A Flick in the Right Direction

An Evaluation of Simple Gesture Based Controls

Michael Moyle   Dr. Andy Cockburn

(Supervisor)

November 2, 2001

Abstract

Flick gesture systems provide an efficient alternative to common interface controls. This paper investigates the properties of natural flick gestures. These properties include gesture magnitude, gesture duration and angular error. Flick gestures are then applied to the frequent user task of web navigation. The web gesture system is then evaluated to determine the efficiency and learnability of the flick gesture in a real world application. Results showed that subjects were able to navigate significantly faster using the gesture system compared to the standard web browser. Subjects were also extremely enthusiastic about the technique, with many expressing their wish that "all browsers should support this". Subjects also found the gesture system easy to learn.

Contents

Chapter 1

Introduction

The traditional user interface provides many ways to access commands, from shortcut keys to desktop icons. The devices used to control these interfaces have also evolved, from the keyboard and mouse of a traditional computer, to the pen based interface of the Personal Digital Assistant (PDA). As this technology develops, so too does the opportunity to add new features to existing environments.

In a typical Windows, Icons, Mouse and Pointers (WIMP) environment, two of the most common means of accessing commands are menus and buttons. An alternative to using these methods is the use of gesture enabled interfaces. Traditionally, gestural interfaces have been limited to pen based systems. In this situation they provide a means of textual input by way of a special character alphabet, such as Unistrokes (Goldberg & Richardson 1993) or Graffiti (MacKenzie & Zhang 1997).

Gestural based input in the WIMP environment is relatively uncommon, although the efficiency gain from using such a system has the potential to save user time. Nielsen states: "The smallest of usability problems, when multiplied across thousands of users, becomes a source of untold frustration" (Nielsen 1993). This statement emphasises the fact that good interface design is very important. This report looks to evaluate the mouse as a means of gesture input, in terms of accuracy, time and user satisfaction.

To aid implementation of gesture input systems, a set of guidelines are suggested, that draw on both the related work in the area and the empirical results gathered. These empirical results are then applied to guide an implementation of the flick gesture in a web browser. The system is then evaluated to determine the viability of the flick gesture in such an application, in terms of performance, learnability and user satisfaction.

Chapter two presents previous work in relation to computer gesture systems including marking menus, flick gestures, and a selection of common gesture applications. Chapter three discusses the design issues in a flick gesture implementation. This is followed by a set of empirical evaluations designed to determine the properties of natural flick gestures. Chapter five presents the implementation details of a web browser gesture system, and is followed by an evaluation of the relative performance of the standard and enhanced versions of the web browser. The final chapter presents the conclusions of the report.
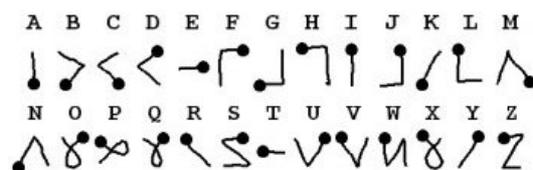

Figure 1.1:
The Unistrokes alphabet (MacKenzie & Zhang 1997).

Chapter 2

Related Work

Gestures are used in everyday life and form a major part of natural human communication. The Oxford dictionary defines the word 'gesture' as: "a move calculated to evoke a response from another or to convey intention". Gestures help emphasise other forms of communication, by the use of body or hand movements, some of which are simple, and some very complicated. As stated in the gesture definition, gestures convey information, and are sometimes the only form of interaction between people. Like spoken language, gestures have different meanings depending on the country of residence. They also have different meanings depending on the context of the conversation.

Gestures in the area of computing have been applied to a range of different tasks, primarily as an alternative to conventional input techniques, such as typing on the keyboard and button activation with the mouse. There are two methods used to control common gesture based interfaces; the pen or stylus, and the mouse.

Pen based gesture systems have been successfully applied to mobile computing using Personal Digital Assistants (PDAs), where devices such as the mouse and keyboard are impractical. In a similar fashion, pen computers have used gesture recognition as a means of input, and implementations have been seen in operating systems, such as Windows for Pen Computing V1.0 for example. Digital whiteboards also use the pen as a form of gesture input. This allows the user to issue commands from the current location of the pen, rather than reaching to a button projected at the perimeter of the board.

Mouse based gesture systems have seen relatively few implementations when compared to the pen. One reason is that the mouse was designed for pointing, whereas a pen is also used for writing and drawing (Pedersen, McCall, Moran & Halasz 1993). The mouse is indirect as physical mouse movement does not directly map to the movement of the cursor. In comparison, the pen is direct, as the cursor moves to where the pen is pointing. This means gestural techniques are more natural with a pen than a mouse. One application that does apply gesture techniques with the mouse is the marking menu (Kurtenbach, Sellen & Buxton 1993). Marking menus, an extension of pie menus, were originally developed for pen computing, although work well when used with the mouse, as the movements required are relatively simple.

The following sections discuss the above techniques in more detail.

2.1   Marking Menus

Marking menus are an extension of pie menus. Pie menus are circular menus with each sector representing a menu item (Callahan, Hopkins, Weiser & Shneiderman 1988) as shown in Figure 2.1(a). Pie menus provide an alternative to linear menus such as that shown in Figure 2.1(b). When activated, the centre of the pie is placed at the current pointer position, and a menu item is selected by dragging to one of the pie segments. Pie menus help to minimise Fitts' Law constraints on time-to-target, as each menu item is equally spaced from the initial pointer position. In theory, a movement of one pixel is sufficient to reach any of the menu items, and further movements result in the target effectively becoming larger.

Marking menus (Kurtenbach & Buxton 1991) extend the pie-menu approach by allowing users to select an item before the menu appears. The menu is only displayed after a delay period, commonly about 1/3s.
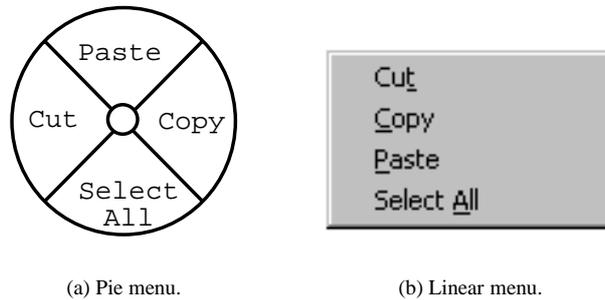
(a) Pie menu.   (b) Linear menu.

Figure 2.1: A comparison of pop-up menus.

If the user knows the location of the desired item, they can make a 'flick' in the appropriate direction to select the item. If the user hesitates, the pie menu is displayed to assist learning of the gesture set. Research has also shown that users eventually become expert users, using the mark-ahead feature 90% of the time (Kurtenbach & Buxton 1993).

The performance of marking menus can be measured by the time savings gained in comparison to other techniques. In an evaluation lasting a cumulative total of 36 hours, it was estimated that the user completed the 16026 marking menu selections 3.66 hours sooner than an equivalent system using linear menus (Kurtenbach & Buxton 1994). This shows that for repetitive and common tasks, marking menus can provide great efficiency gains. In such tasks, the benefits can be attributed to the user knowing the location of menu items, and thus using the mark-ahead feature. Kurtenbach & Buxton (1994) point out that marking menus are less beneficial when the list of menu items dynamically change, as users will be unaware of each item's location in the menu.

2.1.1   Item Location

The location of items around the radial marking menu have been shown to have a significant effect on user performance. Increasing the number of items in the menu reduces user performance in terms of time and errors (Kurtenbach et al. 1993). To counteract the increase in menu size, Kurtenbach & Buxton (1993) found that an even number of items allowed better performance, due to the mental ability to segment the radial menu into equal portions. The paper also found that the location of items according to common metaphors also performed better in comparison to other layouts. Layouts of eight items, corresponding to the points on a compass, and twelve items, corresponding to the points on a clock, were shown to perform better than layouts of six and ten items for example. This was attributed to the cognitive burden of making a mark rather than the physical difficulty of a particular layout.

Kurtenbach et al. (1993) discussed the location of commands relative to others when using marking menus. The paper suggested operations that oppose each other in their action, such as cut and paste, or open and close, should be located opposite each other in a marking menu layout. This scheme provides an extra cue as to the location of a command, and also provides a degree of iconicity, as to invoke each command requires marks in opposite directions.

2.1.2   Input Device

Marking menus are not limited to a single device. Kurtenbach et al. (1993) conducted an evaluation to compare marking menu performance when using the pen, mouse and trackball. The study timed user selections from six different menu sizes ranging from four to twelve items. Each user conducted 40 menu selections for each menu size and each of the three devices. The item to be selected from the menu was randomly chosen by the evaluation software. The evaluation found the mouse and stylus outperformed the trackball. Using standard marking menus, the study also found no significant difference between the mouse

and pen in response time or errors. The paper also suggested that the mouse may be inferior to the stylus when making marks other than straight strokes.

Marking menus have also been applied to touch-pad input, controlled using the middle or index finger of the user's non-dominant hand. Balakrishnan & Patel (1998) presented the idea of a combined touch-pad and mouse controlled by the non-dominant hand. The technique, known as 'marking keys', allowed the non-dominant hand to execute shortcut commands by stroking the finger to select an item from the marking menu, while at the same time the hand can be used to control the mouse for spatial positioning.

The Balakrishnan & Patel (1998) paper included an evaluation to determine whether marks made in certain directions were easier to perform than others. The primary focus was to uncover the motor skills required to make a mark with the finger, rather than the cognitive barriers for a particular direction. The study found that in an eight item layout, the mark direction had a significant effect on the error rate, with the north-west direction proving the most difficult.

2.1.3 Hierarchic Marking Menus

Linear menus allow multiple sub-menus to spawn from the first level, providing an effective way to present hierarchical menu options. Marking menus also provide this ability, and have been evaluated by Kurtenbach & Buxton (1993). The study compared different levels of marking menus using both the pen and the mouse. Overall, the study found that hierarchical marking menus were a feasible approach for the presentation of many commands. In terms of performance, the pen was found to perform better than the mouse when making hierarchic marking menu selections, although when the number of items and depth of menu were small, performance was approximately similar.

2.1.4 Ink Trails

The traditional marking menu provides little feedback to the user when using the mark-ahead feature. This provides a benefit, as the user is unobstructed by the menu pop-up, although it can also be detrimental if an incorrect mark is made. An ink trail helps to provide a form of feedback that is unobtrusive to the user and the underlying application. Kurtenbach et al. (1993) applied this idea when making marks with both the pen and the mouse. The paper found that the stylus performed better than the mouse when leaving ink trails. This can be attributed to the proper use of the pen metaphor, as drawing on the screen actually leaves a visible mark.

The use of ink trails can also help to enforce idealised marks. Tapia & Kurtenbach (1995) discussed a system that left an ink trail that followed the path of the cursor, and when the item is recognised by the application, the drawn mark is removed, and replaced by the idealised mark for that item. It was suggested this may encourage expert behaviour, as users will adapt their marks to suit the idealised mark. This would help mark accuracy, and hence mark recognition. Although no empirical evaluation was conducted, the paper was based on the preferences of experienced marking menu users.

2.2 The Flick

Dulberg, Amant & Zettlemoyer (1999) evaluated the use of 'flicks', similar to the mark aspect of marking menus. The paper defined the flick gesture to be when "the left mouse button is briefly pressed and held, while the mouse is quickly moved a short distance. Releasing the mouse completes the gesture." They suggested the main benefit of the flick is that the precise targeting, or the zeroing-in phase of conventional Fitts' Law tasks, is bypassed. This means the user is not required to accurately locate the cursor at a precise point, rather the cursor is flicked in the direction of the target. This can be compared to the target acquisition required in pie menus, where as the cursor is moved from the centre to select a target, the target itself is actually increased (see Figure 2.1(a)).

The flick gesture also presents other desirable properties. After a flick command has been issued, the cursor remains close to the previous location. This helps to minimise the relocation required to resume the previous activity. Also, when compared to keyboard shortcuts, the flick does not require the user's hand to move from the mouse, which helps to minimise the time required to issue a command.

2.2.1 Flick Evaluation

In a laboratory based system, Dulberg et al. (1999) asked users to make flicks in arbitrary directions using the mouse. They then compared these results to issuing the same command using keyboard shortcuts and normal button activation. They found the flick gesture to be 26% faster than button selection. When compared to keyboard shortcuts, the time required was comparable, although "the flick gesture is superior to the enter gesture (keyboard shortcut) in terms of time required for the user to 'recycle' or get ready for the next task."

Time and distance measurements were taken, and found 90% of flicks lasted longer than 163ms, and were associated with movements of 21 pixels or more. The median distance travelled was 48.4 pixels. They also found that 90% of button clicks lasted less than 157ms, and were associated with a movement of only 3 pixels. The difference between optimal angle and the actual flick angle was also measured. Results showed a mean error of 5.73 degrees over all directions. They also mentioned that the variance for targets not located on the horizontal or vertical axes was greater, although an analysis of this variance was not presented.

Also, as part of the paper, an informal study using flicks on the Windows Desktop was included. Subjects were reported to have no problem learning the technique, and five out of the six participants said they would use it if available.

Although interesting results, the study did not mention the level of mouse acceleration used in the evaluation, a factor that could heavily influence the final results. Also, as each flick was generated for an arbitrary direction, an analysis of bias introduced as a factor of direction was not presented.

2.3 Gesture Applications

Gesture based systems have seen many implementations and sparked much research over the past decade. Gestures provide a means of access when the pen is the only practical device. They also provide an alternative to button or icon selection when used with the pen or mouse.

2.3.1 Character Input

The most well known form of gesture control is the realm of character input. Devices such as Personal Digital Assistants (PDAs) and pen computers make use of such systems, as often the stylus is the only input device available.

There are many forms of character input, primarily based around the character set that is used. Early forms of character input attempted to 'recognise' normal handwriting just as the user would normally write it. Although these systems were easy for users to learn, the recognition software often performed poorly due to the many different styles of handwriting people possess. To overcome this problem, a single stroke character set was introduced, primarily aimed at achieving better character recognition. Isokoski (2001) classifies this family of character sets as *Unistrokes*, as each is created with a single stroke. The following summarises the most common character input techniques:

- Unistrokes (Goldberg & Richardson 1993) — Unistrokes were first introduced as an alternative character set for the Roman alphabet. Each character is written with a single stroke which aids recognition as each character has a clearly defined beginning (pen-down) and end (pen-up). This allows the
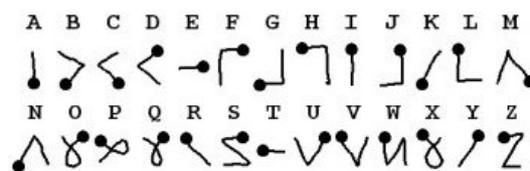

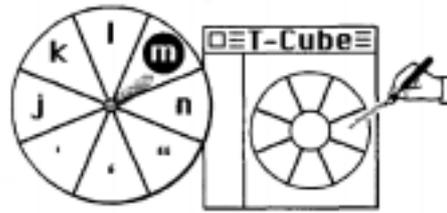Figure 2.2: A portion of the Unistrokes alphabet.

Figure 2.3: The T-Cube target (right), and the resultant marking menu (left) from clicking the pen in the middle right segment of the target.

recognition algorithm to focus on the resultant stroke rather than deciphering which parts of the curve belong to each character. Also, in comparison to handwriting recognition, the input space required is that of a single letter, as each character effectively overwrites those previously written. This also requires less movement of the wrist. The downside to this approach is the learning time required by the users, as each character is tailored to help the computer recognise the text, rather than aiding the user to remember each character.

The Unistrokes' alphabet is based on five different strokes; $| \, Z > L \propto$, each of which can be rotated 90 degrees, and given in two directions (see Figure 2.2). This gives access to 5 x 4 x 2 = 40 different strokes. In an evaluation of Unistrokes, writing time for each character varied. The dot was quickest at 91ms, the left-right gesture was 135ms, and the right-left gesture was 143ms. After using the system for one week, the performance increase was large, about half the time on average.

- T-Cube (Venolia & Neilberg 1994) — T-Cube, a self-disclosing alphabet, was first presented as an alternative to Unistrokes by Venolia & Neilberg (1994). The method used a combination of marking menus and flicks to produce a possible 72 different characters. Figure 2.3 shows the T-Cube target (right), which is used as a starting point. The user depresses the stylus in one of the nine possible regions of the target (the eight outer pie segments, plus the inner circle), and then proceeds to flick in the direction of the desired letter. Each region of the target gives access to eight characters, as shown by the left pie menu in Figure 2.3. As an example, the user starts in the middle right region of the target (right of Figure 2.3) which pops up the offset marking menu (left of Figure 2.3). The user then flicks in the NE direction selecting the character 'm'. Although the movement required for each character does not provide an iconic mapping, the use of marking menus aids the learning process.

- Graffiti (MacKenzie & Zhang 1997) — Graffiti also uses a single stroke to represent each character, although in comparison to Unistrokes and T-Cube, Graffiti was modelled closer to the Roman alphabet. This approach means each characters are easier for the user to learn and remember. Figure 2.4 shows an example of the Graffiti alphabet.

- Quikwriting (Perlin 1998) — Quikwriting is a technique that turns gestures into a set of scribbles. A description of the alphabet is given by Perlin (1998). The technique is somewhat similar to the

Figure 2.4: The Graffiti alphabet.

T-Cube method, and divides each area around the start position into eight cells. The character drawn depends on the entry and exit cell of the pen. Each character is recognised after an exit and entry from the central cell or start position, which means multiple characters can be entered without lifting the pen. Users reported an increase in performance over the commonly used Graffiti, as Quikwriting effectively allows "common words to become a single learned iconic gesture" (Perlin 1998).

Evaluation of Unistroke Character Sets

Many of the papers that introduced the character input methods above included evaluations measuring the performance in comparison to other techniques. A major problem when making these comparisons is the bias introduced by prior knowledge of a commonly used technique. Often, the test subjects are completely literate in the old technique, which means the new character set, no matter how promising, is considered to be unnatural. The main goal of these experiments is to compare expert performance, which is only attained after a long period of time. For this reason, Isokoski (2001) introduced a model for predicting writing time for a given 'unistroke' alphabet, that allows comparison between character sets without the need for lengthy user evaluations.

The model assigns complexity points to each character, which are determined in the following way (Isokoski 2001):

- Each straight line needed to draw the character has a value of one complexity point.
- If the stroke contains any round shapes, they are drawn with a minimal number of straight lines, hence a circle is drawn as a triangle.
- If the character consists of more than a single stroke, add another line to join stroke n to stroke $n+1$.
- Count the lines to get the time complexity for the character.

To obtain the overall complexity for the character set, the complexity for each individual character is multiplied by the probability that the character will occur in a piece of text. In the paper, the probabilities are modelled on the character's probability in the English language. The results of the paper gave complexity ratings of 2.76 for the Roman alphabet, 2.54 for Graffiti, and 1.40 for Unistrokes.

2.3.2   Gestures for Control of User Interfaces

Although relatively uncommon in comparison to traditional button selection techniques, gesture systems have been used in various desktop applications. In some cases, gestures provide another alternative to traditional WIMP techniques, and in others they provide the sole means of access to a system.

Graphical Editors

Graphical editors commonly have a large number of functions available to the user, and presenting these items efficiently can be a challenge. Kurtenbach, Fitzmaurice, Baudel & Buxton (1997) used marking menus in an attempt to avoid directing the user's attention away from the artwork. They also wanted to maximise the amount of screen space available for artwork, which meant the screen saving marking menus were of great benefit.

In an extension to the marking menu idea, Kurtenbach, Fitzmaurice, Owen & Baudel (1999) introduced the Hotbox, a marking system in which over 1200 commands were present with multiple items in each menu. Each of the three mouse button were assigned a different marking menu. The system has been implemented in the production version of Maya (Maya 2001), and an analysis of the users found three different groups. Some users chose to ignore the Hotbox, and used the regular GUI interface instead. Some used it as part of their work-flow, although not to its full extent. Finally, some users used it exclusively and extensively, commonly hiding the traditional menubars and toolbars, and using the Hotbox to issue commands. A benefit of the system was that it supported both novice and experts without requiring customisation or radically different behaviours.

Web Browsing

In web use, back navigation is an extremely common activity. Catledge & Pitkow (1995) found that, on average, each user visited 58% of URLs more than once. In a more recent study, Cockburn & McKenzie (2001) found the revisitation rate to be 81%.

Navigation to these previously seen pages is also well supported, including bookmarks, history browsing, and the back button. The Tauscher & Greenberg (1997) study found the back button to account for 41% of user actions, and the Catledge & Pitkow (1995) found this rate to be 30%. In comparison, the forward button was found to account for only 2% of user actions.

Microsoft Internet Explorer, and Netscape Navigator both support many shortcuts to issue the back command. Both support the 'Alt + Left-arrow' keyboard shortcut, and Explorer also supports the 'Backspace' button. Tauscher & Greenberg (1997) reported 50% of user actions while browsing were link selection, a function almost exclusively designed for the mouse. The downside to using keyboard shortcuts for web browsing is the homing time required to move the hands from the mouse to the keyboard and back again.

An alternative to button selection and keyboard shortcuts is the use of gestures. Opera Software (2001) in their web browser Opera employ a gesture system that allows shortcuts for many common commands. Mozilla (Mozilla Organization 2001) also recently added a gesture recognition system to its web browsing capabilities.

Web browsing using PDAs poses many challenges, which are primarily based around the typically small display. Unlike desktop computers, PDAs have only a few inches to display information, which means normal web pages are not easily displayed on screen. Screen space is at a premium, and therefore much thought is required for an effective layout. In addressing the need for web browsing on PDAs, Buyukkokten, Garcia-Molina, Paepcke & Winograd (2000) introduced "Power Browser", which was implemented on 3Com's Palm Pilot. The need to save screen space was addressed by using gestures to invoke common commands in place of buttons. As an example, while the browser was in text mode, a flick gesture in the left direction replaced the function of the back button. The Power Browser also incorporated many other screen saving techniques such as wrapping the page into text summaries for example. Overall, pen movement was reduced by 42% in comparison to other systems.

Gaming

The authors of Black & White (Lionhead 2001), a medieval role playing game, identified a major problem when designing the user interface. The game itself was three times bigger than any attempted by the Lionhead studio, and therefore required many commands. In previous developments, icons were used to control the different aspects of the game. Black & White contained too many icons to be displayed at a single time, so in an attempt to reduce screen clutter, the interface design implemented a gesture recognition system. User feedback for the system was positive, although some users preferred to use the keyboard shortcut alternative provided, possibly due to the complex nature of the gestures used.

Digital Whiteboards

Whiteboards provide a means of temporary idea storage during formal and informal meetings. Digital whiteboards allow this temporary storage to be made permanent. They also provide a means of direct interactivity via the whiteboard pen, which is lost when using alternatives such as projection screens driven by desktop computers. The Liveboard introduces such a digital whiteboard (Elrod, Bruce, Gold, Goldberg, Halasz, Janssen, Lee, McCall, Pederson, Pier, Tang & Welch 1992).

In an extension to the Liveboard, Pedersen et al. (1993) introduced Tivoli, which used gestures to issue commands while the pen was in gesture mode. The gesture mode, activated by a button on the side of the pen, allowed single stroke gestures to issue commands. The motivation behind using gestures was their availability to the user from any point on the board. In comparison, buttons would be located in a fixed position around the perimeter of the screen. When using such a large input area, the homing time required to reach these buttons would be extremely long, thus gestures provide a major benefit.

Damm, Hansen & Thomsen (2000) also used an electronic whiteboard coupled with gesture input to produce UML diagrams. The tool, Knight, used the marking menu approach for common commands. A

left flick was assigned to 'undo', and a right flick 'redo'. In a textual context, these mappings can also be considered iconic, as undo is a command to go back to a previous state, and redo is effectively going forward.

General Desktop Control

Sensiva is a system that allows the same gesture set to be used across all applications. Primarily developed for MS Windows, the system provides a set of initial gestures that can be customised by the user. The system acts on the context of the currently selected window, and tailors its gesture set appropriately. Possibly the most beneficial feature is the ability to launch applications by drawing a gesture anywhere on the screen. To activate MS Word, the user simply draws the letter 'W', for example, with the right mouse button depressed. The path of the cursor is highlighted by a blue line, which provides feedback as to the current gesture. Although not formally evaluated, the performance gains are real, as the application launcher prevents the need to traverse the many levels of the MS Windows 'Start Menu'.

Miscellaneous

Gesture systems have also been applied to applications that require frequently repeated commands. GRI-GRI (Chatty & Lecoanet 1996), French for scribble, was a system designed for air traffic control that used gestures drawn by a pen or finger tip as the sole means of input. Based on Rubine's classification algorithm (Rubine 1991b), the system required training for each gesture. This involved the user giving around fifteen samples for each gesture. The evaluation of GRIGRI involved the recording of 5000 marks, of which 5% were made out of context, 5% were interpretation failures (the system failed to recognise the command at all), and 1.3% were interpretation errors (the system thought the gesture meant something else). The evaluation also found users had a fast familiarisation with the system, possibly due to their initial training of the system. This familiarisation led to the system being used like pen and paper, and therefore users increased the speed of gestures and manipulations. From a user interface view this is good, although from a system view, this lead to higher rates of interpretation errors and failures.

Chapter 3

Design Issues

The information presented in Chapter 2 will now be discussed in relation to the design issues of a flick gesture system.

3.1 Successful Gesture Applications

Chapter 2 presented a selection of applications that have used gesture systems as a means of interface control. The following is a summary of the situations where gestures can be of benefit.

Input Device Restrictions

Gesture based interfaces increase the efficiency of user input, particularly in mobile computing where standard input devices such as the keyboard and mouse are impractical. In this situation, the pen is often the only input device available. Pen gesture systems allow users to control these devices and also allow text entry using gesture sets such as Unistrokes and Graffiti.

Efficient Use of Screen Real-estate

Applications such as Maya, a graphical editing tool, and Black & White, a role playing game, possess a large number of controls that can be displayed on screen. As identified by Kurtenbach et al. (1997), visually displaying these controls by way of buttons can reduce the ability of the user to focus on the underlying task. This was also relevant in Black & White, as initial designs of the game using icons meant little space was available to display the game itself. In these situations, gestures reduce the number of visible controls, allowing more effective and efficient use of screen real-estate.

Unistroke text entry also utilises the space saving benefits of gestures. Unistroke input only requires space for a single character, as each character effectively overwrites those previous. This not only reduces the space needed for text entry, but also allows "heads-up writing" (Goldberg & Richardson 1993), as the user is not required to relocate the stylus across the page.

Shortcuts for Controls

To select an icon, the user must shift focus from the working position to that of the button's location. Digital whiteboards show this movement to be very inefficient. Early digital whiteboards used virtual buttons located at the bottom of the board. Selecting one of these buttons often required the user to physically move their hand by metres in order to select the control. The use of gestures in this situation allows command shortcuts to be issued from the current working location, by way of a simple flick, thus saving targeting time and physical effort. This example also translates to the computer environment. Frequently repeated actions commonly involve the Fitts' Law cost of time-to-target. Flick gestures allow the user to bypass this targeting phase, which can potentially save hours of user time (Kurtenbach & Buxton 1994).

3.2   Button Overloading

When designing a gesture system for use with the mouse, a modifier is required to indicate the mouse is in gesture mode. To assign the gesture command to one of the mouse buttons results in an overloading of commands. In the standard WIMP environment, the most commonly used mouse buttons are the left and right. This can be attributed to the two-button mouse, which has been the standard for many years. Over the past few years, the three-button mouse has gained prominence, although the middle button is somewhat under-utilised in comparison to the others. Due to the universal nature of the two-button mouse, the middle mouse will not feature as part of our discussion.

The traditional tasks assigned to the left and right mouse buttons can be classified into one of two groups depending on the physical actions they require:

1. **Point, click and release (PCR)**. This group is characterised by locating an item on the screen by moving the mouse pointer, then making a stationary click and release. The click and release is generally very quick.

   - Left Button
     - Button activation/Icon selection — A single mouse click allows the user to activate a button, or select an icon (a double click will generally activate an icon's function).
   - Right Button
     - Menu Pop-up — A right mouse click will generally result in the display of a contextual menu.

2. **Point, click, drag and release (PCDR)** . This group is characterised by locating the cursor over an item, pressing the mouse button, and a subsequent move or "drag" of the mouse. When the position of the mouse is in the desired location, the mouse button is released. This task requires a degree of accuracy while the mouse button is depressed, which results in the time taken to complete the command increasing compared to the PCR group.

   - Left Button
     - Dragging (text selection or icon manipulation) — Text is selected from the pointer's position on mouse down, to the pointer's position on mouse up. Icon manipulation involves similar mechanics; the icon is located, then selected with a mouse down, followed by a "drag" then mouse up at the desired location.
   - Right Button
     - There are generally no functions assigned to the right mouse button in the PCDR group.

The physical characteristics of a simple flick gesture are somewhat similar to the PCDR group. The gesture requires a mouse down with a quick movement in a certain direction, followed by a mouse release. This poses potential problems when implementing flicks in a general purpose system, due to the potential conflict between the gesture command and the PCDR group.

The major difference between the flick and the PCDR group is the accuracy of the drag movement. Traditional dragging tasks require a level of precision to select a portion of text, or place an icon in the desired position. In comparison, the flick requires a relatively imprecise movement. The user simply flicks the pointer in the general direction of the target. It therefore seems reasonable to suspect that the time taken to issue a command in the PCDR group will be longer than that required to issue a flick.

If the time distributions do not overlap such as those portrayed in Figure 3.1(a), a gesture recognition system could distinguish between dragging tasks and flick gestures by examining the time taken to complete the task. Actions lasting longer than a specified threshold would be classified as dragging actions, and those lasting less than the threshold would be classified as flick gestures. If, however, the distributions overlap significantly (see Figure 3.1(b)), time could not be used to distinguish between each action. In this case an alternative to the left mouse button would need to be found. Chapter 4 looks to establish measures for dragging tasks and natural flick gestures to determine the feasibility of this theory.

(a) The two distributions do not overlap   (b) The two distributions are indistinguishable
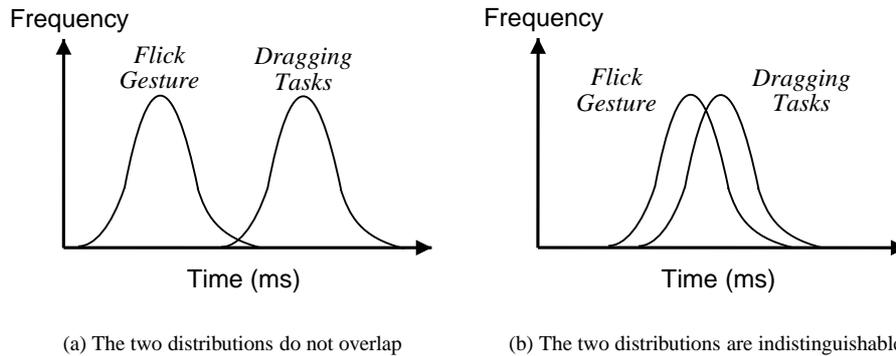
Figure 3.1: Two possible distributions for the mean time taken to issue a flick gesture, and the mean time to perform traditional dragging tasks. If these distributions do not overlap, the flick gesture could be overloaded with dragging tasks.

3.3   Gesture Learning and Usability

When designing gesture systems, a major factor is the ability of users to learn the gesture set provided by the system. If the gesture set is complicated, inexperienced users are less likely to use the system. Simple gesture systems can avert this learning difficulty by way of marking menus (see Section 2.1) which provide visual depictions of gesture directions on hesitation. In gesture systems that involve larger gesture sets, marking menus are inappropriate.

There are three possible methods to aid the learning of extensive gesture systems:

- Gesture Set Design — Iconic mappings between the shape of the gesture and the intended function aid both learning and memorability (Long, Landay & Rowe 1999, Long, Landay, Rowe & Michiels 2000). Unfortunately, not every function has an appropriate iconic mapping.

- Gesture Feedback — As shown by Tapia & Kurtenbach (1995), gesture feedback can aid user learning. An ink trail that follows the path of the user's gesture can then be 'snapped' to the idealised shape when the gesture is recognised. This enforces the users perception of the ideal gesture.

- User Gesture Definition — Rubine (1991a) suggested that gestures systems should be trained by the people that use them. This allows the gesture set to be tailored to that person, which also helps the learning of the gesture set for that person. The downside of this technique is the time required for the user to train the system. Also, the system can generally only be used by one person efficiently.

These issues can also be applied to the design of natural flick gesture systems.

Flick Gesture Systems

The flick gesture presents an attractive alternative to traditional interface controls such as buttons and linear menus. The user's target when selecting a button or menu item is a rectangular area, defined by the boundaries of the control. In comparison, the target when issuing a flick gesture is simply a single direction. This means the flick gesture effectively bypasses the time-to-target constraints of Fitts' Law, as accurate placement of the cursor is not required.

Due to the simple nature of the flick gesture, recognition is relatively straight forward. Each gesture is comprised of a straight line, therefore a recognition system need only record the beginning and end coordinates of a gesture. To aid the usability of flick gestures, however, an effective flick gesture system would be tailored to the 'natural' properties of flicks. There has been little research reporting these natural properties. The following chapter presents such an evaluation, and also looks to provide a means to differentiate standard interface actions, and those intended to be gestures.

Chapter 4

Establishing Quantitative Metrics

This chapter presents the results of two evaluations. The first investigates timing issues in relation to typical text selection tasks. The evaluation determines the lower-bound time limit for common mouse 'dragging' actions. The second evaluates the properties of natural flick gestures by way of user evaluation. These properties include gesture magnitude, gesture duration and angular errors for each direction. The results of this chapter are intended as guidelines for the design of pen and mouse based flick gesture systems.

4.1   Text Selection Evaluation

A natural implementation of a flick gesture system would use the left mouse button as the means to issue the flick. As stated in Section 3.2, the flick gesture is very similar to that of the Point, Click, Drag and Release (PCDR) group. Traditionally, the left mouse button has been used to issue PCDR actions, therefore assigning the flick gesture to the left button would theoretically be easier for users to learn.

By assigning the flick to the left mouse button, there are potential task conflicts, primarily those involving dragging tasks. As mentioned in Section 3.2, if the time to create a flick is different from the time to complete common dragging tasks, a time based distinction could be used to classify each action. Text selection is a frequent dragging task. The following evaluation uses text selection to determine a lower-bound of common dragging tasks. This helps to determine if overloading the left mouse button with the flick gesture is feasible.

There has been little work evaluating the time taken to select a portion of text. Card, English & Burr (1978) conducted an evaluation comparing expert performance of text selection using four different devices: a mouse, a joystick, step keys (or arrow keys), and text keys (such as 'home' and 'end'). The study measured the time required to home the hands to the device starting from the space bar, and the time required to position the cursor at the appropriate position in the text. Although the mouse was found to produce the least number of errors, and required the least overall time, the time measurement did not include the time required to actually select a portion of text, just the homing and positioning times.

Gillan, Holden, Adam, Rudisill & Magee (1990) examined the relationship between Fitts' Law and elementary dragging tasks such as text selection. The paper reported that text selection tasks were quickest when the dragging distance was at its smallest; 0.5cm, or a single character. The mean time for such a selection was approximately 600ms. The study did not report the minimum time taken to select a single character, which is the most important aspect when designing a flick gesture system.

The primary focus of the Gillan et al. (1990) paper was to analyse dragging tasks in relationship to Fitts' Law, whereas the primary focus of our evaluation is to determine the minimum time to select an item of text.
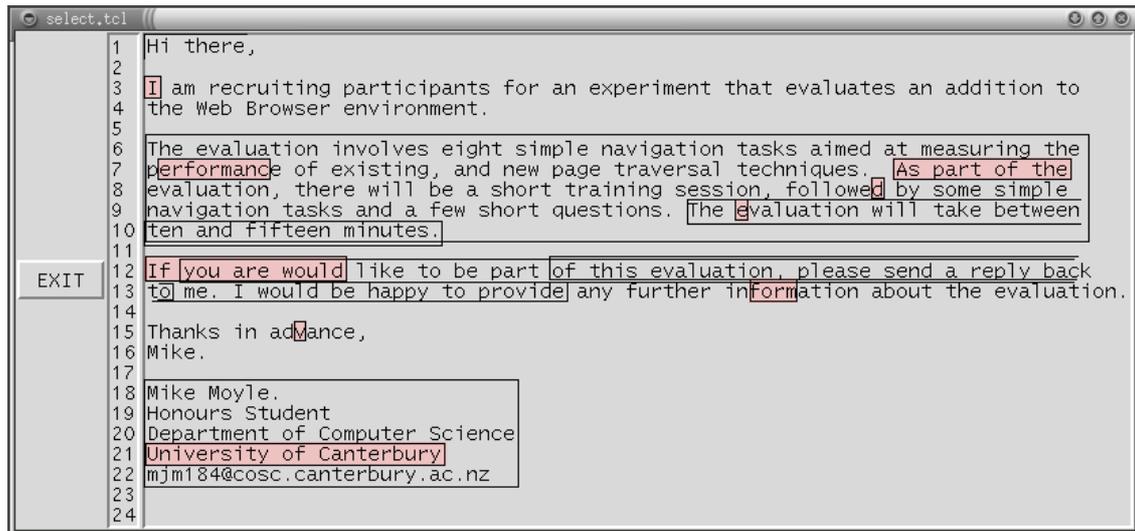
Figure 4.1: The text presented to each subject, with each task highlighted by a rectangular box.

4.1.1 Method

Subjects

Seven right-handed, post-graduate Computer Science students participated in the study. Due to their substantial computer experience, and the fact that textual selection is a common task, the evaluation results should err towards the lower-bound timing of expert performance.

Procedure

Subjects were presented with a window of text with line numbers to the left, as shown in Figure 4.1. The textual content of the window was the same throughout the evaluation. For training, each subject was asked to select five items from different portions of the text to ensure they felt comfortable with the required task. The users were then told that the evaluation was measuring the time required to select a portion of text, and if an incorrect selection was made they would not be penalised.

Each subject was then asked to select fourteen different pieces of text, ranging from a single character to an entire paragraph. Each task was verbally presented to the subject. They were then asked to point with their finger at the location of the task on the screen. Each task was given in sequential order, with the time and direction measured via an electronic log from mouse down to mouse up. If the selection was incorrect, the time was not recorded and the user was asked to repeat the selection until a successful selection was recorded. The primary purpose of the evaluation was to measure the minimum time required to select an item, therefore errors were not measured.

Apparatus

The experiment was run on a 32x24cm display with a 1280x1024 pixel resolution. The mouse was set to an acceleration factor of two-to-one, with a threshold of four, which is the default in the XWindows environment. A Tcl/Tk program was used to measure the time from mouse down to mouse up. The program also computed the direction of the selection from observed changes in the (x,y) coordinates of the mouse pointer.

Group	Task	Description
Single Letters	1	'v' in "advance" on line 15.
	2	'd' in "followed" on line 8.
	3	'e' in "evaluation" on line 9.
	4	'I' on line 3 without the space.
Multi-line word group	5	The last sentence of the second paragraph, starting on line 9.
	6	From "you" on line 12 to the first "to" on line 13.
	7	From "of" on line 12 to the "provide" on line 13.
Paragraphs	8	The signature paragraph from lines 18 to 22.
	9	The second paragraph that starts on line 6.
Middle of word	10	"form" from "information" on line 13.
	11	The word "performance" less the first 'p' and last 'e' on line 7.
Word Group – Left	12	"If you are would" on line 13.
	13	"University of Canterbury" on line 21.
Word Group – Right	14	"As part of the" on line 7.

Table 4.1: The selection tasks for each subject.

Design

The focus of the evaluation was to determine the minimum time from mouse down to mouse up, therefore tasks that involved complicated selections of text were not included. Table 4.1 lists the selection tasks subjects were asked to complete. Figure 4.1 shows these items highlighted by rectangles.

Each selection task was chosen to reflect a different aspect of text selection. As an example, tasks 1, 2, 3 and 4 represent the selection of single letters. Within this group, each task represents a different position in the text for a single letter. Task 1 represents selection of a single character in the middle of a word, task 2 at the end of a word, task 3 at the beginning of a word, and task 4 a single letter by itself. Although the task list was not exhaustive, it was felt that these tasks would produce the quickest selection times, and as mentioned earlier, tasks that would clearly take longer were not included.

Each selection task was then grouped with similar tasks as shown in Table 4.1. It is reasonable to hypothesise that selection tasks consisting of a single letters will produce the quickest selection times. This considered, Gillan et al. (1990) pointed out that the horizontal size of the text is not the only determining factor. The target size, or the area within which the mouse must stop to make a valid selection, must also be considered. Taking this into consideration, one could predict that tasks selecting items from the middle of words will be the hardest to perform as the effective target to make a valid selection is only a few millimetres wide.

The direction each item was selected was also recorded to allow us to determine if selection tasks were performed in a preferred direction.

It should be noted that some text environments allow a word (or line) to be selected by clicking that word (or line) multiple times. This type of selection task was not evaluated as the focus of the evaluation was to measure mouse 'dragging' movements rather than stationary clicks.

4.1.2   Results

All tasks were successfully completed by subjects with minimal delay. Users also commented that the system behaved as expected, and had no difficulty using the interface. The overall mean for the text selection tasks was 1054ms (s.d. 354.8), with a minimum time of 294ms, and a maximum selection time of 2682ms. Table 4.2 summarises the results for each of the six different task groups.

Rank	Group	Mean Time	Minimum Time
1	Single Letters	581ms	294ms
2	Word Group - Right	1201ms	545ms
3	Multi-line word group	1217ms	539ms
5	Paragraphs	1283ms	628ms
4	Word Group - Left	1290ms	473ms
6	Middle of word	1295ms	718ms

Table 4.2: The mean and minimum times for each of the groups, ranked in order of increasing mean time.
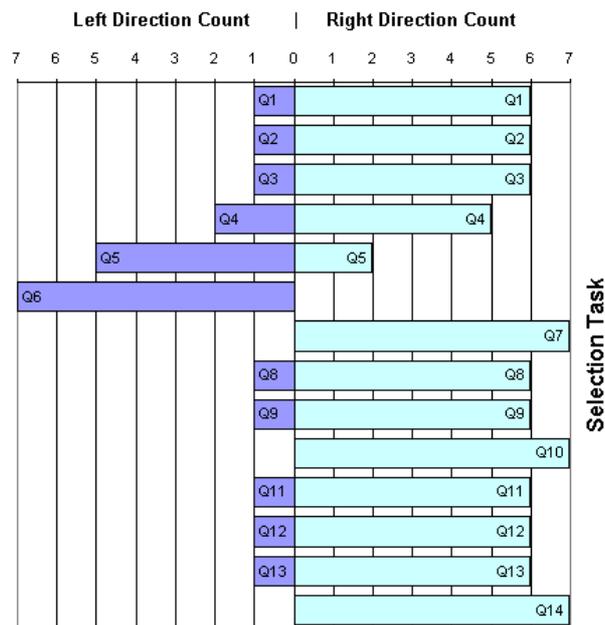

Figure 4.2: The direction that each task was selected.

As predicted, single letters were selected the quickest, and the 'middle of words' group the longest. Although many of the single letter tasks required precise targeting, the short distance helped to offset the cost of targeting.

Figure 4.2 presents a summary of the drag direction for each selection task. The graph shows a dominance of selections made in the right direction. This can be attributed to people selecting text as they read, which is left to right. Task 5 and 6, however, show most selections were made in the left direction. Interestingly, this can also be attributed to subjects selecting text as they read. The sentence selected for task 6 starts begins at the word "you" on line 12 of Figure 4.1. The sentence finishes to the left of this point at the word "to" on line 13. Therefore, to select the text as it would be read, the task would be selected in the left direction. Similarly, the text selected for task 5 started to the right of where it ended (last sentence of line 9 in Figure 4.1). This explains the left direction dominating the selections for these two tasks.

4.1.3 Summary

Overall, the minimum time taken to select a single letter of 294ms, and the mean time for single letter selection was 581ms. The direction of text selection was heavily influenced by the direction subjects read the text.

4.2   Natural Gesturing

The flick gesture is quick and 'sloppy' by nature. An effective implementation would allow users to create flick gestures naturally, thus gaining the full benefit of the flick gesture's quickness. The properties of natural gestures, therefore, are an important aspect in the design of a flick gesture system. If the user can create a flick naturally, the system is more likely to be learnable and efficient.

The following evaluation investigates the naturalistic properties of the flick gesture. The aim is to tailor the system to user behaviour, rather than the users modifying their behaviour to match the recognition system.

The evaluation measures three properties in relation to mouse and pen flick gestures issued in the left, right, up and down directions:

- Gesture Magnitude — Size of the gesture.

- Gesture Duration — Time required to issue a flick.

- Angular Error — Direction of the created flick compared to the optimal angle.

4.2.1   Motivation

Implementing software to recognise linear flick gestures is straight forward. The single directional motion of the flick gesture means a recognition system need only record the beginning and end coordinates of a gesture to determine the intended direction. Although direction detection is relatively simple, the properties of natural flick gestures, such as distance, timing and angular error can influence the design of an implementation. The aim of this evaluation is to determine sensible constraints on the magnitude, timing and angular error of flick gestures. These constraints can then be applied to recognition software.

Directional Properties

Although Dulberg et al. (1999) mentioned an overall median error of 5.73 degrees, there was no evaluation specifically reporting the errors for each direction. Off-axis targets (where the direction of the flick does not lie on either the X or Y axis) were reported as producing a noticeably greater variance than on-axis flicks. However, we are unaware of any studies reporting a direct comparison between the different directions of flicks, and the errors they produce.

Mouse Acceleration

We are also unaware of any studies comparing the influence of mouse acceleration on flick gestures. Most research has focused on dragging tasks with the mouse such as MacKenzie, Sellen & Buxton (1991) who found the mouse to be slower than the stylus when dragging, although produced fewer errors. Although the flick shares many of the mechanics of dragging, the flick does not require the final targeting of dragging which has a major influence on execution time. We will evaluate the effect acceleration has on the accuracy of the resultant flicks.

The visual feedback given to the user when performing tasks could also have a bearing on the size and time taken to create a flick gesture. Inherently, mouse acceleration will affect this visual feedback, hence we will look to analyse the effects of acceleration on the size and time taken to make a flick.

Mouse versus Pen

There have been many studies comparing the performance of the pen and the mouse for various activities, both in terms of time and errors. Kurtenbach et al. (1993) presented such a study, and found no significant difference between the pen and mouse in terms of time and errors when using marking techniques. Dulberg et al. (1999), as stated above, reported the median error for flicks in arbitrary directions using the mouse. They did not, however, provide a detailed analysis for each direction. We are unaware of any studies directly comparing the mouse and pen and the errors they produce in each direction. We will focus on flicks made in the on-axis directions, also known as the 'compass4' (Kurtenbach & Buxton 1993) layout.

4.2.2 Mouse Acceleration

Before presenting our evaluation, it is necessary to summarise some of the low-level properties of mouse operation.

Mouse motion is normally controlled by one of two user-configurable parameters that determine the mapping between the physical movement of the mouse and the corresponding movement on the screen. These values are normally termed 'acceleration' and 'threshold'. When the mouse is moved slowly, a base mapping between physical mouse-motion and cursor movement applies. Normally the default value for base movement is approximately four to one, meaning that for every four centimetres the cursor moves on screen, the physical mouse moves one centimetre.

The acceleration setting determines the maximum mapping between mouse movement and screen distance. This mapping applies during rapid mouse movement, and the normal default value is approximately double the base value, meaning that during rapid motion, each centimetre of mouse motion causes the cursor to move approximately 8cm.

The threshold value determines the mouse-movement rate (distance per unit time) that must be reached before the accelerated mouse mapping applies.

For information about the XWindows implementation of mouse acceleration, an extract of the `xset` man page is provided in Appendix B.

4.2.3 Method

Subjects

Twenty-four right handed post-graduate Computer Science students participated in the evaluation. All had no previous experience using pen based systems. Although these subjects have a substantial computer background, we believe their ability to issue gestures will not be significantly different from that of other users.

Each subject was assigned to one of three gesture input systems:

- **Mouse input, acceleration** — Gestures were issued using the default XWindows[1] mouse acceleration setting of two-to-one, with a threshold of four. The two-to-one setting results in an approximate eight-to-one mapping between physical movement and cursor motion.
- **Mouse input, no acceleration** — Gestures were issued using a constant linear mapping between physical mouse movement and the resultant pointer movement on screen. The linear setting results in a constant four-to-one mapping between physical movement and cursor motion.
- **Pen input** — Gestures were created using a pressure sensitive pen computer, which inherently has a direct mapping of one-to-one between physical movement and the resultant gesture.

Procedure

Subjects were informed of our intention to evaluate the natural properties of flick gestures. This was followed by a verbal description of the flick gesture. Mouse users were told that "flicks are quick motions with the mouse in a desired direction, during which the mouse button is pressed then released". Pen users were told that "flicks are quick motions with the pen in a single direction, similar to making a flick with a biro on a piece of paper". Pen gestures do not require the stylus button to be pressed. Subjects were also informed that the evaluation involved issuing gestures in four directions, left, right, up and down.

Subjects were then asked to practice making flick gestures in each of the four directions until they felt comfortable with the system. This was followed with each user issuing flick gestures grouped by direction. Initially, each subject completed 50 gestures in each of the four directions. This was later reduced to 25 flicks in each direction, as some users mentioned they felt repetitive strain problems in their wrists and hands. The completion of 25 or 50 gestures gave 100 or 200 gestures for each user in all directions for a particular input setup. The reduction in number of gestures has a minimal impact on our data analysis, as only one mean sample in each direction is calculated per user.

---

[1]These settings are invoked using the `xset` command.

Direction	Mouse Button		Mouse Acceleration		Gesture Input Device	
	Left	Right	Zero	Two-to-one	Mouse	Pen
Left	S1-8	S1-8	S9-16	S1-8	S9-16	S17-24
Right	S1-8	S1-8	S9-16	S1-8	S9-16	S17-24
Up	S1-8	S1-8	S9-16	S1-8	S9-16	S17-24
Down	S1-8	S1-8	S9-16	S1-8	S9-16	S17-24
	(a) Experiment one		(b) Experiment two		(c) Experiment three	

Table 4.3: Experimental design for Experiments one, two and three, showing how subjects (S*n* for subject number) were assigned to conditions.

Although unfamiliar with the pen system, users had no problem issuing flicks with the pen. This can be attributed to the similarity of writing with pen and paper.

At the conclusion of the evaluation, subjects were asked to comment on the system they had just used. Each evaluation lasted approximately five to six minutes.

Apparatus

All gestures were issued using a large window of 500x400pixels, which meant users were not constrained by the boundaries of the window when issuing gestures. A program written in Tcl/Tk logged the time and coordinates at the beginning and end of each gesture, along with the gesture's intended direction.

A mouse gesture was recorded when the appropriate button was depressed, followed by the corresponding button release. The program was running under Solaris 8 using the Gnome windows environment.

A pen gesture was recorded when the pen touched the screen surface, and concluded when the pen left the surface. The program was running under Windows 1.0 for Pen Computing.

Experiments using the mouse were run using a 32x24cm display running at a 1280x1024pixel resolution, giving 40 horizontal pixels per centimetre. The pen experiments were run on a Toshiba T200/80 pen computer, with a 19x14.25cm display running at a 640x480 pixel resolution, giving 33.7 horizontal pixels per centimetre.

The number of pixels per centimetre allows us to translate the logged magnitude of gestures, which were logged in display pixels, to millimetre motions of the physical device. This millimetre measurement will give insight into how gesture creation differs at the physical device.

Design

The evaluation was primarily aimed at determining distance and timing values using different input devices, and in different directions. Three dependent variables were measured from the logged data as follows:

- **Gesture Magnitude** — The distance measured from the beginning to end of a flick gesture. Data values are recorded in pixel coordinates. This measure will help determine the typical size of a natural flick gesture.
- **Gesture Duration** — The duration measured in milliseconds from the beginning to the end of each flick gesture.
- **Angular Error** — The variation from the optimal flick angle measured in degrees. If the flick direction was to the left, the optimal angle is 90 degrees. If the actual angle of a particular flick is 95 degrees, the angular error is the absolute value of (95 - 90) = 5 degrees. This measure will show any biases for a particular direction.

The data was analysed in three different experimental designs, summarised in Table 4.3.

- **Experiment one: Mouse Button** — The first experiment examines three dependent variables, gesture magnitude, timing and angular error in a 2x4, repeated measures, analysis of variance (ANOVA). The first factor, 'mouse button', is a within-subjects factor with two levels; left or right mouse button. The second factor, 'gesture direction', is a within-subjects factor with four levels; left, right, up and down. Table 4.3(a) summarises the experimental design. This experiment will allow us to compare gestures created with the left and right mouse button across the four directions. As the primary aim of this evaluation is to determine constraints for a real implementation, the mouse with acceleration was used as it was felt this was the most common or default setting.

- **Experiment two: Mouse Acceleration** — The second experiment compares gestures created with a non-accelerating mouse and those created with a 'standard' two-to-one (threshold four) setting for mouse acceleration. The experiment is a mixed factors 2x4 ANOVA, summarised in Table 4.3(b). The first factor, 'mouse acceleration', is between-subjects factor with two levels; zero or two-to-one. The second factor, 'gesture direction', is a within subjects factor with four levels.

- **Experiment three: Input Device** — The third experiment compares gestures created with a non-accelerating mouse and those created using the pen. The experimental design is summarised in Table 4.3(c), and represents a mixed 2x4 ANOVA. The first factor, 'input device', is a between-subjects factor with two levels; non-accelerating mouse, and pen. The second factor, as before, is flick direction. This experiment allows us to compare the differences between issuing flicks with a pen and a mouse, and hence determine if a universal recognition system could be used for both the pen and the mouse.

4.2.4 Results

Overall

Subjects generally completed the full gesture sets extremely rapidly, typically taking around five to six minutes to complete 200 gestures and training. In an attempt to gather gesture information regarding natural flick gestures, subjects were given minimal instruction and training. With this considered, there was little variation in the magnitude, timing and angular error produced by each subject and between subjects.

Across the 6400 gesture set the mean gesture size was 117 (s.d. 59) pixels, the mean gesture time was 155 (s.d. 54) ms, and the mean angular error was 3.8 (s.d. 4.9) degrees. Figure 4.3 shows a distribution of
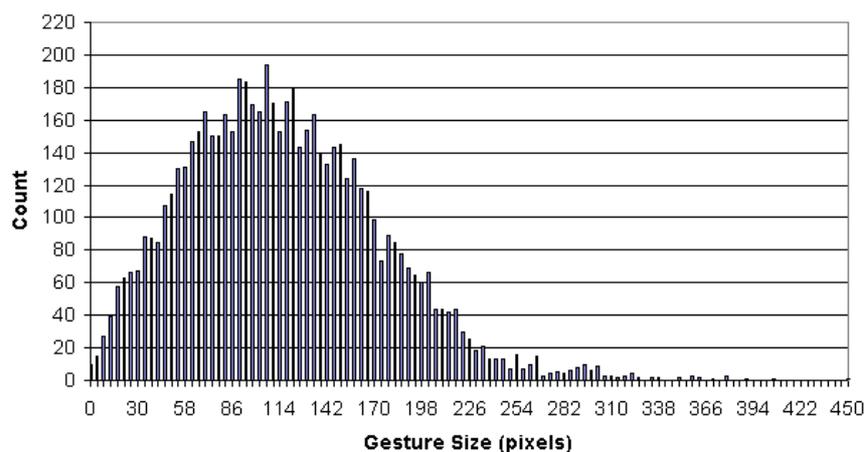

Figure 4.3: Distribution of gesture magnitude across all 6400 gestures, segregated into four pixel increments.

the gesture sizes across all devices, segregated into four pixel increments. The relatively normal shape of this graph was typical of the results gathered for each of the dependent variables.

Experiment One — Left versus Right Mouse Button

The overall mean distance for both buttons in all directions was 147 (s.d. 44) pixels. The mean distance for gestures using the left and right mouse buttons were 138 (s.d. 37) pixels and 156 (s.d. 49) pixels respectively. This did not give a significant difference: $F(1,7) = 1.00, p = 0.349$. The distances in each direction were also not reliably different: $F(3,21) = 1.36, p = 0.281$.

The overall mean time for both buttons was 162 (s.d. 45)ms. The mean times for gestures using the left (155ms s.d. 41) and right (167ms s.d. 49) mouse buttons were not reliably different: $F(1,7) = 0.215, p = 0.657$.

Interestingly, there is a significant difference between the mean mean times to create gestures in each of the four directions. Mean times were 153 (s.d. 44)ms, 160 (s.d. 46)ms, 156 (s.d. 40)ms, and 175 (s.d. 52)ms for the left, right, up and down directions respectively: $F(3,21) = 7.639, p < 0.01$. This issue is discussed further in experiment two.

The mean angular error for gestures issued using the mouse was 2.7 (s.d. 1.73) degrees. There were no significant differences between the angular means for the two mouse buttons, or between the angular errors for each direction.

These results indicate that the mouse button used to issue gestures has a negligible result on the performance of gesture input. The subjects' comments, however, revealed a preference for creating gestures with the left mouse button (discussed further in "Observations and User Comments").

Experiment Two — Mouse Acceleration

The overall mean distance for the factors zero acceleration and two-to-one acceleration was 128 (s.d. 37) pixels. The mean distance for the zero acceleration setting was 108 (s.d. 33) pixels, and the mean distance for the two-to-one setting was 147 (s.d. 31) pixels. As expected, this gave a significant difference: $F(1,14) = 6.1, p < 0.05$. There was no significant difference between gesture magnitude in each of the four directions.

The overall mean time to issue gestures using the mouse was similar to that of experiment one, and took 165 (s.d. 33) ms. There was no significant difference between the mean times taken to issue gestures using zero acceleration (169ms s.d. 37) and two-to-one (161ms s.d. 29) acceleration.
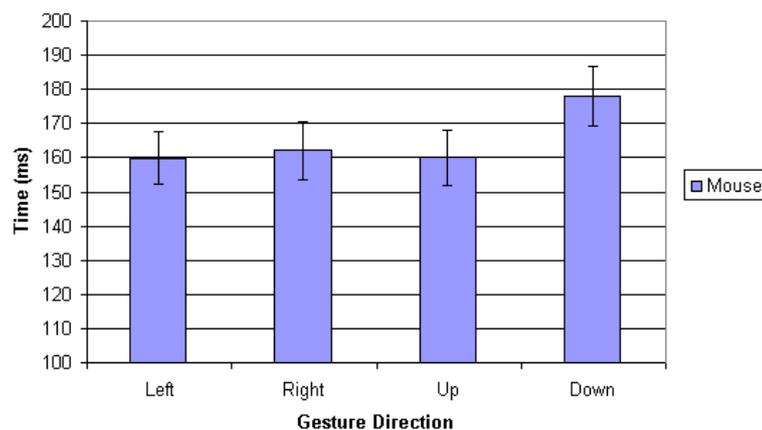

Figure 4.4: Mean time in milliseconds for all mouse gestures in each direction. Error bars show one standard error above and below the mean. Also note that the time scale begins at 100ms.
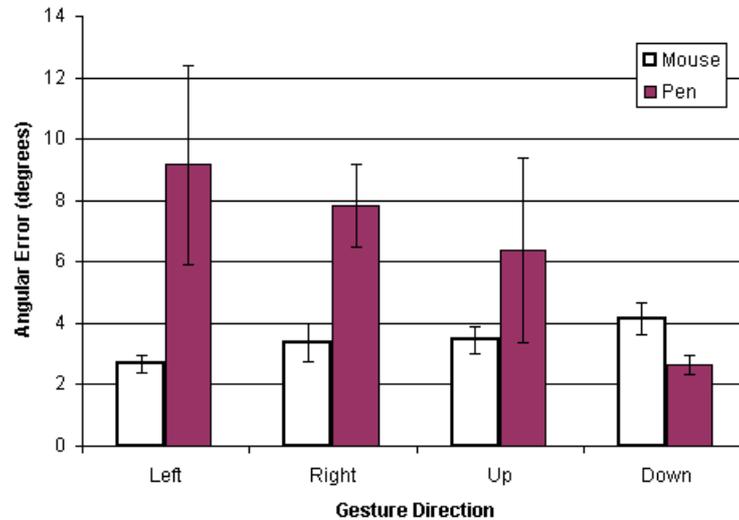
Figure 4.5: Mean deviation from the optimal angle in each direction.

The mean times for mouse gestures in each direction are presented in Figure 4.4. Like experiment one, there was a significant difference between the mean times in each of the four directions: $F(3,42) = 10.37, p < 0.001$. The mean times were 160 (s.d. 31) ms, 162 (s.d. 34) ms, 160 (s.d. 32) ms, and 178 (s.d. 34) ms for left, right, up and down. Surprisingly, the down gesture took approximately 11% longer than the others to issue.

The mean angular errors for the two acceleration factors (zero and two-to-one) were 3.4 (s.d. 1.41) degrees and 2.7 (s.d. 1.3) degrees. This does not give a significant difference: $F(1,14) = 3.026, p = 0.104$. There was no significant difference for the angular error in each direction.

As expected the accelerating mouse produces gestures longer than those of the non-accelerating mouse. The mean time, however, is not significantly different. This suggests that natural flick gestures do not rely on the visual feedback given by the mouse cursor.

Experiment Three — Pen versus Mouse Input Device

The mean distance for gestures issued using the pen was 64 (s.d. 32) pixels, and when compared to the mean for gestures using a non-accelerating mouse of 109 (s.d. 33) pixels provides a significant difference: $F(1,14) = 8.01, p < 0.05$. There was no significant difference between the magnitude of gestures issued in each of the four directions.

Translating the above figures into physical movement at the device rather than the screen shows distances of 19.0 (s.d. 8.65) mm and 6.8 (s.d. 2.06) mm for the pen[2] and mouse[3]. This is a reliable difference: $F(1,14) = 12.81, p < 0.01$. The gesture distances in the four directions were not reliably different.

Interestingly, although the mouse produces significantly greater screen distances compared to the pen (attributed to the four-to-one mapping of screen to mouse movement), the physical movement required when issuing natural flick gestures was just 36% of the distance when issued using the pen. This issue is further discussed under "Physical Characteristics of Gestures" in the following section.

---

[2]The mapping for the pen device was 1cm:1cm, or for every cm the pen moved, the cursor moved one cm on screen.
*(pixel movement)/ ((pixels per cm)*mapping) = physical distance (cm)*
(64 pixels) / ((33.7pixels per cm)*(1cm/1cm)) = 1.90cm = 19mm
[3]The mapping for the non-accelerating mouse was 4cm:1cm. For every 1cm moved by the mouse, the cursor moved 4cm.
*(pixel movement) / ((pixels per cm)*mapping) = physical distance (cm)*
(109pixels)/ ((40pixels per cm)*(4cm/1cm)) = 0.68cm = 6.8mm

(a) Pen Gestures   (b) Mouse Gestures (Direct)
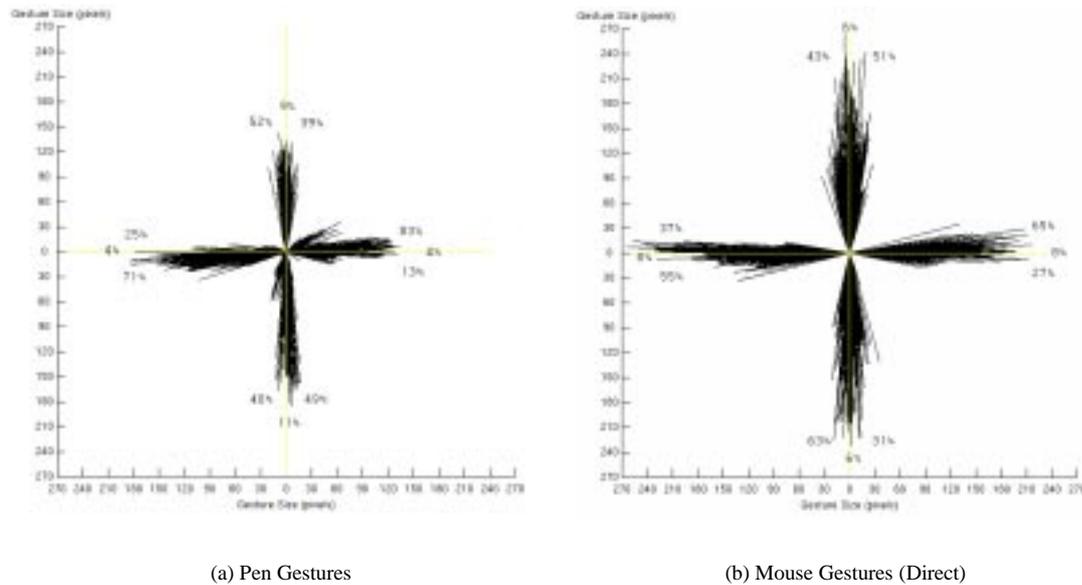
Figure 4.6: Magnitude and angular error of gestures in four directions, left, right up and down. Distances are reported in pixels. Percentage values indicate the bias above, equal to, or below optimal angle for a particular direction.

As the pen computer recorded times to the nearest 55ms, the timing information for the pen device was discarded. We therefore provide no comparison of timing interactions between devices.

When analysing the angular error for each device, an interesting result is revealed. The mean angular errors for the pen and non-accelerating mouse were 6.5 (s.d. 6.7) degrees and 3.4 (s.d. 1.4) degrees respectively. This was not a significant difference: $F(1,14) = 3.01, p = 0.105$. ANOVA also showed there to be no significant difference between the angular errors in each of the four gesture directions: $F(13,42) = 1.58, p = 0.209$. There is, however, a surprisingly significant interaction between the input device and direction: $F(3,42) = 3.572, p < 0.05$. Figure 4.5 clearly reveals the cause of this interaction. When comparing the angular errors issued in the left and right directions, the pen has a relatively high angular error compared to the small error for the mouse. Conversely, the angular error for the down gesture is relatively large using the mouse but small with the pen. This aspect is discussed further under "Physical Characteristics of Gestures" in the following section.

Figures 4.6(a) and 4.6(b) show plots of the flicks made with each device in each of the four directions. The percentages indicate the bias above, equal to, or below optimal angle for a particular direction. The values show that when using the pen to make horizontal (left or right) gestures, there is a strong tendency to err upwards on right gestures (83%), and a strong tendency to err downwards on left gestures (71%). The bias is in the same direction when using the mouse for horizontal gestures, although not of the same magnitude as the pen (55% for left below, and 65% for right above the optimal angle).

All of the subjects' were right-handed. We suspect that the direction of the angular bias partly depends on the orientation of the user's body to the device, and would be reversed for left-handers.

Observations and User Comments

Many subjects commented that gestures made with the left mouse button were much easier than those made with the right mouse button. All but one used their middle finger to create gestures with the right mouse button. The remaining subject placed his middle finger on the middle mouse button, and used his third finger to create gestures with the right mouse button. The index finger was always used on the left mouse button. The subjects' lack of experience dragging with the right mouse button is likely for the left button

23

preference.

When using the mouse, there was a strong preference for creating gestures in the horizontal (left and right) directions as opposed to the vertical directions (up and down). Several subjects mentioned that vertical (up and down) gestures were "no where near as natural". In comparing the left and right gestures with the mouse, those that expressed a preference preferred the left direction.

In contrast, pen users generally found the up and down direction to be more natural than left and right. These directional differences can be attributed to the physical characteristics of gesture generation, as discussed below.

Physical Characteristics of Gestures

The motor movements used to make gestures with each device help to explain some of the preference and performance differences observed.

Left and right gestures made using the mouse involved a lateral flexing of the wrist with almost no arm movement. In contrast, when using the pen, lateral and rotational movement was combined with small amounts of finger extension (left) or contraction (right).

Vertical gestures made using the mouse were made using two methods. The less commonly used technique was to move the whole arm with minimal movement in the fingers or wrist. More often, however, the subjects moved the mouse by extending (up) or contracting (down) their thumb and fourth/little fingers. With the pen, vertical gestures were made by simply extending and contracting the fingers and thumb.

The physical movement at the device level was much greater when using the pen than the mouse. This is unsurprising, as pen gestures begin as soon as the mouse makes contact with the display. The gesture size is directly equivalent to that created with a pencil on paper. With the mouse, however, the gesture magnitude is the subset of the physical mouse movement that occurs between the mouse-button being depressed and released (see Figure 4.7).

4.2.5   Summary

The results show that the mean distance and time of gestures created using the left and right mouse buttons were not significantly different. There was also no significant difference for the mean angular error in each direction. Users, however, preferred using the left mouse button to create gestures.

As expected, the distance of gestures created using an accelerating mouse were significantly different to those created using a non-accelerating mouse. The time and angular errors for both settings were not significantly different.

Gestures created with the mouse were comparable for the left, right and up directions, although they were 11% slower when issued in the down direction. Interestingly, although there was a significant difference for the time taken to create a gesture, there was no significant difference in angular error for each of
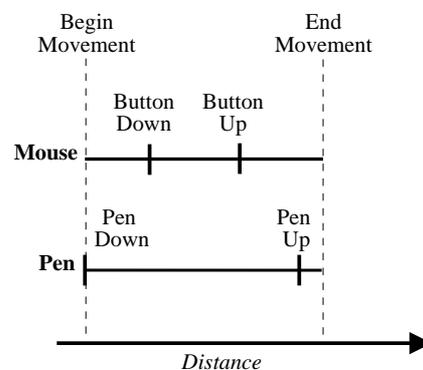

Figure 4.7: Summary of the physical movements made with the pen and the mouse.

the four directions using the mouse. This suggests that although the down gesture took longer to create, the down direction was no less accurate in terms of angular error compared to the other three directions.

Gestures created with the pen were less accurate than those created with the mouse. There was a strong bias downwards when issuing gestures in the left direction, and upwards when issuing gestures in the right direction. This bias was not nearly as pronounced with the mouse.

Pen gestures were significantly smaller in display pixels when compared to a non-accelerating mouse. This can be attributed to the mapping of cursor-to-physical movement. The mouse is mapped at four-to-one, whereas the pen has a direct mapping of one-to-one. Interestingly, however, the physical movement of each device showed the mouse to require only 36% of the physical movement required by the pen during gesture creation. This difference in physical movement can be attributed to mouse gestures recording only a subset of the mouse movement during mouse down. In comparison, a pen gesture records nearly the entire movement of the flick.

4.3   Metrics Established

From the results of the two evaluations presented in Sections 4.1 and 4.2 we have determined numerical values for both common dragging tasks and the natural flick gesture. The evaluation reported in Section 4.1 found the minimum time for common text selection tasks to be 294ms. The results of Section 4.2 found the mean time for flicks created with the left mouse button to be 155 (s.d. 41) ms. These results suggest that overloading the left mouse button to act as a gesture modifier is feasible. The time to complete each action can be used to distinguish flick gestures from standard dragging tasks.

Figure 4.8 shows an idealised representation of the results from the two evaluations. In the figure, q signifies the mean time for the flick gesture using the left mouse button, s represents the minimum time for dragging tasks, and r represents a possible position that a constraint could be placed to enable flick gestures to be distinguished from dragging tasks. Actions lasting longer than r would be classified as dragging tasks, while those lasting less than r would be classified as flick gestures. This gives a time gap, n, which represents a buffer between the minimum time taken to perform a dragging task, and the upper-limit imposed on flick gestures.
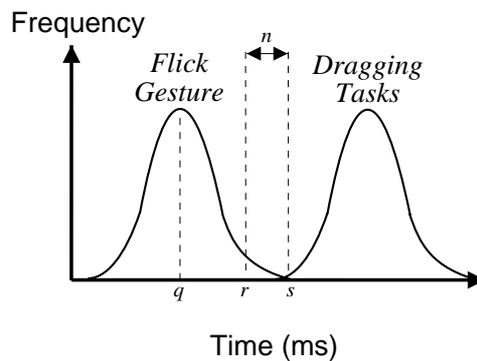

Figure 4.8: An idealised representation of the results found from the two evaluations, with q showing the mean time for flick gestures, s showing the minimum time found for dragging tasks, and r showing a possible position an upper bound could placed when creating flick gestures.

Chapter 5

Gesture Navigation System

This chapter presents an implementation of the flick gesture in a real world application. Using the quantitative metrics determined in the previous chapter, the flick gesture will be applied to the frequent user activity of web browsing.

5.1 Web Browsing

Web browsing, as discussed in Section 2.3.2, provides a logical test application for the flick gesture. The web browsing interface is primarily controlled using the mouse. This is emphasised by the number of user browsing actions that involve clicking on links, reported to be as high as 50% (Tauscher & Greenberg 1997).

Web-page revisitation is a common activity for users while browsing. Tauscher & Greenberg (1997) found that, on average, each user visited 58% of URLs more than once. A more recent study reported this to be as high as 81% (Cockburn & McKenzie 2001). Web browsing applications support many mechanisms for revisiting web pages, including 'favourites' (or bookmarks), history tools, and the back button. The back button is a common source of user actions. In Catledge and Pitkow's study, the back button accounted for 41% of all page requests. Tauscher & Greenberg (1997) also found heavy usage of 'back', with it accounting for 30% of commands. In contrast, the forward command was lightly used, accounting for 2% of user actions.

Figure 5.1(a) shows the common visualisation of the back and forward command. The association of the left and right arrow for the back and forward commands provides a logical iconic mapping to the left and right flick gestures.
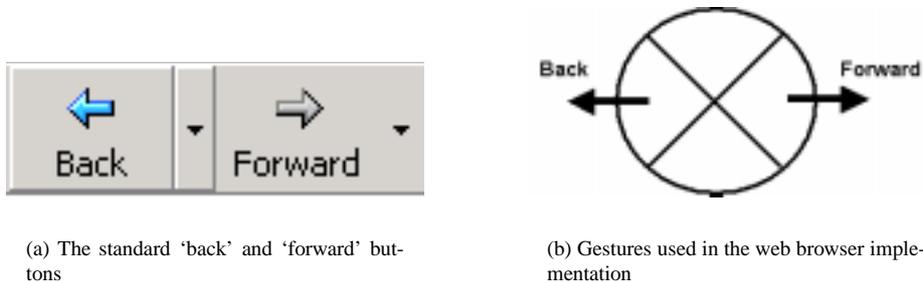

(a) The standard 'back' and 'forward' buttons

(b) Gestures used in the web browser implementation

Figure 5.1: The iconic nature of 'back' and 'forward'

5.2   Implementation Overview

As shown in Figure 5.1(b) four gesture regions were recognised, with two gesture commands being implemented in the browser environment, namely back and forward. The up and down gestures were unused in our system, as the back command itself accounted for over 40% of user actions while browsing. As mentioned earlier, the iconic mapping of flick to function was also a desirable property.

The addition of gesture recognition to an application can be achieved in one of two ways. First, intercept mouse actions before they reach the application. Second, intercept mouse actions after the application has processed them. The latter approach was chosen, due to its relative simplicity and effectiveness. The recognition itself was achieved by intercepting the mouse-events generated by the browser using `Javascript`. Although not the optimal solution, the system was very effective. If implemented in a commercial environment, gestures would be treated as first class actions, hence the software would intercept events before being processed by the browser.

The `Javascript` implementation allowed a simple way to achieve gesture recognition. The script itself was written as a stand-alone `HTML` file, and then placed on an IIS 5.0 web server. The script was then included as a footer to every page served by the server. This meant any page retrieved from that server effectively provided gesture recognition. This also meant that no additional changes were required at the browser level, ensuring the implementation of the gesture system was transparent to users. The source code for the script is presented in Appendix A.

An implementation of the flick gesture requires three parameters; size, duration and direction of the gesture. The following sections discuss how each of these properties were determined from the evaluation as reported in Section 4.2.

Mouse Button

The evaluation reported in Section 4.2 found users preferred creating flick gestures using the left mouse button. This presents some overloading issues such as those discussed in Section 3.2, primarily with link selection (which can be likened to button activation), and text selection. Both of these actions are completed using the left mouse button. Although dragging with the right mouse button does not present such overloading issues, the `Javascript` implementation was unable to use the right mouse button to create gestures as Internet Explorer displays the right mouse button context menu after all right mouse button events.

Gesture Size

To prevent the script from recognising a simple mouse click as a gesture command, the mouse coordinates must change by a minimum of 35 pixels. As discussed in Section 3.2, button activation (link selection in the browser) involves a mouse down with very little movement of mouse itself. In comparison, a flick gesture is generally associated with a mouse down, followed by movement in a single direction. To differentiate between the two actions, a minimum gesture size was imposed.

The results of Section 4.2.4 suggest the mean size of a flick using the left mouse button using the default acceleration setting is 138 (s.d. 37) pixels. This influenced the decision to set the minimum gesture size at 35 pixels. This value is over 2.78 standard deviations away from the mean ($(138-35)/37 = 2.78$). As there is no upper bound on the size of a gesture, this means over 99% of gestures will be valid ($Z-value(2.78) = 0.994$).

Gesture Duration

The left mouse button is also frequently used to select and highlight text on web pages. As discussed in Section 4.1, text selection is generally a slow and deliberate task. In comparison, the flick gesture is quick and inaccurate. To differentiate between the two actions, a maximum gesture time of 250ms was imposed.

The results of Sections' 4.1 and 4.2 were used in determining the upper bound of 250ms. The minimum task completion time in the text selection evaluation was 294ms from 98 trials. From the natural gesturing evaluation, the mean flick duration using the left mouse button was 155 (s.d. 41) ms. In order
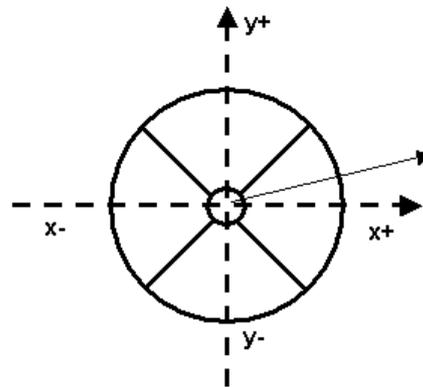
Figure 5.2: The four sectors of the gesture implementation. In this example, a gesture has been made in the right direction.

to satisfy these requirements, a time of 250ms was imposed as the upper limit for a flick gesture. This value is 2.32 standard deviations away from the mean $((250 - 155)/41 = 2.32)$. This allows approximately 99% of gestures generated from the natural gesturing evaluation to fall within the time restriction $(Z - value(2.32) = 0.989)$. This also provides a buffer of 44ms between the minimum text selection time, and that of the gesture timeout.

Gesture Direction

The gesture system was setup to recognise four different flick directions, left, right, up and down (although the up and down were not assigned a function). To determine the direction of the gesture, the absolute distances travelled in the x and y direction were measured. This determined if the movement was horizontal or vertical. If the movement was horizontal, the signed distance in the x direction was measured. If this value was negative, the direction was deemed to be left, and if the value was positive, the direction was deemed to be to the right.

Figure 5.2 shows the four sectors of the gesture implementation. In this example, the cursor has travelled the greatest distance in the x axis, and the x change is positive, therefore this movement would be recognised as a right flick gesture.

Design Compromises

The use of `Javascript` enforced some undesirable design compromises. First, gestures are unable to be recognised when initiated over an image or a link. In this situation, the context of the cursor changes, and the subsequent mouse down event is unable to be caught using our `Javascript` implementation. Second, when a gesture is initiated over an area of text, dragging the mouse to issue a flick also selects that portion of the text. Although the flick gesture is still recognised, this has the potential to confuse users. These issues arise as our implementation does not treat gestures as first class actions. In a commercial implementation, however, these issues would be easily solved.

Chapter 6

Evaluating Gesture Navigation

The system proposed in Chapter 5 was evaluated to determine the overall effectiveness of flick gestures when accessing common controls. Flick gestures do not incur the targeting time required by Fitts' Law, and thus provide a quick alternative to button and menu selection techniques. Results from the evaluation show that subjects were able to navigate significantly faster using the gesture system. Furthermore, the subjects were extremely enthusiastic about the technique, with many expressing their wish that "all browsers should support this". Subjects also found the flick gesture easy to learn.

6.1 Motivation

The effectiveness of a flick gesture system to activate common controls can be measured in three ways; empirical results derived from a user evaluation, subjective measures and user comments, and the learnability of the system. By applying the flick gesture to a common application, we will measure each of these items to determine the overall effectiveness of the gesture system.

As mentioned in Section 2.2, Dulberg et al. (1999) conducted an informal evaluation of flick gestures in a real system. User feedback for the Microsoft Windows Desktop application was positive, with five from six participants saying they would use it if available. As part of the paper, Dulberg et al. (1999) also compared flick gestures with normal button clicks and keyboard shortcuts. The drawback of this evaluation was the artificial nature of the system used. Our evaluation looks to address this issue, by evaluating the flick gesture in a real world application.

6.2 Method

Using the system proposed in Chapter 5, an evaluation was conducted to determine the effectiveness of the flick gesture in a real application. Although the Opera (v5.11) browser does support gestural navigation, the current user base is extremely small in comparison to that of Internet Explorer and Netscape Navigator. Also, the screen layout and functionality differs to that of the major browsers. As our subjects were unfamiliar with the Opera layout, the browser was discarded as a possible experimental platform.

Subjects

The twenty subjects were all volunteer postgraduate Computer Science students familiar with web navigation. Although this group has more experience with mouse-use than most others, we do not believe that their motor skills in generating flick gestures would be significantly different to other groups.

Apparatus

The evaluation was conducted using Internet Explorer version 5.0 running on Windows 2000. The browser window was sized at 1152x864 pixels on a 32x24cm display running at a 1280x1024 pixel resolution. As

discussed in Chapter 5, the gesture system was implemented using Javascript, which meant the appearance of the browser was unmodified.

All pages used in the evaluation were stored in the browser's cache before beginning each experiment. This ensured navigation times were not influenced by network conditions. Task completion times were measured using a stopwatch, by the same experimenter throughout the evaluation.

The web pages used in the evaluation were taken from the University of Canterbury web site, intentionally chosen for their familiarity to subjects who were all students at the university. All pages were of a similar format, as shown in Figure 6.1, with a banner across the top, and an index menu to the left.

Procedure

The experimental procedure was consistent across all tasks completed in the evaluation. For each task, the user was shown the precise path through a selection of pages predetermined for each experiment. They were then required to rehearse the path through that set of pages at least twice until they felt comfortable with the task. Subjects then followed the exact same path "as quickly as possible" using the normal back button with the time to taken to complete the task recorded. After completing all tasks using the back button, subjects were then exposed to the gesture interface. As before, subjects were asked to rehearse each navigation task at least twice, this time using the gesture interface. Then, as quickly as possible, subjects traversed the same sets of pages using the gesture interface. It was intended that the rehearsal of the navigation tasks prior to each evaluation would minimise the impact of learning effects. Essentially we were measuring expert performance.

Each evaluation lasted approximately twenty minutes.

Design

All subjects participated in two web browsing experiments, both based around common types of back-tracking behaviour; depth-first search and back, and breadth-first 'hub-and-spoke' browsing (Catledge & Pitkow 1995).

- **Experiment One** — Experiment one examined the effectiveness of the two interfaces in a depth-first navigation. Figure 6.2(a) shows the path followed by each subject, that involved following four links on subsequent pages and then navigating back to the start page.

  There are two reasons for hypothesising that the gesture system might not provide significant performance benefits in this experiment. First, when using the back button the user need only make one positioning movement, thus minimising the time-to-target overhead of Fitts' Law. Having moved the cursor to the back button, the user can simply click the back button four times, without the need to
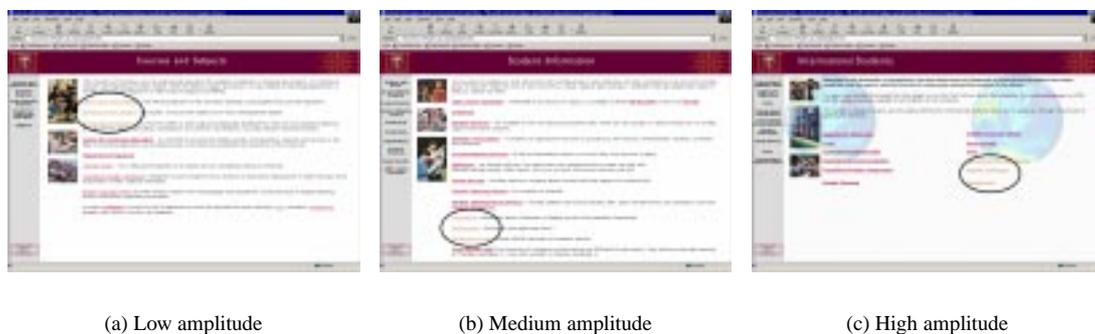

(a) Low amplitude   (b) Medium amplitude   (c) High amplitude

Figure 6.1: The three 'hub' pages used in experiment two. The oval indicates the position of links used on that page.

```
Start Page                        Start Page
   ⇨follow link                       ⇨Follow link to hub page
       ⇨follow link                       ⇨Follow link to spoke page
          ⇨follow link                    ⇦Back to hub page
              ⇨follow link                ⇨Follow link to spoke page
              ⇦Back                       ⇦Back to hub page
           ⇦Back                          ⇨Follow link to spoke page
        ⇦Back                             ⇦Back to hub page
     ⇦Back                          ⇦Back
Start page                        Start Page
```

(a) Depth-first navigation as used in experi-   (b) Breadth-first 'hub-and-spoke' navigation
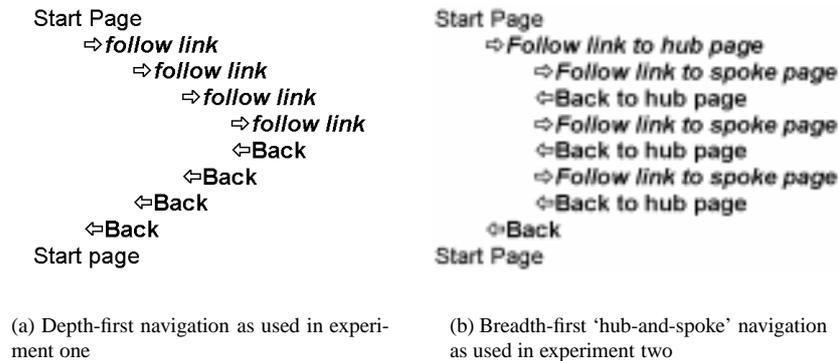ment one   as used in experiment two

Figure 6.2: The navigation schemes used in the evaluation.

reposition the cursor. Second, after positioning the cursor over the back button, the user can short-cut back to the start page using the 'back-menu'. Using the back-menu requires an extra cursor positioning task to allow selection of the start page from the menu.

The data from experiment one was analysed using a paired T-Test to compare task performance using the back button and the gesture system.

- **Experiment Two** — Experiment two examined the effectiveness of each interface in breadth-first navigation, also known as 'hub-and-spoke' navigation (Catledge & Pitkow 1995). Hub-and-spoke navigation involves visiting a series of links (or 'spokes'), one at a time, off a central hub page. This task can be likened to visiting members of a faculty, one at a time, by selecting a series of links on the 'Faculty' page.

  Figure 6.2(b) depicts the navigation path used. Beginning at the 'start' page, subjects navigated to a main 'hub' page, and then navigated to three 'spokes' from that page, issuing the 'back' command to return to the hub each time. Finally, the subject issued a back command to return to the start page.

  When using the back button, the movements required by the mouse are much higher than those of experiment one. After selecting each link, the user must point to and click the back button, and return to the links to select the next page. In this scenario, Fitts' Law predicts the targeting time required by the back button technique will result in slower task performance than the gesture interface.

  A second factor for 'amplitude' was introduced in this experiment. The distance between the back button and links on the hub-page were varied, giving three levels of amplitude, low, medium and high, with mean distances from the back button of 8.5cm, 14cm and 20cm. The location of these links were in the top-left, bottom-left, and bottom-right corners of each page for low, medium and high respectively. Figure 6.1 shows each of the three different hub-pages used in the experiment.

  The performance data in experiment two was analysed using a two-factor, repeated measures, analysis of variance (ANOVA). The factors were 'interface type' with two levels (back button and gesture system) and 'amplitude' with three levels (low, medium and high).

- **Subjective Measures** — During the evaluation subjects were asked seven questions summarised in Table 6.1. The aim of the questions was to measure the subjects' satisfaction with the back button, before and after using the gesture system, and the efficiency and learnability of the gesture system. Throughout the evaluation, subjects were also encouraged to make comments relating to the relative performance of each system. All questions were answered on a five-point Lickert-scale from one (disagree) to five (agree).

  Question 1, "The back button is an effective means of navigation", was presented at the start of the evaluation, before any tasks or training had been completed. After both experiments had been

completed using the back button, Question 2 was presented, "The back button allowed me to quickly navigate the pages."

After completing the first practice session using the gesture system, subjects responded to Question 3, "The gesture system will allow me to navigate faster." When both experiments were completed using the gesture system, Question 4 was presented, "The gesture system did allow me to navigate faster."

To conclude the evaluation, users responded to Question 5, "The back button is an effective means of page navigation", and Question 6, "The gesture system is an effective means of page navigation." Finally, users were asked to rate the learnability of the gesture system in response to Question 7, "The gesture system was easy to learn."

6.3 Results

To ensure that a learning effect did not confound the experiment, it was necessary for subjects to perform their tasks in an expert manner, without instruction from the experimenter. From our observation of the subjects' performance, we can confirm that the subjects' repeated rehearsal of the navigation paths promoted expert performance.

Across both interfaces, mean task completion time was 6.63 (s.d. 0.97) seconds. Considering that both experiments involved displaying a total of nine pages per task, the task completion time may seem unrealistically low. Prior research, however, indicates web browsing is an extremely rapid activity (Cockburn & McKenzie 2001), with a high percentage of page visits lasting less than one second.

Although most subjects' had no difficulty in learning to make valid gestures, one subject persisted in making slow and deliberate gestures, even when the importance of rapid flicks was stressed. The resultant slow gestures meant his time for the gesture tasks was three and a half standard deviations away from the mean of the remaining nineteen subjects. As the intention of the evaluation was to measure expert performance, the subject's data was removed from the evaluation.

Experiment One

Experiment one involved depth-first navigation through a series of links, followed by backtracking to the start page.

The mean completion times for the back button and gesture systems were 6.1 (s.d. 1.10) seconds and 5.4 (s.d. 0.96) seconds respectively, showing a reduction of 11% in the mean task time when using the gesture system. This is a significant difference: two tailed T-Test, $t(18) = 2.68, p < 0.05$.

The result in favour of the gesture system is surprising given the small amount of mouse movement required when using the back button and that the back-menu can be used as a shortcut to the start page. Six subjects did use the back-menu to return to the start page, although four of these were still faster using the gesture system. One subject incorrectly identified the start page in the back menu, and had to issue one further back command.

Two subjects used the backspace key to issue the back command when using the normal browser. Both solved the task extremely rapidly, with times of 4.9 and 4.7 seconds. Interestingly, one of these subjects was still faster using the gesture system, with a time of 3.1 seconds.

Experiment Two

The second experiment compared the efficiency of the two interfaces for breadth-first 'hub-and-spoke' browsing. The experiment also measured the relative performance of the two interfaces as the distance between the back button and links on the page were increased. The mean task completion times for each interface and amplitude level are shown in Figure 6.3.

The mean task completion time for the back button was 7.44 (s.d. 1.12) seconds compared to a mean of 6.09 (s.d. 1.19) seconds for the gesture system, giving a significant main effect: $F(1, 18) = 82.2, p < 0.001$. The gesture system reduced the mean task time by 18%.
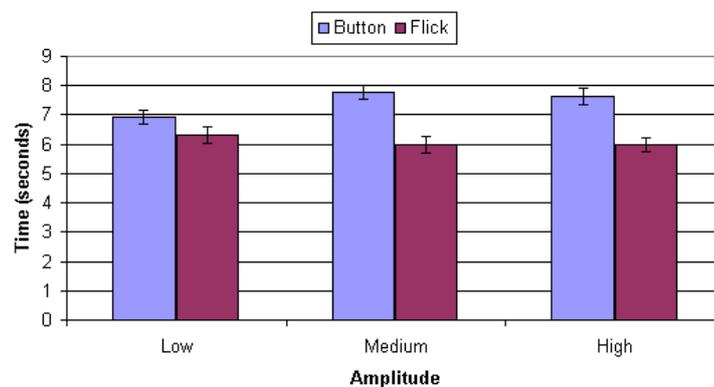
Figure 6.3: Mean task completion times for experiment two. Error bars show one standard error above and below the mean.

The mean task time for the three levels of amplitude low, medium and high were 6.6 (s.d. 1.13) seconds, 6.9 (s.d. 1.48) seconds and 6.8 (s.d. 1.39) seconds respectively. This was not a significant different: $F(2,36) = 1.17, p = 0.32$.

The interaction between factors 'interface type' and 'amplitude' was significant: $F(2,36) = 6.78, p < 0.01$. Figure 6.3 shows the cause for this interaction. As the amplitude increases, the mean completion time using the gesture system stays relatively constant, whereas the time when using the normal system increases.

Subjective Measures

The subjects' responses to the Lickert-scale questions showed some interesting results. Mean responses to each question are summarised in Table 6.1.

Before beginning either experiment, subjects were asked to rate the back button as a means of page navigation (Q1). The mean response was 2.95 (s.d. 1.13). After using the gesture system, the subjects were again asked the same question (Q5). The responses showed a significant decrease to 2.42 (s.d. 1.12): Wilcoxon Signed Ranks test, $N = 8, z = 2.45, p < 0.01$.

Comparing the mean responses to Question 3 and Question 4 shows that the subjects found the gesture system faster than they expected it to be after their initial training. Mean responses increased from 4.11 (s.d. 0.81) to 4.63 (s.d. 0.76) giving a reliable difference: Wilcoxon Signed Ranks test, $N = 11, z = 2.4, p < 0.01$.

Eighteen of the nineteen subjects rated the efficiency of the gesture higher than that of the back button (Q5 and Q6). The remaining subject gave the same rating for both interfaces. The mean responses for the effectiveness of the back button and gesture system were 2.42 (s.d. 1.12) and 4.26 (s.d. 0.81), giving a reliable difference: Wilcoxon Signed Ranks test, $N = 17, z = 3.46, p < 0.01$.

Finally, subjects rated the gesture system highly for ease of learning with a mean response of 4.26 (s.d. 0.93).

User Comments

Many subjects were extremely enthusiastic about the system, making comments such as "Really really nice", and "That's amazing". There were, however, some negative comments regarding the implementation used in the evaluation. Six subjects mentioned they were distracted by the overloading issue, which arose from assigning the gesture system to use the left mouse button. This caused a problem in two situations. First, when the cursor was over a link in the page, and second, when starting a gesture over non-link text.

As mentioned in Chapter 5, gestures are not recognised when initiated over a link. Consequently, the subjects had to put some care into the location of gestures in the browser window. This requirement is

Question	Mean	SD
Q1. The back button is an effective means of page navigation.	2.95	1.13
Q2. The back button allowed me to quickly navigate the pages.	3.26	1.19
Q3. The gesture system will allow me to navigate faster.	4.11	0.81
Q4. The gesture system did allow me to navigate faster.	4.63	0.76
Q5. I think the back button is an effective means of page navigation.	2.42	1.12
Q6. I think the marks system is an effective means of navigation.	4.26	0.81
Q7. Was the marks system easy to learn?	4.26	0.93

Table 6.1: Mean responses to 5-point Lickert scale questions

somewhat contradictory to the relatively informal and 'sloppy' nature of the flick gesture. This problem would easily be overcome in a commercial browser.

The second problem mentioned by users, was caused when initiating a gesture over an area of text. This did not cause the gesture recognition to fail, although it did cause momentary concern for some users. As mentioned in Chapter 3, dragging the cursor over an area of text with the left mouse button causes text selection. To differentiate between a gesture action and a text selection action, rapid mouse-down motions were recognised as gestures, and longer mouse-down motions as text selection (as described in Chapter 5). This overloading meant that when the user initiated a gesture over a piece of text, the text was highlighted as normal. This brief 'flash' of selected text prior to releasing the button meant some users resorted to "having to find a clear bit of the page".

6.4   Discussion

To summarise the results, the gesture system significantly reduced the mean task completion times in both experiments. In the depth-first traversal browsing task of experiment one, the gesture system reduced the mean task times by approximately 11%. In the hub-and-spoke traversal browsing task of experiment two, the reduction was approximately 18%.

Considering that backtracking activities such as those in experiment one and two are completed hundreds of millions of times every day, gesture navigation has the potential to enhance the overall efficiency of web navigation.

The potential of the gesture navigation is further demonstrated by the the minimal training that our subjects required and by the enthusiastic comments and subjective measures provided by the subjects.

The applicability of our results are not limited to that of web browsing. Tasks that are dominated by the mouse also stand to benefit from a simple gesture implementation such as that discussed in this evaluation. A possible candidate is the standard windows based file explorer which is somewhat similar to the web browser environment. This environment supports the 'back' and 'forward' commands. The system and could also include mappings for the up and down flick gestures to control window and maximisation minimisation, both of which would provide iconic mappings.

Explorer Mouse

An alternative to using flick gestures or the back button to go 'back' is the Microsoft IntelliMouse Explorer (Microsoft 2001). This five button mouse allows users to configure the extra buttons to execute desired tasks. This has an obvious advantage over the two techniques discussed in our evaluation, the fact that a 'back' command can be issued independent of the location and relocation of the mouse cursor. This does not invalidate the results of our research, however, as such devices are extremely rare amongst users, and are generally an optional extra when purchasing a computer package. For the hundreds of millions of users without this technology, the flick gesture provides an effective alternative to traditional navigational techniques.

Confounding Factors

There were several potential confounding factors that might have affected the results of the evaluation.

- Implementation limitations — As mentioned in Chapter 5, the Javascript implementation was not optimal, as subjects had to adjust their flicks to ensure they were not initiated over a link. Some subjects were also distracted by the momentary 'flash' of selected text when a flick was initiated over an area of text. These factors are likely to have slowed performance with the gesture system although as discussed in Section 6.3, overcoming these limitations in a commercial browser should be relatively straight forward.

- Browser preference — All experiments were conducted using Internet Explorer (IE). Although each subject had used IE previously, approximately half used Netscape Navigator as their normal browser. The differences between these two browsers at the interface level are extremely minor. Also, the location and behaviour of the back button is similar for both browsers. It is therefore unlikely that using IE affected our results.

- Subject pool — All of the subjects were post-graduate Computer Science students. Although this group has more experience using the mouse than most others, we do not believe that their motor skills in generating the 'flick' gesture will be significantly different to other user groups.

- Measurement tool — All tasks were timed using a stopwatch rather than software logging. This presents some obvious inaccuracies. Initial trials of the experiment used server logs to record navigation times, but these provided access times at a one-second granularity, which is a coarser measure than can be achieved using a stopwatch.

Overall, we do not believe removing or controlling these factors will have a significant impact on the primary result, that reveals the efficiency offered by gesture navigation.

Chapter 7

Conclusion

This report investigated issues relating to the design, implementation, and usability of the flick gesture by way of three user evaluations.

The first evaluation determined the minimum time taken during the common dragging task of text selection. Overall, single character tasks were selected the quickest, and selection occurred predominantly in the right direction.

The second evaluation investigated the natural properties of linear flick gestures made with the mouse and pen in the four directions; left, right, up and down. These properties included the size, timing and directional accuracy of flick gestures. The left and right mouse button were found to have no impact on the size and time taken to create flick gestures. The pen was found to produce high angular errors for gestures created in the left and right directions. Also, down gestures created using the mouse were approximately 11% slower than gestures created in the other three directions.

The results of the two evaluations were then used to guide the implementation of a web browser gesture system. The left flick gesture was assigned to the 'back' button, and the right flick gesture to the 'forward' button.

The web gesture system was then evaluated to determine the efficiency and learnability of the flick gesture in a real world application. The evaluation involved two navigational tasks; depth-first traversal then back, and breadth-first traversal then back. Results showed that the gesture system significantly reduced the time taken to complete these tasks, with mean task time reductions for depth-first and breadth-first navigation tasks of 11% and 18% respectively. Users' found the gesture system easy to learn, and their subjective ratings showed a strong preference for the gesture system.

The results of this report show the flick gesture to be an effective and efficient alternative for the activation of common controls. We look forward to its wide spread use in future applications.

Appendix A

Gesture System Script File

```
<script language="JavaScript">

<!--
// Our Variables...
// IExpBro tells us if the client is using IExpBro...

// The maximum time duration of a gesture (milliseconds)
var MAXTIME = 250;

// The minimum pixel value the cursor must travel
var XMIN = 35;
var YMIN = 35;

var IExpBro = document.all?true:false

// Variables to record the x and y movement
var xMovement = 0;
var yMovement = 0;

// Variables to record the x and y position at mouse down and mouse up
var xOnButtonDown = 0;
var yOnButtonDown = 0;
var xOnButtonUp = 0;
var yOnButtonUp = 0;

// Variables to record the time on mouse down and mouse up
timeOnDown = new Date();
timeOnUp = new Date();


// Function called when the mouse button is depressed
// Records the time of the event, and the x,y coordinate of the mouse
function mouseDown(e) {
    if (parseInt(navigator.appVersion)>3) {
        // Record the time pressed
        timeOnDown = new Date();

        // Variable to determine which button was pressed.
        var buttonNum = 0;

        // Depending on the browser, get the button number
        if (navigator.appName=="Netscape")
            buttonNum=e.which;
        else
            buttonNum=event.button;

        // If it was the left button, record the x,y coords.
```

10

20

30

40

37

```
        if (buttonNum==1) {
            if (IExpBro) {
                xOnButtonDown = event.clientX;                                    50
                yOnButtonDown = event.clientY;
            } else {
                xOnButtonDown = e.pageX;
                yOnButtonDown = e.pageY;
            }
            return false;
        }
    }
    return true;
}                                                                                 60


// Function called when the mouse button is released..
// Calculates the time duration of the button press, and also
// the x,y changes. If the movement satisfies the constraints, then
// issue the appropriate command...eg. back or forward.
function mouseUp(e) {
    // If the browser can handle JavaScript then...
    if (parseInt(navigator.appVersion)>3) {
        // Record the time on button release                                      70
        timeOnUp = new Date();

        // Make a variable to determine which button was pressed
        var buttonNum = 0;

        // Get the number of the button pressed
        if (navigator.appName=="Netscape")
            buttonNum=e.which;
        else
            buttonNum=event.button;                                               80

        // Filter out long mouse clicks which may indicate
        // highlighting of text etc...
        if ( (timeOnUp − timeOnDown) > MAXTIME )
            return true;

        // If this was the left mouse button, record the x,y coords
        if (buttonNum==1) {
            if (IExpBro) {
                xOnButtonUp = event.clientX;                                      90
                yOnButtonUp = event.clientY;
            } else {
                xOnButtonUp = e.pageX;
                yOnButtonUp = e.pageY;
            }

            // Calculate the movement
            xMovement = xOnButtonDown − xOnButtonUp;
            yMovement = yOnButtonDown − yOnButtonUp;
                                                                                  100
            // If the movement is bigger than the minimum
            if (xMovement > XMIN && yMovement < xMovement &&
                yMovement > −xMovement ) {
                // Movement was to the left
                history.go(−1);
            } else if (xMovement < −XMIN && yMovement > xMovement &&
                    yMovement < −xMovement ) {
                // Movement was to the right
                history.forward();
            }                                                                     110
            return false;
        }
    }
    return true;
```

```
}

// Assign the mouseDown events to invoke the above functions...
if (parseInt(navigator.appVersion)>3) {

    if (document.captureEvents)                                           120
        document.captureEvents(Event.MOUSEDOWN);
    document.onmousedown = mouseDown;
    document.onmouseup = mouseUp;

}

//->

</script>
```

Appendix B

Man Page Extract for `xset`

```
...
        The m option controls  the  mouse  parameters.    The
        parameters  for  the  mouse  are  'acceleration' and
        'threshold'.  The acceleration can be   specified   as
        an   integer, or as a simple fraction.   The mouse, or
        whatever pointer the machine is connected  to,   will
        go 'acceleration' times as fast when it travels more
```

```
        than 'threshold' pixels in a short time.   This   way,
        the   mouse can be used for precise alignment when it
        is moved slowly, yet it can be set to travel   across
        the   screen   in   a   flick of the wrist when desired.
        One or both parameters for the m option can be omit-
        ted,  but  if   only   one is given, it will be inter-
        preted as the acceleration.  If no parameters or the
        flag  'default' is used, the system defaults will be
        set.
...
```

Bibliography

Balakrishnan, R. & Patel, P. (1998), The Padmouse: Facilitating selection and spatial positioning for the non-dominant hand, *in* 'Proceedings of CHI'98 Conference on Human Factors in Computing Systems Los Angeles, April 18–23', pp. 9–16.

Buyukkokten, O., Garcia-Molina, H., Paepcke, A. & Winograd, T. (2000), Power Browser: Efficient Web Browsing for PDAs, *in* 'Proceedings of CHI'2000 Conference on Human Factors in Computing Systems The Hague, The Netherlands, April 1–6', pp. 430–437.

Callahan, J., Hopkins, D., Weiser, M. & Shneiderman, B. (1988), An Empirical Comparison of Pie versus Linuear Menus, *in* 'Proceedings of CHI'88 Conference on Human Factors in Computing Systems', pp. 95–100.

Card, S., English, W. & Burr, B. (1978), 'Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT', *Ergonomics* **21**(8), 601–613.

Catledge, L. & Pitkow, J. (1995), Characterizing Browsing Strategies in the World Wide Web, *in* 'Computer Systems and ISDN Systems: Proceedings of the Third International World Wide Web Conference. 10–14 April, Darmstadt, Germany', Vol. 27, pp. 1065–1073.

Chatty, S. & Lecoanet, P. (1996), Pen Computing for Air Traffic Control, *in* 'Proceedings of CHI'96 Conference on Human Factors in Computing Systems Vancouver, April 13–18', pp. 87–94.

Cockburn, A. & McKenzie, B. (2001), 'What Do Web Users Do? An Empirical Analysis of Web Use', *International Journal of Human-Computer Studies* **54**(6), 903–922.

Damm, C. H., Hansen, K. M. & Thomsen, M. (2000), Tool support for cooperative object-oriented design: Gesture based modelling on an electronic whiteboard, *in* 'Proceedings of CHI'2000 Conference on Human Factors in Computing Systems The Hague, The Netherlands, April 1–6', pp. 518–525.

Dulberg, M., Amant, R. & Zettlemoyer, L. (1999), An Imprecise Mouse Gesture for the Fast Activation of Controls, *in* 'Proceedings of INTERACT'99', pp. 375–382.
URL: *citeseer.nj.nec.com/martin99imprecise.html*

Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pederson, E., Pier, K., Tang, J. & Welch, B. (1992), Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration, *in* 'Proceedings of CHI'92 Conference on Human Factors in Computing Systems Monterey, May 3–7', Addison-Wesley, pp. 599–607.

Gillan, D., Holden, K., Adam, S., Rudisill, M. & Magee, L. (1990), How does Fitts' Law fit pointing and dragging, *in* 'Proceedings of CHI'90 Conference on Human Factors in Computing Systems Seattle, April', pp. 227–234.

Goldberg, D. & Richardson, C. (1993), Touch-Typing With a Stylus, *in* 'Proceedings of CHI'93 Conference on Human Factors in Computing Systems Amsterdam, April 24–29', Addison-Wesley, pp. 80–87.

Isokoski, P. (2001), Model for Unistroke Writing Time, *in* 'Proceedings of CHI'2001 Conference on Human Factors in Computing Systems Seattle, Washington, March 31–April 6', pp. 357–364.

Kurtenbach, G. & Buxton, W. (1991), Issues in combining marking and direct manipulation techniques, *in* 'UIST Fourth Annual Symposium on User Interface Software and Technology', pp. 137–44.

Kurtenbach, G. & Buxton, W. (1993), The Limits of Expert Performance Using Hierarchic Marking Menus, *in* 'Proceedings of INTERCHI'93 Conference on Human Factors in Computing Systems', pp. 482–487.

Kurtenbach, G. & Buxton, W. (1994), User learning and performance with marking menus, *in* 'Proceedings of CHI'94 Conference on Human Factors in Computing Systems Boston, April 24–28', Vol. 2, pp. 258–264.

Kurtenbach, G., Fitzmaurice, G., Owen, R. & Baudel, T. (1999), The Hotbox: efficient access to a large number of menu-items, *in* 'Proceedings of CHI'99 Conference on Human Factors in Computing Systems Pittsburgh, May 15–20', pp. 231–237.

Kurtenbach, G., Fitzmaurice, G. W., Baudel, T. & Buxton, W. (1997), The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency, *in* 'Proceedings of the ACM SIGCHI'97 Conference on Human Factors in Computing Systems, Atlanta, Georgia, March 22-27', pp. 35–42.

Kurtenbach, G., Sellen, A. & Buxton, W. (1993), An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus, *in* 'Human-Computer Interaction, 8(1)', pp. 1–23.

Lionhead (2001), 'Black & White'.
    URL: *http://www.bwgame.com*

Long, J., Landay, J. & Rowe, L. (1999), Implications for a Gesture Design Tool, *in* 'Proceedings of CHI'99 Conference on Human Factors in Computing Systems Pittsburgh, May 15–20', pp. 40–47.

Long, J., Landay, J., Rowe, L. & Michiels, J. (2000), Visual similarity of pen gestures, *in* 'Proceedings of CHI'2000 Conference on Human Factors in Computing Systems The Hague, The Netherlands, April 1–6', pp. 360–367.

MacKenzie, I. S. & Zhang, S. X. (1997), The Immediate Usability of Graffiti, *in* W. A. Davis, M. Mantei & R. V. Klassen, eds, 'Graphics Interface '97', Canadian Human-Computer Communications Society, pp. 129–137.

MacKenzie, I., Sellen, A. & Buxton, W. (1991), A Comparison of Input Devices in Elemental Pointing and Dragging Tasks, *in* 'Proceedings of CHI'91 Conference on Human Factors in Computing Systems', ACM, pp. 161–166.

Maya (2001), 'Maya Product Brochure'.
    URL: *http://www.aliaswavefront.com*

Microsoft (2001), 'Microsoft IntelliMouse Explorer'.
    URL: *http://shop.microsoft.com?Referral/Productinfo.asp?siteID=11070*

Mozilla Organization (2001), 'Mozilla'.
    URL: *http://www.mozilla.org*

Nielsen, J. (1993), *Usability Engineering*, London: Academic Press.

Opera Software (2001), 'Opera Web Browser'.
    URL: *http://www.opera.com*

Pedersen, E. R., McCall, K., Moran, T. P. & Halasz, F. G. (1993), Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings, *in* 'Proceedings of CHI'93 Conference on Human Factors in Computing Systems Amsterdam, April 24–29', pp. 391–398.

Perlin, K. (1998), Quikwriting: Continuous Stylus-Based Text Entry, *in* 'ACM Symposium on User Interface Software and Technology', pp. 215–216.
   URL: *citeseer.nj.nec.com/perlin98quikwriting.html*

Rubine, D. (1991*a*), Specifying Gestures by Example, *in* 'Computer Graphics 25(4)', pp. 329–337.

Rubine, D. (1991*b*), The automatic recognition of gestures, PhD thesis, Carnegie Mellon University.

Tapia, M. & Kurtenbach, G. (1995), Some Design Refinements and Principles on the Appearance and Behavior of Marking Menus, *in* 'Proceedings of ACM Conference on User Interface Software and Technology, Pittsburgh, November 14–17.', pp. 189–195.

Tauscher, L. & Greenberg, S. (1997), 'How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems', *International Journal of Human Computer Studies, Special issue on World Wide Web Usability* **47**(1), 97–138.

Venolia, D. & Neilberg, F. (1994), T-Cube: A Fast, Self-Disclosing Pen-Based Alphabet, *in* 'Proceedings of CHI'94 Conference on Human Factors in Computing Systems Boston, April 24–28', Addison-Wesley, pp. 265–270.