right of the display, and moving the finger into a drift zone would move the touch origin away along the line between the current touch position and the touch origin. In both cases, dimensional stability should be considered; if a user wants to scroll down, for example, they are unlikely to be able to be precise enough to keep their finger in the same position horizontally, resulting in some unwanted horizontal scrolling. For rate based scrolling, this could be rectified by implementing threshold angles, such that the view only scrolls in a dimension if the movement from the touch origin in that dimension is sufficiently large relative to the movement in the other direction. For zero-order scrolling the problem is not as simple to solve since this technique would result in large jumps when the threshold is first exceeded, however a similar approach based on it could potentially be used.

7

Conclusions

We have designed, implemented and evaluated three navigation techniques for mobile devices, two of which can be combined into a hybrid approach using multitouch. A tiered alphabet scroller offers the ability to scroll to precise locations in long sorted lists. An initial evaluation showed that small target sizes are problematic but gave insights into usage patterns and future designs. We implemented a rate based scrolling technique as an alternative to flick scrolling and a zero-order scrolling technique that allows instantaneous scrolling to any point in the document. Evaluations showed that the rate based scrolling technique is competitive with flick scrolling and is well suited to skimming tasks, and that the zero-order scrolling technique is well suited for revisiting targets that are far away. The results lend support to a hybrid approach that gives the benefits of both techniques.

This report began with a description of five factors in the design space for mobile scrolling: mobility, purpose of scrolling, landmarks, dimensions and size. Our interfaces explore these factors in the following ways:

- All techniques used touch input so are not particularly disadvantaged in mobile contexts, however we evaluated rate based scrolling in both mobile and stationary settings to investigate the effects of movement.
- The three techniques are each designed for different task types: the tiered alphabet scroller for visual search tasks, rate based scrolling for visual search and analysis tasks, and zero-order scrolling for visual search and recall tasks.
- The tiered alphabet scroller is designed for a specific type of document involving landmarks in the form of letters, while the other techniques are designed for generic documents without landmarks.
- Our prototypes were all for scrolling in a single dimension, but rate based scrolling and zero-order scrolling would also work in two dimensions.
- The tiered alphabet scroller was developed with the hypothesis that it would perform better with longer lists, although this was not shown to be the case for the current implementation. Our hybrid scrolling technique was also designed to work well for long documents, using zero-order scrolling to approach a target and then rate based scrolling for extra precision.

Combined, we have extensively explored the mobile scrolling design space and offered valuable insights into scrolling techniques.

Bibliography

- [1] Jason Alexander, Andy Cockburn, Stephen Fitchett, Carl Gutwin, and Saul Greenberg. Revisiting Read Wear: Analysis, Design and Evaluation of a Footprints Scrollbar. In *CHI '09: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page in press, New York, NY, USA, 2009. ACM.
- [2] Dzimitry Aliakseyeu, Pourang Irani, Andrés Lucero, and Sriram Subramanian. Multi-flick: an evaluation of flick-based scrolling techniques for pen interfaces. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1689–1698, New York, NY, USA, 2008. ACM.
- [3] Christine Anderson, Sandra G. Hirsh, and Andre Mohr. Wheels around the world: windows live mobile interface design. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 2113–2128, New York, NY, USA, 2008. ACM.
- [4] Caroline Appert and Jean-Daniel Fekete. Orthozoom scroller: 1d multi-scale navigation. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 21–30, New York, NY, USA, 2006. ACM.
- [5] Joel F. Bartlett. Rock 'n' scroll is here to stay. *IEEE Comput. Graph. Appl.*, 20(3):40–45, 2000.
- [6] Stephen Brewster. Overcoming the lack of screen space on mobile computers. *Personal Ubiquitous Comput.*, 6(3):188–205, 2002.
- [7] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. Navigation techniques for small-screen devices: An evaluation on maps and web pages. *Int. J. Hum.-Comput. Stud.*, 66(2):78–97, 2008.
- [8] W. Buxton and B. Myers. A study in two-handed input. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326, New York, NY, USA, 1986. ACM.
- [9] Sung-Jung Cho, Changkyu Choi, Younghoon Sung, Kwanghyeon Lee, Yeun-Bae Kim, and Roderick Murray-Smith. Dynamics of tilt-based browsing on mobile devices. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 1947–1952, New York, NY, USA, 2007. ACM.
- [10] Andy Cockburn and Andrew Gin. Faster cascading menu selections with enlarged activation areas. In *GI '06: Proceedings of Graphics Interface 2006*, pages 65–71, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.
- [11] Andy Cockburn, Carl Gutwin, and Jason Alexander. Faster document navigation with space-filling thumbnails. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1–10, New York, NY, USA, 2006. ACM.
- [12] Andy Cockburn, Carl Gutwin, and Saul Greenberg. A predictive model of menu performance. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 627–636, New York, NY, USA, 2007. ACM.
- [13] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1):1–31, 2008.
- [14] P. Eslambolchilar and R. Murray-Smith. Tilt-based automatic zooming and scaling in mobile devices - a state-space implementation. In *Proceedings of the Mobile HCI conference (Mobile HCI 2004)*, 2004.
- [15] Parisa Eslambolchilar, John Williamson, and Rod Murray-Smith. Multimodal feedback for tilt controlled speed dependent automatic zooming. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, 2004.
- [16] Stephen Fitchett and Andy Cockburn. Evaluating reading and analysis tasks on mobile devices: A case study of tilt and flick scrolling. In *OZCHI '09: Proceedings of the 21st Australasian Conference on Computer-Human Interaction*, pages 225–232, New York, NY, USA, 2009. ACM.

- [17] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. 1954. *J Exp Psychol Gen*, 121(3):262–269, September 1992.
- [18] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.
- [19] S. Hart and L Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In P Hancock, editor, *Human Mental Workload*, pages 139–183, 1988.
- [20] Ken Hinckley, Edward Cutrell, Steve Bathiche, and Tim Muss. Quantitative analysis of scrolling techniques. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 65–72, New York, NY, USA, 2002. ACM.
- [21] Takeo Igarashi and Ken Hinckley. Speed-dependent automatic zooming for browsing large documents. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 139–148, New York, NY, USA, 2000. ACM.
- [22] Jesper Kjeldskov and Jan Stage. New techniques for usability evaluation of mobile systems. *International Journal of Human-Computer Studies*, 60:599–620, 2004.
- [23] Jari Laarni. Searching for optimal methods of presenting dynamic text on different types of screens. In *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, pages 219–222, New York, NY, USA, 2002. ACM.
- [24] Bonnie MacKay, David Dearman, Kori Inkpen, and Carolyn Watters. Walk 'n scroll: a comparison of software-based navigation techniques for different levels of mobility. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 183–190, New York, NY, USA, 2005. ACM.
- [25] Takashi Miyaki and Jun Rekimoto. Graspzoom: zooming and scrolling control model for single-handed mobile interaction. In *MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–4, New York, NY, USA, 2009. ACM.
- [26] Jonathan Mooser, Suya You, and Ulrich Neumann. Large document, small screen: a camera driven scroll and zoom control for mobile devices. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 27–34, New York, NY, USA, 2008. ACM.
- [27] I. Oakley and S. O'Modhrain. Tilt to scroll: evaluating a motion based vibrotactile mobile interface. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 40–49, March 2005.
- [28] Ian Oakley, Jussi Ängeslevä, Stephen Hughes, and Sile O'Modhrain. Tilt and feel: Scrolling with vibrotactile display. In *Proceedings of Eurohaptics*, pages 316–323, 2004.
- [29] Gustav Öquist and Kristin Lundin. Eye movement study of reading text on a mobile phone using paging, scrolling, leading, and rsvp. In *MUM '07: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 176–183, New York, NY, USA, 2007. ACM.
- [30] H. Petrie, S. Furner, and T. Strothotte. Design lifecycles and wearable computers for users with disabilities. Presented at The First Workshop on Human-Computer Interaction With Mobile Devices, 1998.
- [31] Antti Pirhonen, Stephen Brewster, and Christopher Holguin. Gestural and audio metaphors as a means of control for mobile devices. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 291–298, New York, NY, USA, 2002. ACM.
- [32] Ivan Poupyrev, Shigeaki Maruyama, and Jun Rekimoto. Ambient touch: designing tactile interfaces for hand-held devices. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 51–60, New York, NY, USA, 2002. ACM.

- [33] Jun Rekimoto. Tilting operations for small screen interfaces. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 167–168, New York, NY, USA, 1996. ACM.
- [34] Volker Roth and Thea Turner. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1523–1526, New York, NY, USA, 2009. ACM.
- [35] Itiro Siio and Hitomi Tsujita. Mobile interaction using paperweight metaphor. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1325–1330, New York, NY, USA, 2006. ACM.
- [36] Erum Tanvir, Jonathan Cullen, Pourang Irani, and Andy Cockburn. Aamu: adaptive activation area menus for improving selection in cascading pull-down menus. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1381–1384, New York, NY, USA, 2008. ACM.
- [37] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422, New York, NY, USA, 2004. ACM.
- [38] Shumin Zhai, Barton A. Smith, and Ted Selker. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 286–293, London, UK, UK, 1997. Chapman & Hall, Ltd.

A iPhone Development Issues

This section describes several iPhone development pitfalls that were encountered during this project and provides the solutions used. This acts as a guide to anyone doing related research in the future.

A.1 View Size and Memory

One issue was related to the list of words used in the alphabet scroller prototype. The iPhone OS will not render views which are greater than 16384 pixels in either dimension, however the list we wanted to render was longer. Additionally, as we were not using the standard table view control, so as to allow for greater customisation, this problem was not handled for us. The solution we used was to break the list up into smaller views, with references to the point in the list data to which each segment starts. These segments were all added as subviews to a list view. The parent view was greater than 16384 pixels, but as it was drawn entirely by smaller subviews, this was sufficient.

A related issue was memory issues. The iPhone OS renders all views when they are added to the view hierarchy, regardless of whether they are visible. For example, the entire list is rendered even though only a small section is displayed at a time. Unfortunately this consumes a lot of memory, and the memory capacity on the iPod touch and iPhone is low compared to desktop machines. As a result, it was common during development for our prototypes to crash due to lack of memory. The solution was to only have visible segments of the list as subviews. Scroll events are intercepted to add or remove segments from the list of subviews. Other views are still stored in memory, but they are not rendered, which greatly reduces the memory footprint.

A.2 Frame Rate Control

When animated views are used (such as for rate based scrolling when a finger is held down), it is tempting to redraw the view frequently to produce a smooth animation. Redrawing too frequently, however, consumes a great amount of resources. This is not always apparent, as the view will usually appear as normal, however when this happens events are handled very slowly and the response time only lengthens the longer the view is redrawn at the high frequency. This caused problems, for example, when a view continued to scroll for seconds after the finger was released as it was too busy redrawing the view repeatedly to handle the *touch ended* event. The solution was to adjust the time between redraws to ensure the animation was still smooth while keeping the redraw frequency below the threshold at which the device's ability to process was overwhelmed.

A.3 Faulty Touch Sensors

On the development device, all touches more than 470 pixels from the top of the display were reported as being a touch 470 pixels from the top (note that the display is 480 pixels tall). This caused problems when detecting proximity to the display edges for rate based scrolling. The workaround was to decrease the scroll view's height by 20 pixels and set the background of the parent view to black. Touch events were then intercepted so that a screen edge could be simulated 20 pixels from the bottom of the display.

B

Fishing Through Data

Despite warnings not to fish through experiment data too much, the temptation was irresistible. Indeed, after enough fishing, a fish was caught. This fine specimen is shown in Figure B.1.

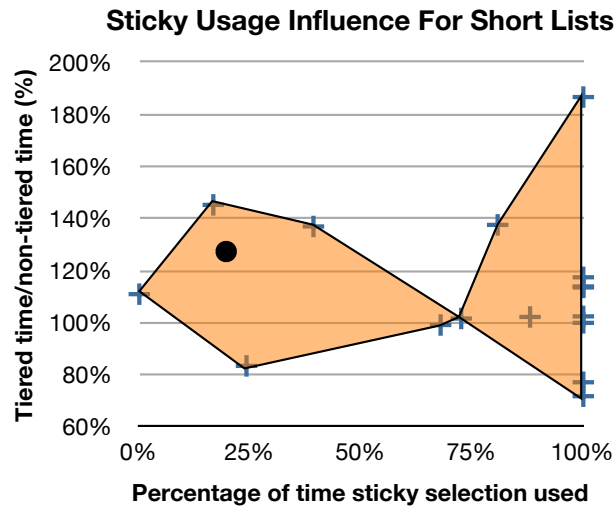


Figure B.1: The effect of use of sticky selection in the tiered alphabet scroller on task performance for short lists. Each data point represents a participant. The x -axis represents the percentage of the time that the participant used sticky selection. The y -axis represents the time to complete tasks using the tiered scroller relative to the non-tiered scroller.