THE DESIGN AND EVALUATION OF MARKING MENUS

by

Gordon Paul Kurtenbach

A thesis submitted in conformity with the requirements
of the Degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

This research focuses on the use of hand drawn marks as a human-computer input technique. Drawing a mark is an efficient command input technique in many situations. However, marks are not intrinsically self-explanatory as are other interactive techniques such as buttons and menus. This research develops and evaluates an interaction technique called marking menus which integrates menus and marks such that both self-explanation and efficient interaction can be provided.

A marking menu allows a user to perform a menu selection by either popping up a radial menu and then selecting an item, or by drawing a straight mark in the direction of the desired menu item. Drawing a mark avoids popping up the menu. Marking menus can also be hierarchic. In this case, hierarchic radial menus and "zig-zag" marks are used. Marking menus are based on three design principles: self-revelation, guidance and rehearsal. Self-revelation means a marking menu reveals to a user what functions or items are available. Guidance means a marking menu guides a user in selecting an item. Rehearsal means that the guidance provided by the marking menu is a rehearsal of making the mark needed to select an item. Self-revelation helps a novice determine what functions are available, while guidance and rehearsal train a novice to use the marks like an expert. The intention is to allow a user to make a smooth and efficient transition from novice to expert behavior.

This research evaluates marking menus through empirical experiments, a case study, and a design study. Results shows that (1) 4, 8 and 12 item menus are advantageous when selecting using marks, (2) marks can be used to reliably select from four-item menus that are up to four levels deep or from eight-item menus that are up to two levels deep, (3) marks can be performed more accurately with a pen than a mouse, but the difference is not large, (4) in a practical application, users tended towards using the marks 100% of the time, (5) using a mark, in this application, was 3.5 times faster than selection using the menu, (6) the design principles of marking menus can be generalized to other types of marks.

THE DESIGN AND EVALUATION OF MARKING MENUS

Gordon Paul Kurtenbach
Degree of Doctor of Philosophy
Graduate Department of Computer Science, University of Toronto, 1993

abstract

This research focuses on the use of hand drawn marks as a human-computer input technique. Drawing a mark is an efficient command input technique in many situations. However, marks are not intrinsically self-explanatory as are other interactive techniques such as buttons and menus. This research develops and evaluates an interaction technique called marking menus which integrates menus and marks such that both self-explanation and efficient interaction can be provided.

A marking menu allows a user to perform a menu selection by either popping up a radial menu and then selecting an item, or by drawing a straight mark in the direction of the desired menu item. Drawing a mark avoids popping up the menu. Marking menus can also be hierarchic. In this case, hierarchic radial menus and "zig-zag" marks are used. Marking menus are based on three design principles: self-revelation, guidance and rehearsal. Self-revelation means a marking menu reveals to a user what functions or items are available. Guidance means a marking menu guides a user in selecting an item. Rehearsal means that the guidance provided by the marking menu is a rehearsal of making the mark needed to select an item. Self-revelation helps a novice determine what functions are available, while guidance and rehearsal train a novice to use the marks like an expert. The intention is to allow a user to make a smooth and efficient transition from novice to expert behavior.

This research evaluates marking menus through empirical experiments, a case study, and a design study. Results shows that (1) 4, 8 and 12 item menus are advantageous when selecting using marks, (2) marks can be used to reliably select from four-item menus that are up to four levels deep or from eight-item menus that are up to two levels deep, (3) marks can be performed more accurately with a pen than a mouse, but the difference is not large, (4) in a practical application, users tended towards using the marks 100% of the time, (5) using a mark, in this application, was 3.5 times faster than selection using the menu, (6) the design principles of marking menus can be generalized to other types of marks.

Acknowledgments

Many years ago when I was in high school my classmates and I spent three days writing occupation aptitude tests. Months later the computer graded tests were returned to us. I remember my friends' and my own excitement as we ripped open the envelopes to see what the computer had recommended. My friends cheered as they read out their long list of possibilities: doctor! lawyer! pilot! writer! scientist! With great anticipation I opened my computer recommendation. There, before my eyes, was one lonely recommendation: *pre-cast concrete worker.*

Although I have failed to fulfill my destiny as pre-cast concrete worker, I have created this thesis with the support of many people. In particular, I would like to thank:

- My supervisor and friend, Bill Buxton. Bill's creativity, intellect, and humor inspired me to pursue research and make bad jokes.

- The members of my committee: Ron Baecker, Mark Chignell, Marilyn Mantei, Ken Sevcik, and Cathy Wolf. Each contributed in helping me polish my research into a doctoral thesis.

- Great researchers and friends. Abigail Sellen greatly helped by designing experiments, writing, and putting on excellent parties; Tom Moran, Stuart Card, and Ken Pier provided creative insights and guidance; George Fitzmaurice and Beverly Harrison waded through treacherous drafts of my thesis, helped me make it a better document, and listened to my concerns over many a cappuccino; Gary Hardock utilized my work in his research and put up with my kidding; George Drettakis and Dimitri Nastos kept the lab systems running, humored me, and organized the most delicious Greek barbecues; Tim Brecht advised me, made me laugh way too loud and long, yet still managed to keep me sane.

I don’t think I’ll thank the computer that graded the aptitude tests...

To my parents, Helen and Leo,

and my brother and sisters,

Robert, Beverly, Donna, Carole, and Tammy:

“My thesis is done, you can probably reach me at home now”

Table of Contents

Chapter 1: Introduction

Research in the last forty years has brought great improvements in the quality of human-computer interactions. In the past, human-computer dialogs were optimized for the computer; humans communicated with computers using protocols that were easy for the computer to understand but were hard for a human to understand and use, for example, machine languages. Advances in human-computer interaction have changed this situation. Controlling a computer no longer requires memorizing obtuse, cryptic codes or an intimate understanding of the internal workings of the computer. In well-designed systems, human-computer interactions are optimized for the human. Interfaces now make use of sophisticated graphics, sound, and pointing devices to make the human's job easier.

The major advances in human-computer interaction have been in making computers easier to use. Specifically, research on methods to reduce the amount of training a person needs before being able to operate a computer has come a long way. For example, the *Apple Macintosh* has set standards for the minimal amount of instruction that a person needs before operating a computer. Because of these advances, the world of computers opened up for people who otherwise would not have invested the time in training to operate a computer system.

Given these advances in human-computer interaction, we can think of the interface as currently being optimized for the human, specifically, the novice computer user. Clearly, this is of great value, but we can consider another important class of user—the expert. Human capacity for the development of skills is great. Virtuoso pianists are proof of this. Virtuosos invest a great deal of time in practicing their skills—eight hours of practice a day is not uncommon. Now consider expert computer users. It is not uncommon for an expert computer user to spend eight hours a day working on the computer. Therefore, there is untapped potential for human skill

development in human-computer interactions. A good interface should take advantage of this potential and not limit the efficiency of a skilled user.

In order for this skill potential to be tapped, an interface must have certain properties. First, the interface must provide interaction methods that are suitable for an expert. Experts require efficient interactions. As a result, interactions may be terse and unprompted. Second, and most critically, the interface must also provide support for a novice to become expert. We look at the interface design not so much as making the interface easier to use but rather as *accelerating the rate at which novices begin to perform like experts*. This goal demands three components: support for the novice, support for the expert, and an efficient mechanism to support the transition from novice to expert (see Figure 1.1).

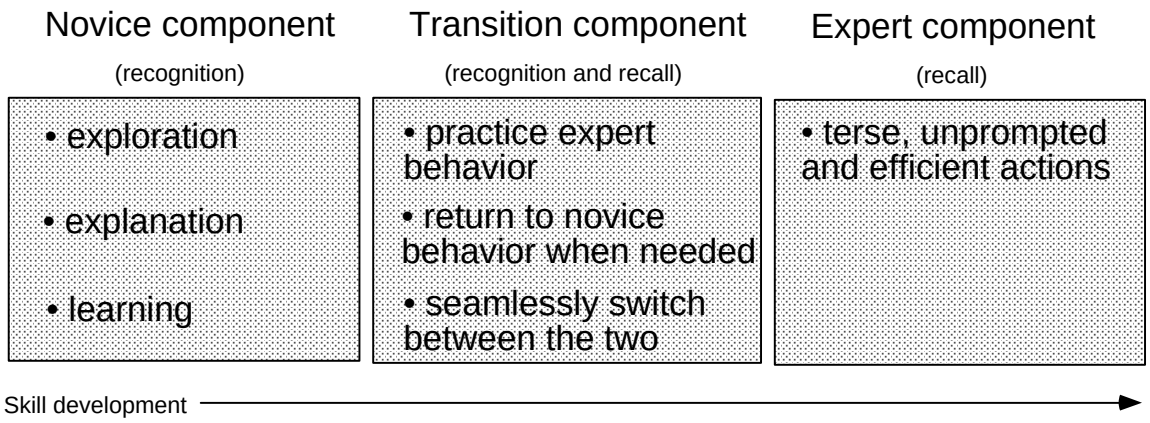


Figure 1.1: The components required to accelerate the rate at which users begin to perform like experts. The novice component allows a user to issue commands by searching for them and recognizing them. The expert component allows a user to efficiently issue commands by recalling the action associated with the command. The transition component allows a user to efficiently switch between these two methods to learn and practice command action associations.

In this dissertation, we focus on an interaction technique that is intended to take advantage of this skill potential and support the development of skill. We propose an interaction technique which has a two modes. In the first mode, the style of interaction is intended to facilitate novice use. In the second mode, the style of interaction is intended for skilled expert behavior. The first mode is also designed to allow a novice to practice the skills required in the second mode. A user can switch to the second mode by operating the technique quickly. One can think of this in metaphorical terms. When you are learning to drive a car, its suitable to have a car

that is designed for a student driver. However, as your driving skills improve, the car incrementally transforms into a Ferrari.

1.1. GENERAL AREA AND DEFINITIONS

To support the expert component described in the previous section, we focus on a style of human computer interaction in which a user "writes" on the display surface. This style of interaction is similar to writing or drawing with a pen on ordinary paper. Writing on a display, however, is accomplished with a special pen and the computer simulates the appearance of ink.[1]

We define a *mark* as the series of pixels that are changed to a special "ink" color when the pen is pressed and then moved across the display. The pixels that are changed to an ink color are those which lay directly under the tip of the pen as it is moved across the display. Free hand drawings, ranging from meaningless scribbles to meaningful line drawings and symbols, including handwriting, are examples of marks. The act of drawing a mark is referred to as *marking*.

Marks can be created not only with a pen but also with other types of input devices. For example, a mouse can leave a trail of ink (commonly referred to as an *ink-trail*) behind the tracking symbol when the mouse button is pressed and the mouse is dragged. Some systems use a pen and tablet. In this case, marks are made on the display by writing on the tablet instead of the display.

From a user's point of view, these interfaces allow one to make marks and then have the system interpret those marks. There are, however, systems in which marks can be made but not recognized by the system. They are interpreted strictly as annotations, for example, *Freestyle* (Perkins, Blatt, Workman, & Ehrlich, 1989). The focus of this dissertation, however, is on systems in which marks are interpreted as commands and parameters.

Much of the literature refers to marks as *gestures*. However, the term gesture is inappropriate in this context. Indeed creating a mark does involve a physical

---

[1] The pen, in these types of systems, is sometimes referred to as a stylus.

gesture but the real object of interpretation is the mark itself.[2] For example, the "X" mark requires a completely different physical gesture if performed with a pen instead of a mouse. Gesture is an important aspect of mark because some marks may require awkward physical gestures with the input device. However, the two terms should be distinguished. The term gesture is more appropriate for systems in which the gestures leave no marks, for example, *VideoPlace* (Krueger, Giofriddo & Hinrichsen, 1985). The term mark is more appropriate for pen-based computer systems or applications that emulate paper and pen.

1.2. WHY USE MARKS?

Current human-computer interfaces are asymmetric in terms of input and output capabilities. There a number of computer output modes: visual, audio and tactile. Most computers extensively utilize the visual mode; high resolution images which use thousands of colors of can be displayed quickly and in meaningful ways to a user. In contrast, a computer's ability to sense user input is limited. Humans have a wide range of communication skills such as speech and touch, but most computers sense only a small subset of these. For example, keyboards only sense finger presses (but not pressure) and mice only sense very simple arm or wrist movements. Therefore, we believe the advent of the pen as a computer input device provides the opportunity to increase input bandwidth through the use of marks.[3]

There are two major motivations for using marks. The first addresses the problem of efficiently accessing the increasing number of functions in applications. The second motivation is that there are some intrinsic qualities that marks have which can provide a more "natural" way to articulate otherwise difficult or awkward concepts (such as spatial or temporal information). Both of these motivations will now be examined in more detail.

---

[2] There are systems where interpretation depends not only on what is drawn but also how it is drawn. For example, an "X" drawn quickly may have a different interpretation from a "X" that is drawn slowly. By this dissertation's terminology, these systems would contain a combination of marking and gesture recognition.

[3] It is ironic that one of the first input devices for graphics was a light pen which wrote directly on the display surface (Sutherland, 1963).

1.2.1. Symbolic nature

The inadequacy of mouse and keyboard interfaces is exemplified by applications that are controlled through button presses and position information.[4] Buttons must be accessible and thus require physical space. Problems occur when an application has more functions than can be mapped to buttons or reasonably managed on the display. Other problems also exist: arbitrary mappings between functions and buttons can be confusing, and user management of the display and removal of graphical buttons can be tedious.

Expert users of these types of systems find the interface inadequate because button interfaces are inefficient. The existence of interaction techniques that override buttons for the sake of efficiency is evidence of this. Experts, having great familiarity with the interface, are aware of the set of available commands. Menus are no longer needed to remind them of available commands and invoking commands through menu display becomes very tedious.

Designers have addressed this problem in several ways. One solution is *accelerators keys* which allow experts direct access to commands. An accelerator key is a key on the keyboard which, when pressed, immediately executes a function associated with a menu item or button. The intention is that using an accelerator key saves the user the time required to display and select a menu item or button. Many systems display the names of accelerator keys next to menu items or buttons to help users learn and recall the associations between accelerator keys and functions.

Another way of supporting an expert is by supplying a command line interface in addition to a direct manipulation interface. Commodore's command line interface, *CLI*, and graphical user interface, *Intuition*, are an example of this approach.

Both these approaches have their problems. In the case of accelerator keys, arbitrary mappings between functions and keys can be hard to learn and remember. Sometimes mnemonics can be established between accelerator key and function (e.g., control-o for "open"), but mnemonics quickly run out as the number of accelerator keys increases. Further confusion can be caused by different applications

---

[4] The term buttons is used as a generic way of describing menus items, dialog box items, icons, keys on a keyboard, etc., which are typical of direct manipulation interfaces.

using a common key for different functions or by different applications using different keys for a common function. Experts must then remember arbitrary or complex mappings between keys and functions depending on application. Command line interfaces are problematic because they are radically different from direct manipulation interfaces. To become an expert, a novice must learn another entirely different interface.

Marks, because of their symbolic nature, can make functions more immediately accessible. Rather than triggering a function by a button press, a mark can signal a command. For example, a symbolic mark can be associated with a function and a user can invoke the function by drawing the symbol. In theory, because marks can be used to draw any symbol or series of symbols, marks can provide a quicker method of choosing a command than searching for a physical or graphical button and pressing it. In practice, the number of marks is limited by the system's ability to recognize symbols and a human's ability to remember the set of symbols. Nevertheless, even if only a small set of marks are used, a user can invoke the associated functions immediately.

Marks can also be used to hide functions because they are user generated symbols. For example, researchers at Xerox PARC made use of this property when faced with a dilemma during the design of a pen-based application. This application runs on a wall sized display where a user can write on the display using an electric pen (Elrod et. al., 1992). There were two major design requirements. First, the designers wanted the application to look and operate like a whiteboard and maximize the size of the area where drawing could take place. Second, they wanted to provide additional functions commonly found in computer drawing programs. This second requirement meant that many graphical buttons would need to appear on the screen. This, however, violated the first design requirement because the numerous graphical buttons would consume too much of the drawing area and make the interface look complicated.

The design solution was to assign many of the drawing functions to marks. Marks provided a way to hide additional functionality from novices while expert users could use the marks to access additional functions. This design also avoided using buttons for these functions and, in many cases, marks were a much more effective way of articulating a function.

1.2.2. Intrinsic advantages

The advantages of pen input and marks have been expressed in the literature (Bush, 1945; Licklider 1960; Ellis & Sibley, 1967; Hornbuckle, 1967; Coleman, 1969; Ward & Blesser, 1985; Rhyne & Wolf, 1986; Wolf, 1986; Buxton, 1986; Welbourn & Whitrow, 1988; Wolf, Rhyne, & Ellozy, 1989; Morrel-Samuels, 1990; Kurtenbach & Hulteen, 1990). Specifically, marks provide the ability to:

• embed many command attributes into a single mark;

• reduce learning time due to the mnemonic nature of marks and users' existing knowledge of pen and paper marks;

• capture and recognize handwriting and drawing;

• enter different types of data without switching input device. For example, text, menu selections, button presses, and screen locations can be entered without changing input device;[5]

• replace the computer keyboard, thus making computers smaller and more portable;

• maintain a visible audit trail of operations;

• maintain a clear figure/ground relationship (Hardock, 1991). For example, marks written over formatted text can be distinguished from the text.

1.3. SELF-REVELATION, GUIDANCE AND REHEARSAL

Despite all of these advantages, pen input and marks have not been widely used. Pen-based interfaces have many difficult technological requirements. Historically, hardware for pen-based systems was too expensive and recognition was not reliable (Sibert, Buffa, Crane, Doster, Rhyne, & Ward, 1987). Given these limitations pen-based applications presented no advantage (in reality, more of a disadvantage) over a mouse-based version of the application.

---

[5] This eliminates homing time between physical input devices but it does not eliminate homing time between graphical devices such as graphical buttons, sliders, etc.

This situation is changing and this change is clearly evident in the marketplace (Normile & Johnson, 1990; Rebello, 1990). Several companies such as Go, Grid, IBM, Apple, Microsoft, and NCR are introducing pen-based systems. Hardware and recognition has improved to the point where pen-based systems are technically possible. Applications such as portable notebook computers and large whiteboard size computer screens make the pen an attractive input device (Goldberg & Goodisman, 1991; Weiser, 1991).

On the surface, it appears that once the recognition and hardware problems are solved, pen-based systems will be successful. However, there is still a serious interface problem when using marks.

1.3.1. The problem: learning and using marks

An intrinsic problem with marks is that they are not *self-revealing*. In contrast, menus and buttons are self-revealing; the set of available commands and how to invoke a command is readily visible as a byproduct of the way commands are invoked. An interface which uses only marks as a means of command entry cannot support *walk-up-and-use* situations. A first time user has no way of finding out interactively from the system what marks/commands are available. This situation is reminiscent of command line interfaces such as the *UNIX* shell or *MS-DOS* where the only information presented by the system is a command line prompt. Some source of information distinct from the process of making a mark must be consulted before commands can be generated.

The problem is even more acute. Not only do users need to know what marks can be made but also when or where these marks can be made. In menu and button interfaces, one can find out when and where a command can be invoked by which buttons or menu items appear active when an interface object is selected. Marks do not have this property.

Is there a problem? Aren't the existing pen-driven systems easy to use and self-revealing? Hybrid interfaces which use both direct manipulation and marks (e.g., the *PenPoint* or *Momenta* interfaces (Go, 1991; Momenta, 1991)) may be somewhat capable of walk-up-and-use. However, only the direct manipulation components of

the interface can be used without external instruction.[6] Manuals must still be used to find out about marks. Hence these system do not solve the self-revealing/marks problem.

The motivation for creating walk-up-and-use interfaces is strong. Successful computer interfaces such as the Macintosh are based on the notion that "nobody reads manuals". These types of interfaces are designed to help a user learn and remember how to operate the interface without explicit external help such as on-line help or manuals (Sellen, & Nicol, 1990). This situation can be viewed practically: a user wants to get a certain task done; this task can be accomplished using a computer tool; the shortest path between the user and task completion is using the tool; a manual will be consulted only if the tool cannot be used directly.

If we expect a worker in the information age to utilize many different applications, a huge amount of training for each application is an unrealistic demand. Users expect interfaces that are consistent and permit transfer of skills from other applications. They also expect interfaces to be self-explanatory and to guide a user in the operation of the application. Thus, the motivation for walk-up-and-use self-revealing interfaces is paramount.

An argument can be made that walk-up-and-use interfaces are not efficient, but this argument misses the point. The reason to make marks self-revealing is so a user can graduate from using the walk-up-and-use techniques to the more efficient marks. Once this graduation has taken place, the user can benefit from the advantages of marks such as efficient articulation and conservation of screen space. The key to the success of this scheme is in how easily a novice can acquire expert skills.

It can be argued that if marks are mnemonic, then no self-revealing mechanism is needed. However, this argument is analogous to using mnemonic names for commands in command line interfaces. This technique relies on the user "being a good guesser" and it has been shown that they are not; command naming behavior of individuals is extremely variable (Furnas, et al., 1982; Carroll, 1985; Jorgensen et al. 1983; Wixon et al., 1983). The more fail-safe approach is to provide an explicit mechanism which explains the command set (Barnard & Grudin, 1988). On the

[6] Of course, even some of the direct manipulation components may require instruction.

other hand, other researchers have shown or argued that users commonly agree on certain marks for certain operations (Wolf, 1986; Wolf & Morrel-Samuels; Gould, & Salaun, 1987; Buxton, 1990). Nevertheless, if we wish to use marks for operations which do not have commonly agreed upon marks, a mechanism must be provided for learning about these marks.

We define three design principles to support learning and using marks. We do not claim that these principles are unique. Other researchers have described similar general principles, and many systems have interactions which obey these general principles. However, we define specific design principles for two reasons. First, our application of the general design principles to marks is novel, and second, our own specific definitions help us to explain and discuss the details of the application.

The three design principles to support learning and using marks are self-revelation, guidance, and rehearsal.

Self-revelation

> *The system should interactively provide information about what commands are available and how to invoke those commands.*

When an interface provides information to a user about what commands are available and how to invoke those commands, we refer to this as self-revelation or the system being self-revealing. Menus and buttons, for example, are self-revealing. The available commands and how to invoke those commands can be inferred from the display of menus or buttons. Marks, on the other hand, are not self-revealing because they must be generated by the user.

To ensure that every aspect of a system is self-revealing is a difficult task. For example, displaying menu items may help a user understand what functions are available but does not guarantee that the user will understand, from the display, the mechanics of selecting a menu item.

A common approach to interface design, and the approach that we adopt in this dissertation, is to rely on a user receiving a small amount instruction before starting to use the system. These instructions explain the basic mechanics and semantics of operating the interface. For example, pointing, dragging, double clicking, and the meaning of these actions may be explained. The Macintosh computer uses this

technique. The intention is that with this small set of skills a user can start interactively exploring and learning about the remainder of the system.

The interaction technique developed in this dissertation uses this type of design. A user must be informed, *a priori*, that in order to display a menu the pen must be pressed against the display and held still for a fraction of second. We call this "press and wait for more information". Once users have this bit of information, however, they receive further instructions interactively from the system. In our model of the interface, users can interactively learn about what functions can be applied to various displayed objects by "pressing and waiting" on the objects for menus.

The principle of self-revelation is based on interface design principles and psychological mechanisms proposed by others. Norman and Draper (1986) propose a design principle to "bridge the gulfs of execution and evaluation". Specifically, a designer should make interface objects visible so users can see what actions are possible, how actions can be done, and the effects of their actions. Shneiderman (1987) proposes a similar principle: "offer informative feedback". The principle states that objects and actions of interest should be made visible to the user. Shneiderman claims that this design principle is the basis of direct manipulation interfaces.

The principle of self-revelation is distinct from affordance theory (Gibson, 1979; Gibson 1982). Self-revelation is concerned with absence/presence of information about what functions are available and how to invoke those functions. Affordance theory, in human computer interaction, is concerned with an interface object's appearance suggesting its function (Gaver, 1991). These two notions, however, are related. For example, consider the display of a pop-up menu. The principle of self-revelation dictates, first, that function names or icons must be displayed, and, second, that they are displayed in a menu so that a user knows by convention how to invoke them. Affordance theory, on the other hand, dictates that the name or icon for an item accurately suggests its function, and that the appearance of the menu suggest items are selectable. Correct use of affordances may help reduce the amount of *a priori* instruction a user requires. For example, items in a menu may "look" selectable (they "afford" selection) and therefore the user does not have to be explicitly taught these mechanics.

Guidance

The way in which self-revelation occurs should guide a user through invoking a command.

If an interface actually assists a user in the articulation of commands we refer to this as guidance. For example, in the editor *emacs*, by hitting a "command completion key" while typing a command, emacs will display all the command names that match the partially completed command. In effect, emacs "guides" a user in completion of the command, as opposed to waiting for the command to be completely typed before examining its validity. Another example is selection from a hierarchic menu. In this case, selection of an item guides a user to the next menu.

Guidance does not necessarily have to be triggered by the user. Some on-line help systems prompt the user with information to guide them through a command. The critical point is that in these systems getting or receiving helpful information on how to invoke a command (guidance) does not interrupt the articulation of a command. On the other hand, a system like the on-line manual pages in UNIX violates the principle of guidance. In this case, in order to receive information about what commands are available and how to invoke those commands, a user must terminate or at least suspend the act of invoking a command.

Rehearsal

Guidance should be a physical rehearsal of the way an expert would issue the command.

Rehearsal is the notion of designing interactions such that the physical actions made by a novice in articulating a command are a rehearsal of the actions an expert would make invoking the same command. The goal of rehearsal is to develop skills in a novice that transfer to expert behavior. It is hoped that this leads to an efficient transition from novice to expert.

Many interaction techniques support rehearsal. When the basic action of the novice and the expert are the same for a particular function we can say that rehearsal takes place. For example, novices may draw lines, move icons, or select from menus using the same actions as an expert when there is one and only one way of issuing the command. In many cases, the single way of issuing the command may be suitable for both the novice and expert.

There are also many situations, however, where a single method for invoking a command is not sufficient. The popularity of accelerator techniques is proof of this. Typically, good interfaces provide two modes of operation. The first mode, designed for novices, is self-revealing. Conventional menu-driven interactions are an example of this. The self-revealing component of this mode is emphasized over efficiency of interaction because novices are more concerned with how to do things rather than how quickly things can be done. The second mode, designed for experts, typically allows terse, non-prompted interactions. Command line interfaces and accelerator keys are examples of this mode. However, usually there is a dramatic difference between novice and expert behavior at the level of physical action. For example, a novice uses the mouse to select from a menu whereas an expert presses an accelerator key.

The intention of the three design principles is to reduce this discrepancy in action without reducing the efficiency of the expert and ease of learning for the novice. The basic actions of the novice and expert should be the same. It is hoped that as novice performance develops the skills that lead to expert performance will develop in a smooth and direct manner.

1.3.2. Unfolding interfaces

The principles of self-revelation, guidance and rehearsal support the notion of an *unfolding interface*. An unfolding interface works as follows. Initially, a novice is provided with a small amount of information about how to get information on parts of the interface. For example, double clicking on an object may open it up or "unfold" it to reveal additional functions. Thus, given this key to unfolding objects, a user can explore the interface, learning and using new functions. The intention is that, with experience, exploration and use leads to expert knowledge of the system.

There are other schemes which control the number and types of functions available to a user, for example, *Training Wheels* (Carroll & Carrithers, 1984). These types of systems provide explicit novice/expert modes in which the novice mode has fewer functions than the expert mode. The intention is to avoid confusing a novice with a large set of complex functions. Once the reduced set of functions is mastered, the novice can switch to the larger "expert" set of functions. The major difference between this approach and the notion of an unfolding interface is that an unfolding

interface has no explicit novice and expert modes. An unfolding interface allows users to incrementally add functions to their repertoire.

Marks, self-revelation, guidance and rehearsal can play important roles in an unfolding interface. Unfolding is essentially an inefficient operation. As suggested earlier, by associating marks with "hidden" functions, unfolding can be avoided. For example, rather than double clicking on an object to unfold it and then clicking on a function button, a mark can be made on the object to invoke the function. To help users learn the marks associated with functions, it would be beneficial if unfolding a function also revealed its mark. This is an application of the principle of self-revelation. Ideally, we want the principles of guidance and rehearsal to hold as well; we want to design an interface such that exploration is equivalent to invoking commands, and exploration allows a novice to practice skills that lead to expert behavior.

1.3.3. Solution: ways of learning and using marks

The concerns of this research are interfaces that use marks but are also self-revealing. Therefore, solutions for making marks self-revealing can be classified by how tightly coupled the act of marking is with the act of getting information about command/mark associations.

Interfaces that use marks and only supply information about those marks through off-line manuals are considered to be at one end of a self-revelation continuum. These interfaces are not interactively self-revealing. Interfaces which supply information about marks as a command is actually being articulated can be considered the other end of the self-revelation continuum. These would be considered interactively self-revealing interfaces.

In the following sections we classify solutions based on this criterion. Since interfaces that use marks are still in their infancy there are few pre-existing examples.

Off-line documentation

Off-line documentation consists of manuals which provide information about how marks are used in an interface. Examples of the marks are displayed and text or graphics provides information on their usage. Although this type of scheme is not self-revealing it is of interest because, first, it is the status quo for pen-based

products and, second, it demonstrates the type of information needed for a user to understand marks.

Figure 1.2 shows a section from a pen-based system's manual. Clearly this type of scheme is not interactively self-revealing. However, if the mark set is small, the documentation could be placed directly on the computer in the form of a "cheat sheet". This scheme would be partially self-revealing.

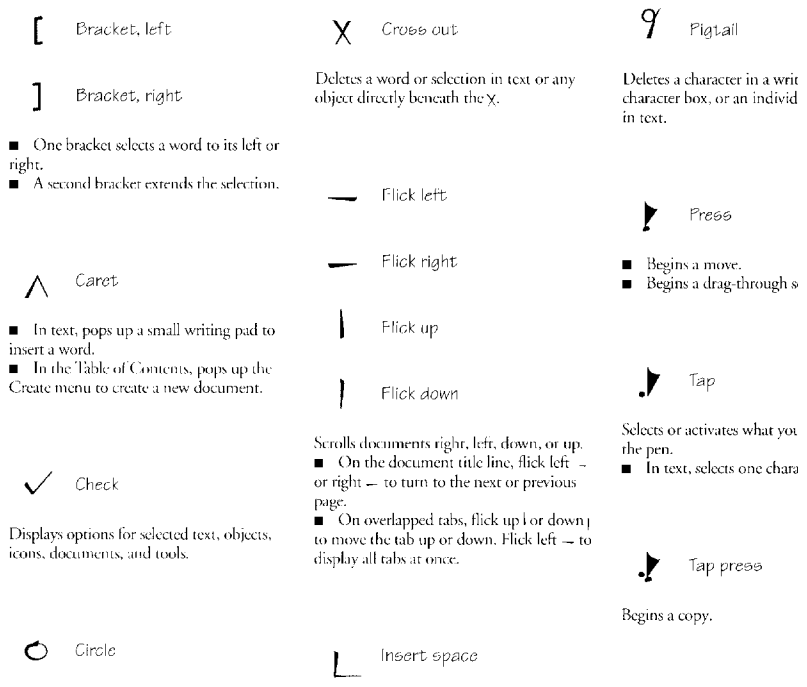


Figure 1.2: Typical off-line documentation for mark commands (PenPoint system, Go, 1991)

On-line documentation

This class is essentially the "on-screen" version of off-line documentation. A user can display manual pages on-screen while the application in question is running. Note that this does constrain the user into suspending the real task of issuing a command while obtaining command information.

Sometimes command information can be found in the application used to train the software module that recognizes marks. Figure 1.3 shows one such example.

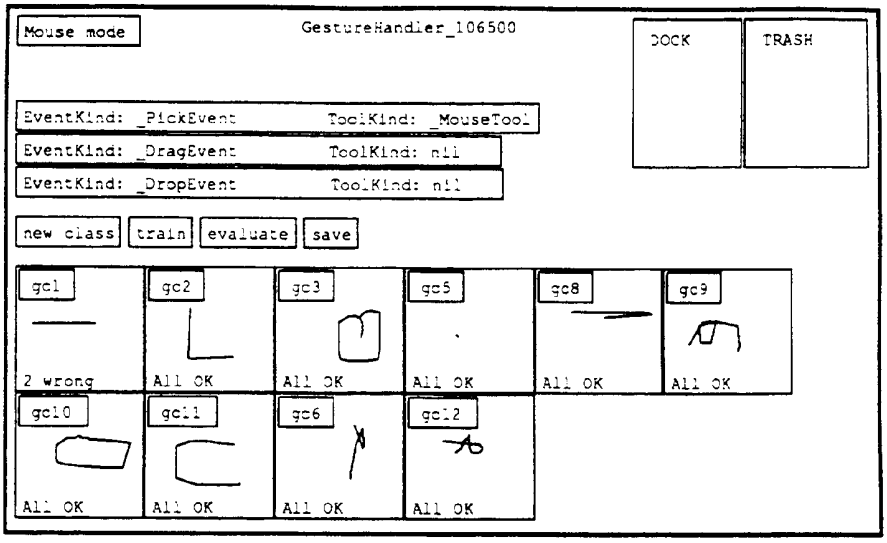


Figure 1.3: Gesture handler window allows inspection of marks associated with a view in Rubine's system. This window is, however, intended for the system programmer. The window shows ten classes of marks but does not shows the semantics associated with each mark. (from Rubine, 1990).

Unfortunately, training interfaces are not designed specifically to deliver this type of information, and the information can be very minimal and confusing to the user.

Microsoft's *Windows for Pen Computing* uses on-line documentation. A special application provides a tutorial which features animations demonstrating marks and editing operations. A user can also practice using the marks on sample text. While the tutorial is effective, a user still has to change context (i.e., switch from the working application to the tutorial application) in order to get information on marks.

On-line interactive methods

On-line interactive methods supply information about marks as one issues a command. Figure 1.4 shows an example where sample marks are displayed beside menu items. Windows for Pen Computing using this technique to a limited degree. This technique relies initially on another interaction method such as menus or buttons to invoke commands. In Figure 1.4, the interaction technique initially relied on is a menu. As the menu is used, it reveals the marks that can be used. Once a user remembers the mark associated with a command, the revealing technique (the

menu) can be bypassed and a more efficient mark can be used. Figure 1.5 shows a system called *XButtons* which also uses this method. In contrast to on-line documentation, an on-line interactive method does not constrain the user into suspending the real task of issuing a command, while obtaining command information.

This method is similar to accelerator keys. Every time a user uses a menu item or button, the mark is seen. Like accelerator keys, the mark can be memorized and used as a shortcut in calling the command. Note that "accelerator marks" are more powerful that accelerator keys because they are not limited to characters on the keyboard, they indicate the object of the requested action by the location of the mark, and they can contain command attributes, such as destinations or modifiers.

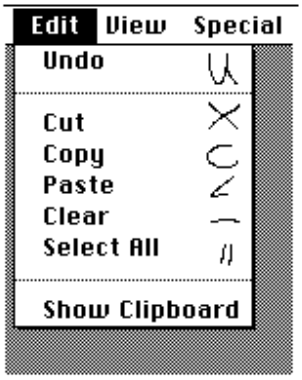


Figure 1.4: An example of "accelerator marks" which allow quick access to menu items similar to accelerator keys.

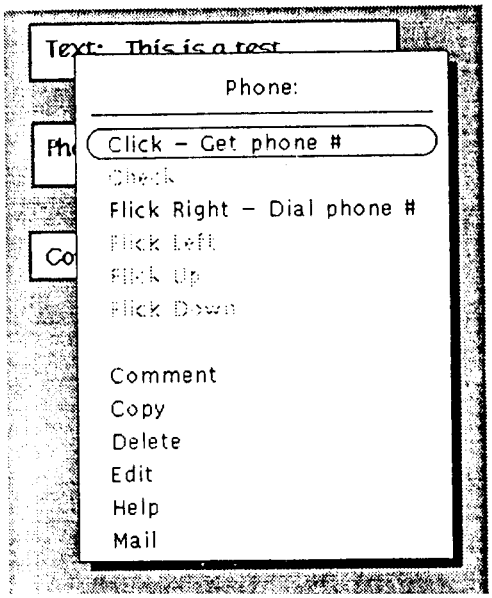


Figure 1.5: XButtons provides a menu which shows what commands are available from a button and the associated marks. A command can be invoked by either a menu selection or by making the mark on the button (Robertson, et al, 1991).

On-line interactive rehearsal methods

This category is similar to on-line interactive methods except invoking a command using the self-revealing technique (i.e., a menu) makes the user physically rehearse making the corresponding mark. In contrast, when using on-line interactive methods, the user does not physically rehearse making the mark (e.g., selecting "copy" from the menu in Figure 1.4 requires a vertical movement, not a hand drawn "C" movement).

Marking menus, the technique focused on in this dissertation, is an example of this class (Kurtenbach & Buxton, 1991). The complete definition of this technique is given in Chapter 2. Figure 1.6 illustrates this technique in the context of creating three simple objects. An expert uses simple shorthand marks to create and place circles, square, or triangles.

If a user is unsure of what marks can be made, the user presses the pen against the display and waits for approximately 1/3 of a second. This signals to the system that no mark is being made and it then prompts the user with a radial menu of the available commands, which appears directly under the cursor. The user may then select a command from the radial menu by keeping the pen tip pressed and making a stroke towards the desired menu item. This results in the item being highlighted (see Figure 1.7). The selection is confirmed when the pen is lifted from the display.

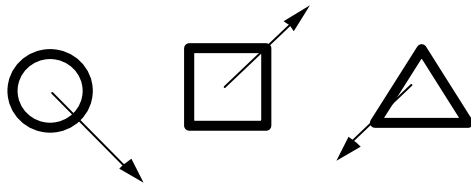


Figure 1.6: An example of the technique using three simple shorthand marks. Three objects can be defined: a circle, square and triangle. A mark which is a simple straight line (shown here with an arrowhead to indicate drawing direction) defines the type of object created, and its placement.

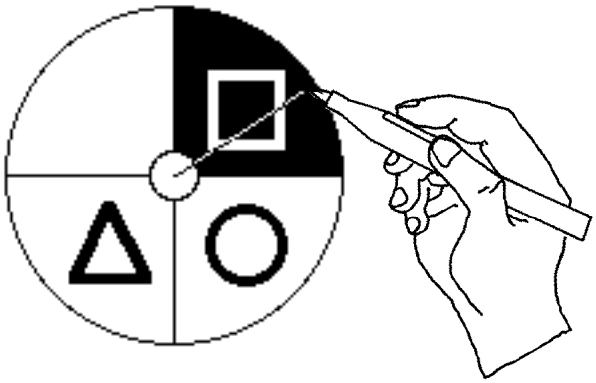


Figure 1.7: A radial (or "pie") menu can also be popped up if the user does not know what commands or marks are available. Rather than drawing a mark as in Figure 1.6 a novice keeps the pen pressed and a menu appears. An object can then be selected from the menu.

The important point is that the physical movement involved in selecting a command is identical to the physical movement required to make the mark corresponding to that command. For example, a command that requires an up-and-to-the-right movement for selection from the pie menu, requires an up-and-to-the-right mark in order to invoke that command. The intention is that selection from the menu is a rehearsal of making a mark.

Other menu layouts can be used for interactive rehearsal methods besides radial menus. Another possibility is a "bull's eye menu" which is a menu that is divided into concentric circles rather than sectors, where each concentric circle corresponds to a different command (Figure 1.8).[7] The corresponding marks are therefore discriminated by length rather than angle. Many more exotic schemes have been proposed and are as of yet unexplored.[8] Chapter 2 presents the motivation for choosing radial menus, and describes in detail the design of marking menus.

---

[7] We thank Professor John W. Senders for this suggestion originally called "donut menus". Professor William Buxton later took great exception to the use of the word "donut" and suggested the more dramatic name of "bull's eye menu".

[8] Dr. Tom Moran has proposed a combination of donut and pie menus. Dr. Stuart Card has proposed a continuous version of hierarchical marking menus.

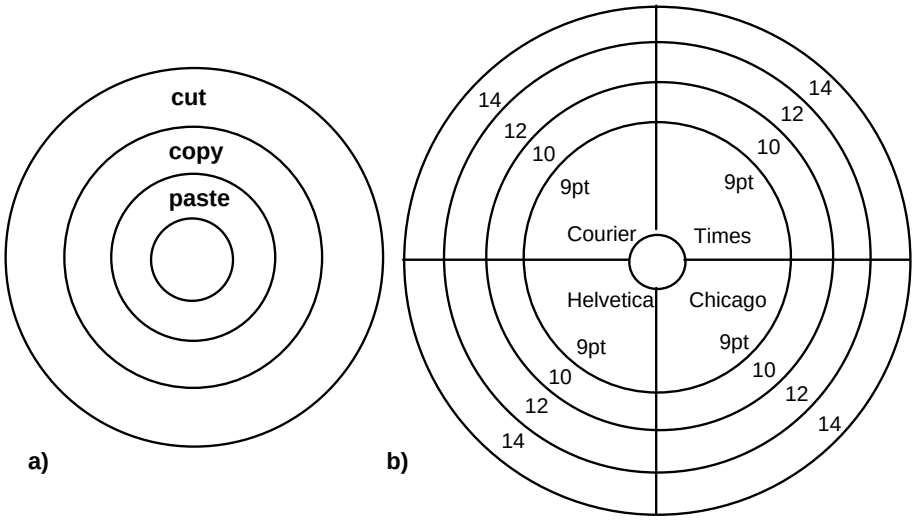


Figure 1.8 Examples of alternate menu styles in which selection will result in a unique marks. a) is a "bull's eye" menu which discriminates by mark length rather than angle. b) is a "dart board" menu which discriminates by length and angle.

1.4. THESIS STATEMENT

This dissertation is an in-depth investigation of marking menus. We present the thesis that marking menus are a valuable interaction technique. When used in the proper situation, marking menus are easy and efficient to use, can be used with different input devices, and integrate well with existing interface techniques. Furthermore, marking menus allow a user to take advantage of writing skills with a pen and attain levels of performance not possible with other interaction techniques. To support this thesis, we present a design for marking menus, evaluate marking menus by means of user behavior experiments, and provide a case study of marking menus in practice. We conclude our investigation by showing how the design concepts of marking menus, self-revelation, guidance, and rehearsal, can be generalized to other situations.

The intention of this investigation is to provide practical guidelines for interface designers interested in using marking menus. With this in mind, we describe when and where marking menus would be an effective technique, and the limitations and properties that must be observed and maintained for marking menus to work well in an interface. We also describe the design principles behind marking menus and give examples of how these principles can be applied to other contexts.

1.5 SUMMARY

This chapter has provided motivation for marks as an interaction technique, described a basic interface problem with marks, set out design principles to solve this problem and introduced an approach, marking menus, which observes these design principles. In Chapter 2 we expand on our motivation for using marking menus and explain in detail the design and design rationale behind marking menus. Chapter 3 reports on an empirical study of the non-hierarchic marking menus. Chapter 3 is a condensed version of a paper that appears in *Human Computer Interaction* (Kurtenbach, Sellen, & Buxton, 1993). Chapter 4 is a case study which reports on how marking menus can be designed into an application and investigates user behavior with marking menus in an "everyday work" situation. Chapter 5 presents an empirical study on the limits of user performance with hierarchic marking menus. Chapter 5 is an expanded version of a paper published in *The Proceedings of InterCHI '93* (Kurtenbach & Buxton, 1993). Chapter 6 describes how we integrated marking menus into a pen-based application and applied the notions of self-revelation, guidance and rehearsal to this application. Chapter 7 summarizes this dissertation and its contributions, and proposes future research.

Chapter 2: Marking menus

In this chapter we expand on our description of marking menus. First, we present a definition of marking menus and the motives for investigation. Next, we describe previous research that is related to marking menus and we identify open research questions and the issues pursued in this dissertation. Finally, we complete our description of marking menus by providing the complete rationale behind our design.

2.1. DEFINITION

A *marking menu* is an interaction technique that allows a user to select from a menu of items. There are two basic ways (or modes) in which a selection can be performed:

menu mode In this mode a user makes a selection by displaying a menu. A user enters this mode by pressing the pen against the display and waiting for approximately 1/3 of a second. We refer to this action as *press-and-wait*. A *radial menu* of items is then displayed centered around the pen tip. A radial menu is a menu where the menu items are positioned in a circle surrounding the cursor and each item is associated with a certain sector of the circle. A user can select a menu item by moving the pen tip into the sector of the desired item. The selected item is highlighted and the selection is confirmed when the pen is lifted from the display. (See Figure 2.1)

mark mode In this mode, a user makes a selection by drawing a mark. A user enters this mode by pressing the pen against the display and immediately moving in the direction of the desired menu item. Rather than displaying a menu, the system

draws an ink-trail following the pen tip. When the pen is lifted, the item that corresponds to the direction of movement is selected. (See Figure 2.1)

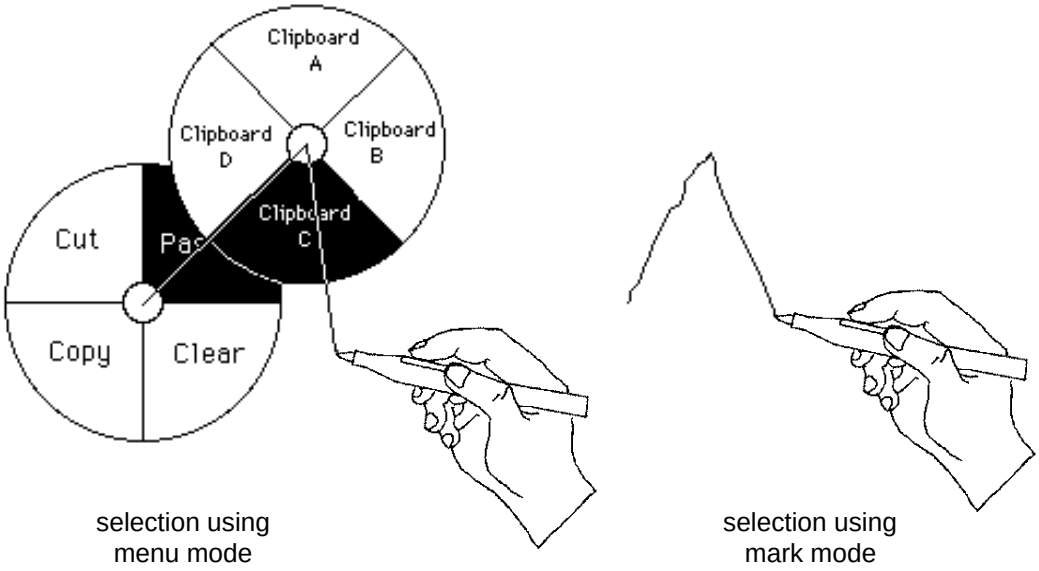


Figure 2.1: The two basic ways of selecting from a marking menu.

The key concept of marking menus is that the physical movement involved in selecting an item in menu mode mimics the physical movement required to select an item using a mark.

Marking menus may also be hierarchic. In menu mode, if a menu item has a subitem associated with it, rather than lifting the pen to select the item, the user waits with the pen pressed to trigger the display of the submenu. The submenu is also a radial menu. The user can then select an item from the submenu in the manner previously described. In mark mode, a user makes a selection by drawing a mark where changes in direction correspond to selections from submenus. Figure 2.1 show an example of selecting from hierarchic menus using menu mode and mark mode.

Using radial menus in this way produces a set of mark which consist of a series of line segments at various angles ("zig-zag" marks). Marking menus which have no hierarchic items produce strictly straight line segments. Figure 2.2 shows an example of a menu hierarchy and the associated marks.

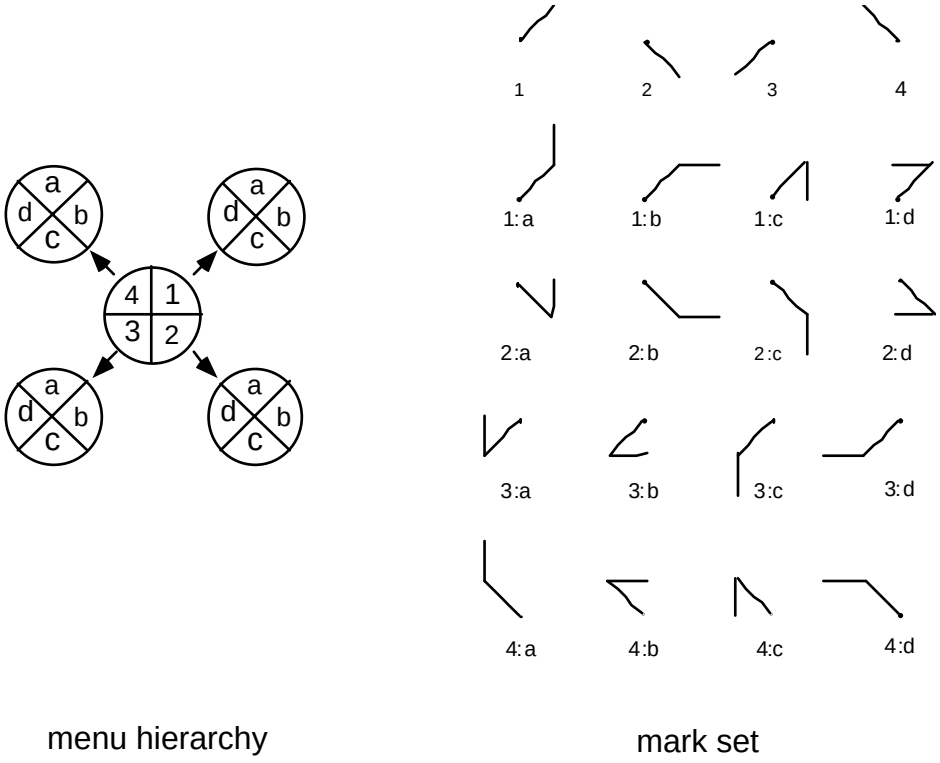


Figure 2.2: An example of a radial menu hierarchy and the marks that select from it. Each item in the numeric menu has a submenu consisting of the items a, b, c and d. A mark's label indicates the menu items it selects. A dot indicates the starting point of a mark.

It is also possible to verify the items associated with a mark or a portion of a mark. We refer to this as *mark confirmation*. In this case a user draws a mark but presses-and-waits at the end of drawing the mark. The system then displays radial menus along the mark "as if" the selection were being performed in menu mode. Figure 2.3 shows an example of this.

Other types of behavior can occur when selecting from a marking menu such as backing-up in a menu hierarchy or reselecting an item in menu mode. Details of the behavior are discussed in Section 2.5.

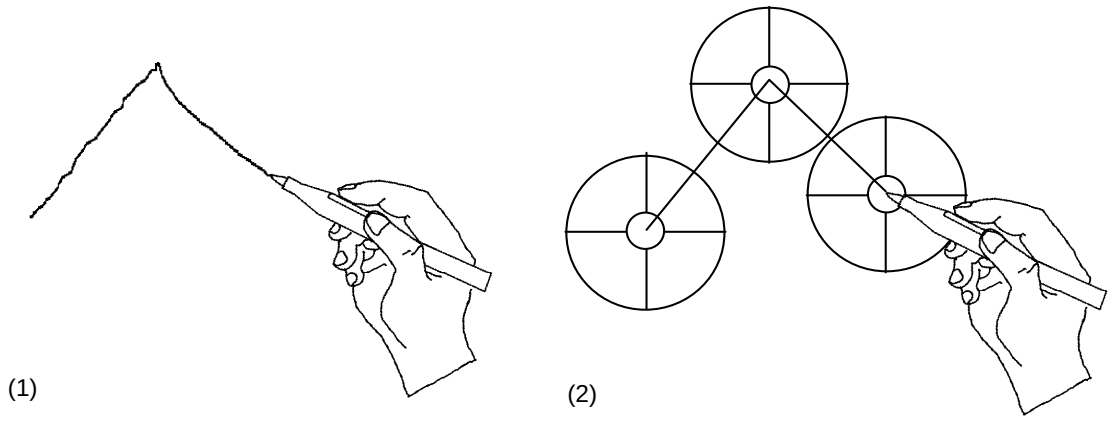


Figure 2.3: An example of mark-confirmation in a menu with three levels of hierarchy. In (1), the user draws the first part of the mark then waits with the pen pressed for the system to recognize the selection so far. In (2), the system then displays its interpretation of the mark and goes into menu mode for completion of the selection.

2.2. MOTIVATION FOR STUDY

We have many motives for studying marking menus; they have advantages over traditional menus; they use marks that are easy to draw and that are easy for computer to recognize; they can be used for functions that have no intuitive mark; they are compatible with different interface styles; and they exploit human motor skills. In this section, we expand on these motivations.

2.2.1. Advantages over traditional menus

One motivation for studying marking menus is that they have many differences and potential advantages over the traditional menus used in current practice. Examples of the current practice in menu design are the pop-up menus or pull-down menus on the Macintosh. With these types of menus, selection is performed by popping up the menu and selecting items by pointing with the mouse. Menu items can also be selected by pressing an accelerator key associated with a menu item. There are several specific advantages marking menus have over these traditional menus:

Keyboardless acceleration

Marking menus allow menu selection acceleration without a keyboard. With traditional linear menus, keypresses must be used to accelerate selection. Marking

menus provide a method of accelerating menu selections when no keyboard is available. This is extremely important for portable, keyboardless, pen-based computers.

Acceleration on all items

Marking menus, if configured accordingly, can permit acceleration on all menu items. With traditional menus, it is common for the application developer to assign accelerator keys to the most frequently used menu items. This assumes that the application designer is able to predict the most frequently used menu items. In many cases, however, it is not possible to accurately predict which menu items will be frequently used, if there is a large variance in the way an application may be used. In contrast, with marking menus, the selection of all items can be accelerated by the user making a mark. The designer does not have to predict, *a priori*, which items will be the most frequently used.

Menu selection mimics acceleration

Marking menus minimize the difference between the menu selection and accelerated selection. Selecting a menu item from a marking menu physically mimics the act of making the accelerating mark. The design intention is to help users become skilled at the movements required for accelerated menu selection. This is dramatically different from traditional menus and accelerator keys where menu selection is performed with the mouse and accelerated selection is performed with the keyboard. In this case selection from the menu in no way physically mimics selection using an accelerator key.

Combining pointing and selecting

Marking menus permit pointing and menu selection acceleration with the same input device. This is an intrinsic property of marks and has been utilized by other researchers (e.g., Coleman, 1969; Rhyne 1987; Wolf & Morrel-Samuels, 1987). In mouse-based direct manipulation interfaces it is very common to point to an object and then select a menu item. If accelerator keys are used, this operation requires coordinating pointing with the mouse and pressing on the keyboard. With a marking menu, not requiring a hand to be on the keyboard frees the hand to control other input devices or perform auxiliary tasks such as controlling a VCR transport or turning the pages of a book.

Spatial mnemonics

Marking menus use a spatial method for learning and remembering the association between menu items and marks. In contrast, traditional menus and accelerator keys, rely on symbolic mnemonics to help users remember the associations between menu items and keys. Due the limited number of symbols on a keyboard, mnemonics often cannot be established between all menu items and their accelerators keys. This results in menu item/key associations that may be arbitrary or inconsistent. Marking menus avoid this problem by relying on a consistent method to establish mnemonics: the shape of a mark corresponds to the spatial layout of a menu item in the menu hierarchy.

2.2.2. Ease of drawing and recognition

Marking menus use a very simple set of marks consisting of straight and zig-zag marks. This simple set of marks has three advantages. First, these types of marks are easy and fast to draw and are therefore suitable for accelerated performance. Ease of drawing is especially important when drawing precision is hampered by imperfect pen/display technology. Second, computer recognition of these types of marks can be reliable, fast and user independent. The recognizer requires little processing power and no training. Third, any interface designer, by using marking menus, can make use of some of the advantages of marks without having to design their own mark symbols. Of course, it is still necessary to design the layout of the menus.

The single contiguous marks in marking menus have several advantages. Other types of marks which require multiple non-contiguous pen strokes create many problems. Recognizer design is more complicated when groups of strokes must be recognized. This is referred to as the *segmentation problem*. Sometimes groups of strokes are distinguished by constraining the user to put all the strokes associated with a mark in a certain region. Alternatively, strokes may be grouped by time. This constrains the user to momentarily pause between making different marks. With a marking menu mark, a user is not constrained by timing, size of mark, or location. Recognition takes places the moment the pen is lifted.

The marking menu mark set does have disadvantages. First, a designer has no choice in the shape of the marks (besides what can be controlled through the layout of the menus). Fortunately, marking menus do not prohibit the use of other mark

sets and mark recognition techniques (see Chapter 6 for a detailed discussion of this issue). Second, the size of the mark set is limited by a user's accuracy at drawing lines at various angles. Third, the mark set is not particularly expressive. The angle at which the stroke is drawn is used to define the type of mark. The line must also be somewhat straight. This leaves starting point, ending point and temporal information about how the line was drawn to be used as additional information encoding parameters. In contrast, other mark vocabularies permit many more parameters to be controlled by the shape of the mark (Makuni, 1986). Nevertheless, we have discovered that the limited set of parameters of a marking menu mark can be quite useful (see Chapter 4).

2.2.3. Marks when no obvious marks exists

Researchers have shown or argued that users commonly agree on certain marks for certain functions (Wolf, 1986; Gould, & Salaun, 1987; Morrel-Samuels, 1990; Buxton, 1990). However, we believe that there are many situations where invoking a function with a mark could be beneficial but no commonly agreed upon mark exists for the function. This is similar to icon design where some functions have no intuitive icon. For example, there is no "natural mark" for "change pen width to thin". Marking menus might work well in these types of situations because the menu can provide textual or pictorial explanations of functions while the mark for the menu item provides a quick way to invoke the function.

2.2.4. Compatibility with unfolding interfaces

Marking menus are compatible with unfolding interfaces (described in Section 1.3.2). The intention is that menus pop up to self-reveal or unfold functions and the marks provide way to efficiently invoke the functionality. Guidance and rehearsal are intended to help a novice learn the efficient way of invoking a function.

2.2.5. Compatibility with existing interfaces

Marking menus are compatible with popular input devices and interface paradigms. First, the type of marks used can be reasonably drawn with a mouse (Chapters 3 and 5 explore this issue in detail). Second, since traditional menus are created by the application calling library routines, by replacing the library routines, marking menus could be used in place of pop-up menus without changing a single line of application code or changing application functionality. Finally, marking menus can

extend existing dialogue styles without major changes to an interface paradigm. An example of this is *HyperMarks*, developed by the author (Kurtenbach & Baudel, 1992), which is a *Hypercard xcommand* that supports marking menus in Hypercard (Apple Computer, 1992). When a marking menu is used from a Hypercard button, the Hypercard button still retains its single function when pressed. However, if the button is kept pressed, a marking menu pops up with more commands. A user can select from the marking menu using menu mode or marks. In this way, the function of a button can be extended.

Marking menus can be effective because they are a pop-up interaction technique. When displays become small or very large, marking menus can be effective. On large displays, a mark or a menu selection can be made at a user's current location without a long trip to a menu bar or tool pallet. On small screens, since both the menu and mark "go-away" once performed, no valuable screen space is consumed.

2.2.6. Novices, experts, and rehearsal

Marking menus are intended to support both the novice and expert user. The intention is that a novice uses menu mode and an expert uses the marks. Menu mode can provide the self-revelation and guidance needed for a novice to invoke a command. The marks can provide efficient interactions for experts.

Marking menus are also intended to support the transition between novice and expert. Selection in menu mode provides the user with rehearsal for making a mark. In essence, using the menu trains a novice to use marks. We believe that rehearsal helps in learning the association between mark and command.

There are other menuing schemes which support the novice and expert and the transition between the two. For example, the Macintosh supports novices by providing menus and supports experts by providing menu accelerator keys. The transition between novice and user is supported by the user being reminded of the keystrokes associated with particular menu items every time a menu is displayed. This is done by having the names of the accelerator keys appear next to menu items in the menu. However, actually using an accelerator key is avoidable. The user can always just select from the menu. Furthermore, this is easiest because the user is already displaying the menu. The end result is that accelerator keys are sometimes not used even after extensive exposure to the menu. With marking menus the user is not only reminded, but rehearses the physical movement involved in making the

mark every time a selection from the menu is made. What makes marking menus unique from the accelerator key scheme is that rehearsal is unavoidable. We believe this helps in learning the association between mark and command.

2.2.7. Utilizing motor skills

The idea of using physical rehearsal to train novices to become experts is a unique concept and is worth investigating for pedagogical reasons. Marking menus purport to reduce the cognitive load of memorizing mark/command association by relying on muscle memory (since each mark/command is a distinct physical movement). This technique is similar to the approach used in the Information Visualizer Project (Card et al, 1991). The Information Visualizer relies on low level sensory input processing such as depth or motion perception to reduce the burden on higher cognitive processes in visualizing information. Marking menus can be thought of in a similar manner. It is believed that low level sensory output processes (muscle memory) are used to reduce the load on higher level cognitive processes. We explore this issue in this dissertation.

2.2.8. "Eyes-free" selection

Selection by a distinct physical movement with a marking menu lends itself to "eyes-free" selection. For example, most of us can draw the eight directions of a compass without looking. Eyes-free selection is useful in situations where a user's visual attention must be on something other than the selection process, for example, selecting commands while watching a video tape. An eyes-free selection technique is also extremely valuable to the visually impaired.

2.3. RELATED WORK AND OPEN PROBLEMS

This dissertation develops and explores the use of marking menus. There is no previous research on this technique, *per se*, however, marking menus are based on radial menus (see Section 2.1 for the definition of radial menus). Therefore, research on radial menus is relevant. The most widely used instance of a radial menu is the pie menu (Hopkins, 1991). A pie menu is a radial menu where the visual representation of the menu resembles a sliced pie. Other types of visual representations are possible, for example, we have developed an alternative representation for a radial menu which does not look like a pie (see Figure 2.12).

Two instances of radial menus are pie menus and command compasses. We now describe these two techniques, contrast them with marking menus, and report on the current state of research on their design and usage.

2.3.1. Pie menus

To date, there is little research on pie menus. The origin of pie menus can be traced back to radial menus proposed by Wiseman, Lemke, & Hiles (1969). Since then, research on pie menus has mainly been concerned with menu layout and suitable applications (Hopkins, 1991; Hopkins, 1987). The only empirical study of pie menus investigated menu item selection time and error rates for 8-item menus but concentrated on comparing them to linear menus (Callahan, Hopkins, Weiser, & Shneiderman, 1988). It was found that selection from pie menus was significantly faster (15%) and produced marginally significant fewer errors (42%) than linear menus. The experiment also investigated the effect of using menu items with a natural linear ordering (i.e., "First", "Second", "Third", etc.), with a natural radial ordering (i.e., "North", "North-east", "East", etc.), and with an unclassifiable ordering (i.e., "Center", "Bold", "Italic", etc.). Callahan et al. hypothesized that certain types of menus (pie or linear) would perform better with items that have a certain type of natural ordering (radial, linear, or unclassified). A marginally significant correlation was found between menu types and types of orderings. The weak correlation occurred because selection time means for the pie menus were lower even on items with natural linear orderings. Results also showed that unclassified menu items produced significantly slower selections than ordered menu items regardless of menu type.

What has not been extensively studied is the claim that muscle memory for different gestures plays a helpful role in menu selection. Anecdotal evidence from designers of pie menu systems suggest that item selection from a menu hierarchy is possible without displaying the menus after practice (Hopkins, 1987). Not only was unprompted selection possible but it was also desirable for efficiency reasons.

Unprompted selection is supported in pie menus by a technique called *mousing-ahead*. Mousing-ahead means the user does not have to wait for the system to display the menu before moving the cursor to make a selection. As the user moves the cursor, the input system buffers cursor location data. When the menu is finally displayed, the system reads the buffered data and analyzes it as if it were generated

with the menu displayed. The system then immediately selects a menu item and removes the menu. In this way a user can make a selection without waiting for the menu to display (in effect, the mouse is being operated "ahead" of the display, hence the term mousing-ahead). Hopkins' implementation is slightly more sophisticated than just described. Menu display is suppressed until the user stops moving the cursor.

On the surface, it appears as if a marking menu is a pie menu with an ink-trail added to cursor. However, there is a major difference in the way the two techniques behave. Marking menus, depending on the context, may use sophisticated recognition. Marking menus analyze the path of a cursor as a mark, looking for certain features. If the interface recognizes other types of marks, a mark has to "look like" a marking menu mark before it can select from the menu. For example, suppose an interface recognizes a "C" mark (e.g., "C" triggers the copy command) and also marking menu marks (i.e., zig-zag marks). If mousing-ahead was used, the "C" would select the bottom item of a menu (assuming the user started drawing from the top of the "C"). With marking menus, the recognizer identifies the mark as a "C" and not as a zig-zag mark. Chapter 6 discusses in more detail, issues of integrating marking menu marks with other types of marks.

As a consequence of mark recognition, marking menu marks can be performed more casually than mousing-ahead movements with pie menus, especially with hierarchic menus. Mousing-ahead on pie menus must be an exact imitation of cursor movement used when selecting with the menu displayed. Marking menus, on the other hand, recognize the shape of the mark, independent of size and therefore the user can be more casual when drawing marks as opposed to mousing-ahead. There are designs where mousing-ahead can be made independent of movement size but, in general, this is not possible. See Section 2.5.6 for a detailed discussion of these issues.

The visual difference between marking and mousing-ahead is that marking leaves an ink-trail after the cursor, whereas mousing-ahead does not. We believe that, without an ink-trail during selection, a user must visualize selection from the menu. With an ink-trail, the user does not have to visualize selection, but rather remember the mark associated with a menu item and then correctly draw the mark. We believe the ink trial provides feedback which helps the user to correctly draw the mark.

2.3.2. Command compass

An interface mechanism very similar to a marking menu is the command compass used in the Momenta pen-based computer. Figure 2.3 shows how the command compass is used to move text.

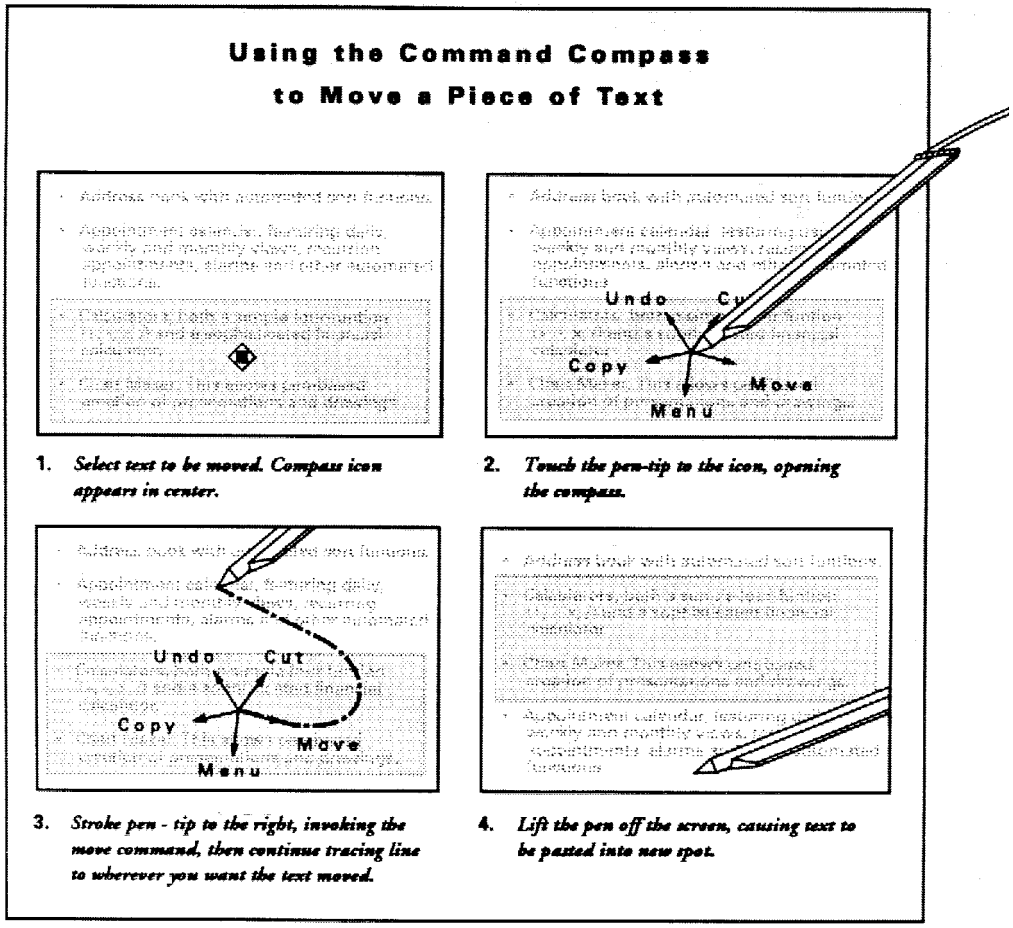


Figure 2.3: The Momenta Command Compass (Momenta, 1991).

There are several differences between the command compass and marking menus. First, the command compass does not permit reselection. Once the pen is moved in the direction of a command, that command is immediately selected. Second, an explicit unprompted selection mode is not provided. No ink-trail is provided and unprompted selection relies on mousing-ahead (or "penning-ahead", since Momenta is a pen-based computer). While the Momenta interface uses marks, the

command compass does not utilize marks. Finally, only one type and size of command compass is used. No hierarchic command compasses are supported.

The subtle difference in the way selection is done with a command compass versus selection with a marking menu affects the type of interactions each technique can support. With marking menus command selection occurs after a sector has been moved into and the pen lifted. With the command compass, command selection is done the moment a sector is moved into. Thus when selection occurs, the user is still in a physical mode (keeping the pen pressed). This physical mode can be used to express more parameters for the command, hence, physically pairing a command verb and its parameters. This is, of course, at the expense of not permitting reselection.

2.4. RESEARCH ISSUES

The ultimate goal of this research is to create a useful interaction technique. To attain this goal, several things must be accomplished. First, we must create a design for marking menus. Next, this design must be evaluated to determine its limitations and possible applications. From these evaluations, we can refine our design and develop recommendations for interface designers about when, where, and how marking menus can be beneficial. Given these goals, research issues surround the following question: what characteristics of marking menus do we need to understand to effectively incorporate this mechanism into the interface?

The most immediate question about marking menus is: how many items can be placed in the menus before it becomes too difficult to make selections using marks? Common sense tell us that parameters governing this aspect are articulation accuracy (i.e., how precisely can a human draw directional strokes), and human memory limitations (i.e., how quickly can a human learn and remember associations between menu items and marks). Other issues concern how hierarchic structure affects selection performance, how command parameters can be attached to marks, and how the design can be varied to accommodate the constraints of an interface. The following sections expand on these issues.

2.4.1. Articulation

Accuracy in selecting menu items and in marking is limited by the human motor system and the input device being used. This constrains the number of items that can be placed in a marking menu. *Articulation* refers to the motor system activities associated with selecting from a menu or making a mark, not memory activities like recalling the mark associated with a menu item. For example, suppose a user remembers the mark for a desired menu item. Can the user draw the mark accurately enough to select the menu item? In other words, can the user successfully articulate the mark once it is remembered?

Many factors may affect the success of articulation:

The type and characteristics of the input device. While the pen appears to be a natural input device for marks, operating marking menus with other types of input devices is also desirable. Thus, it is of interest to study users' performance not only with a pen but also with other popular types of input devices.

The number of items in a menu. As the number of items in a menu increases, the size of the menu items decreases and therefore pointing to them will become more error-prone and slower. Using a mark for selection should behave in a similar fashion. Precision of marking must increase as the number of items increases.

The type of articulation feedback provided. Feedback helps a user verify that a selection is being successfully articulated. For example, highlighting a menu item provides feedback. Supplying an ink-trail is another form of feedback, but is perhaps less salient. Finally no ink-trail (i.e., just the pen's or cursor's movement) provides even less feedback.

Chapters 3 and 5 investigate the effect of these factors through empirical experiments which measure speed and accuracy of selection when using marking menus. The results from these experiments are then interpreted to produce design guidelines.

2.4.2. Memory

Another aspect of marking menus concerns human memory. Using a mark to select from a marking menu involves, first, learning the association between menu item and mark, and then, recalling the association from memory before articulating the

mark. There are several ways in which learning and recall can occur. For example, a user can memorize the association by rote memory ("this mark invokes this command"), or a user can reconstruct a mental image of the spatial layout of the menu or process of selection.

There are other factors affecting learning and recall. Differences in the angles between items must be memorable enough so the angle can be reproduced in drawing the mark. For example, a user may remember an item was the third from the top in a very densely packed menu, but the angular difference between items may be so small that it cannot be remembered precisely enough.

Whatever technique is used to remember the mark/item association, the exact limitations of marking menus relative to the limitations of human memory is a very complex question. Human memory in some situations can be considered almost infinite. For example, humans are capable of memorizing many complex symbol systems such as languages. With enough practice, the paths through extremely complex hierarchies of menus could be memorized and recalled. The question of how quickly one "learns the marks" depends on many variables: frequency of use, presence or absence of mnemonics or metaphors, menu layout, intelligence, motivation, application, etc.

Determining hard figures for "learning time" or "maximum number of items" relative to human memory is not possible. These measures depend largely on the user and the application. The intent of this research is to come up with guidelines that help designers exploit aspects such as frequency of use, metaphors, and menu layout to help make marking menus easier to learn.

In the case of marking menus note that training time is not as critical as with other interface techniques because a user "trains on the job". A user of marking menus does not have to spend time training before the selections can be performed. A novice can use the menus while a forgetful expert may occasionally have to use the menu. In either case, the user will still be performing "training on the job".

Do users learn and use marking menus the way the design suggests? The three modes of interacting with a marking menu (menu, mark-confirmation and mark modes) are intended to support the transition from problem solving to skilled behavior in a user. Card, Moran and Newell (1983) suggest that novices exhibit problem solving behavior ("how do I do this?") and experts exhibit skilled behavior

(an expert knows how to solve the problem and does it efficiently). Rasmussen (1984) further refines this notion to include a middle step called rule-based behavior. Informally, rule-based behavior can be thought of as the user explicitly thinking "in order to do this I must do this". As Figure 2.3 shows, these stages of behavior can be mapped to the three modes of marking menus. The intention is that these modes are designed such that use of one mode builds the skills for the next mode and this assists in making the transition between modes. Do users actually behave this way with marking menus? If not, what sort of behavior is occurring and why?

We examine these issues of learning and remembering through empirical experiments (Chapter 3 and 5), and user behavior case studies (Chapter 4). The empirical experiments reveal learning curves and insights into the sort of menu structures that assist in learning and remembering menu layout and marks. A case study of user behavior using marking menus in a real application investigates learning and behavior patterns when marking menus are used in "everyday work" situations.

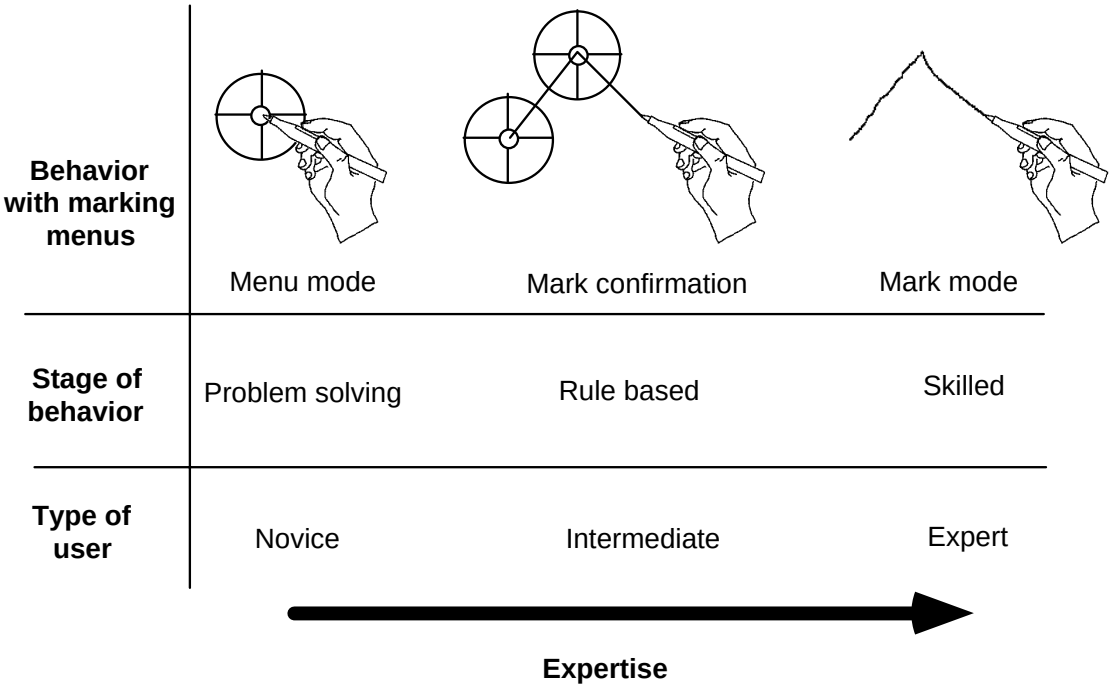


Figure 2.3: The relationship between stages of behavior, type of user, a user's behavior with a marking menu and expertise.

2.4.3. Hierarchic structuring

Another question concerns the effect that the structure of the menu hierarchy has on user performance. Specifically, how is user performance affected when breadth or depth is increased? (Depth is the number of levels in a hierarchy of menus; breadth is the number of items in a menu.)

Most of the research on hierarchic structuring of traditional menu systems focuses on depth versus breadth. This research can be divided into two types of studies: (1) theoretical models describing menu structure and user performance, and (2) empirical studies of menu usage. The theoretical studies concern models that describe menu search performance based on structure. From these models, structures that optimize search-time can be produced. The empirical studies attempt to verify the theoretical models, and estimate search time and error rates. These research efforts have addressed some basic issues concerning depth versus breadth.

The *navigation problem* (getting lost or using an inefficient path to find a menu item) becomes more likely as depth increases. Snowberry, Parkinson and Sission (1983) showed that error rates increased from 4% to 34% as menu depth increases from one to six levels.

Despite the problem of errors, there are several reasons to increase menu depth: *crowding*, *insulation* and *funneling*. Crowding refers to the problem of not having enough space on the screen to simultaneously display all the menu items. Insulation refers to the hiding information in deeper menus to protect a user from information overload. Funneling refers to the structuring of menus such that the hierarchy helps a user "narrow down" the choice and access items more quickly than using a flat menu structure.

Lee and MacGregor (1985) examine the tradeoff between funneling and response-execution time. Assuming all items were viewed before a selection is made, they found that optimal breadth was between 3 to 8 items per menu level depending on user response time and computer processing response time. Depth was effective when user response times were fast and computer processing time per option was slow. If it is assumed that the search terminates on average halfway through the items, then the optimal breadth is between 3 to 13 items at each level. These results

should be tempered by the fact that they are based on a theoretical model and not on empirical user tests.

If meaningful groupings of items are used, Paap and Roske-Hofstrand (1988) show that optimal breadth at any level tends to be in the range of 16 to 36 and sometimes as high as 78 for traditional menu systems depending on human and computer response time. In terms of marking menus, these ranges are well outside the maximum number of items that can be selected with a mark. This raises the issue that reduction of breadth in a marking menu may increase the performance of marking but degrade the efficiency of menu selection in the menu mode.

Menu search time increases monotonically with depth (Landauer & Nachbar, 1985). This produces a log-linear relationship between search time and number of menu items. Kiger (1984) also found that performance (time and accuracy) decreased as depth increased further confirming that depth presents navigation problems to users.

Kiger also included error recovery in his analysis. This increased the variance in search time from 6 seconds to 20 seconds. Since error recovery occurs in the real world, this study more realistically characterizes the costs associated with hierarchical structuring. Kiger tested five types of hierarchical structures varying the depth from two to six levels and the breadth from two to eight items.

Performance can vary at different levels of the hierarchy. Snowberry, Parkinson and Sission (1983) report on error rate versus hierarchy level in a six level hierarchy. A higher proportion of errors occurred at the top two levels of the hierarchy than at the bottom two despite the fact that every level was a binary choice. The explanation for this is that higher level items are more abstract and therefore more subject to misinterpretation. Kiger also found that search times gradually become faster as a user came closer to the goal item. Other studies have revealed opposite results—better performance occurred at top levels (Allen, 1983). The explanation offered for the differences is that users were much more familiar with the top level items than the lower level items. This lends support to the notion that performance, structure, and item semantics in menus are intimately related.

Paap and Roske-Hofstrand (1986) point out that users restrict navigation because the menu structure has semantics or because they have experience with the menu. Both Card (1982), and McDonald, Stone, & Liebelt (1983) report that effects of

organization disappear with practice. In other words, with practice, users navigate directly to the desired menu item. With experience, users move from a state of great uncertainty to one of total certainty. This lends support to the hypothesis that marking menu users will use marks with practice.

The previous research on depth versus breadth in menus indicates two important points relative to marking menus. First, users need to explore to make selections from menus with which they are not familiar, and the semantics associated with the structure has an effect on human performance. Marking menus behave somewhat like traditional menu systems when used in the menu mode (i.e., users can see item names and navigate through the hierarchy). Therefore, we can assume that the research findings mentioned above are applicable in menu mode. Second, once familiar with the menu structure, users of traditional menu systems want to directly select an item. In other words, users no longer require a menu. This behavior bodes well with using a mark to select from a marking menu.

Since the previous research in this area is somewhat applicable to the menuing mode of marking menus, the open research issues concern using mark mode to access hierarchic marking menus. The main issue is the effect of breadth and depth on user performance when using marks. Specifically, how deep and how wide can menus be made before marking becomes too slow or error prone? What sort of structuring makes mark articulation easier? For example, selection using marks from a menu with 16 items seems difficult. Selection from a menu with two levels of four item menus (16 items in total) seems more reasonable. In Chapter 5, we examine the effect of breadth and depth on marking by means of an empirical experiment on human performance using marks to select items from hierarchic marking menus.

2.4.4. Command parameters and design rationale

Besides the angle of a mark specifying the command verb, other aspects of a mark can express command parameters. For example, a mark's starting point, ending point and size can all contribute to command semantics. The question is how can these aspects of a mark be exploited in an interface? Issues of this type are examined in a case study which involved implementing marking menus in a real application (Chapter 4).

Subtle differences in design may have a profound effect on the way in which marking menus can be used. For example, a design that uses selection upon sector entry (e.g., the Momenta command compass) must be used differently than a design that uses selection on pen release (e.g., marking menus). These small design details can have a large impact on a design's ability to support hierarchic menus, command/parameter pairing, and reselection. In section 2.5, we describe this design space and present a design rationale for marking menus.

2.4.5. Generalizing self-revelation, guidance and rehearsal

Marking menus provide self-revelation, guidance, and rehearsal for the particular class of mark. Specifically, this is the type of mark that is created as a byproduct in selecting from directional menus. We referred to this class of marks as "zig-zag" marks. A pen-based application may also use other types of marks (e.g., editing symbols). There are two issues concerning the relationship of marking menus and other types of marks. First, can marking menu marks be integrated with other types of marks? Second, can a mechanism be developed to provide self-revelation, guidance and rehearsal for other types of marks?

A major advantage of marks is the ability to use features of a mark as additional command parameters. For example, a copy mark not only specifies that a copy command should be executed but also specifies what should be copied and to where it should be copied. How self-revelation, guidance and rehearsal can be provided for this type of information is an open question. Chapter 6 addresses this question.

2.5. DESIGN RATIONALE

This section presents the design rationale behind marking menus. First, the fundamental goals and the space of the design are defined. Next, an explanation and taxonomy of design options is presented. Finally, the rationale for choosing a particular set of options for the design of marking menus is given.

2.5.1. Fundamental design goals

The fundamental design goals of marking menus are:

• in the mark mode, speed of selection is emphasized over the self-revealing features.

• in the menu mode, self-revelation and guidance are emphasized over speed of selection

• in menu mode movement must be as close as possible to a rehearsal of marking. Ultimately, using the menu must facilitate learning the marks.

The last goal dictates that marking must mimic selecting in menu mode. Furthermore, marks must be distinguishable from one another. This provides a further goal for the design:

• selection in menu mode must create a unique path which can be reliably recognized by a computer.

We next examine the types of designs that address these goals.

2.5.2. The design space

In the most general sense, the design space can be described as: "discriminating selections from menus by cursor movements". Linear menus, array menus, and radial menus all fall into this design space. Linear menus are menus where the items are laid out in sequential linear fashion (top to bottom, or left to right). Array menus are menus where the items are laid out in both a top to bottom and left to right fashion. Radial menus are menus where the items are laid out in a circle. In these types of menus, the position of the cursor ultimately determines the item selected. A design that does not fit in this class would be menu selection based on time. In this case, the computer cyclically displays each menu item and the user presses a button when the desired item appears. This type of menu selection is often used in interfaces for handicapped users.

In this space, selection is performed relative to a starting point and the amount and direction of movement determines the selection being made. For example, in a linear menu, when the cursor is initially placed on the first item in the list, selection is determined by how far the cursor is moved down the menu.

Within this design space we are only considering designs in which menu selection is a physical rehearsal of marking. We want each movement path traced by a menu selection to be unique relative to the other movement paths involved in selecting

from the menu. This will result in an unambiguous language of movements (or marks when the cursor leaves an ink-trail).

Within this design space we can identify several important design issues. These issues are discrimination, control, selection, display, backing up, and aborting.

2.5.3. Discrimination method

Discrimination method is defined as the type of movement used to discriminate selections. This can be either angle, length, or a combination of the two. Figure 1.8 shows a menu that uses length, and another menu that uses the combination of length and angle. Whether humans are better at discrimination by length or by angle is an open question.[9] In our context, discrimination by angle is preferable to discrimination by length for two reasons: efficiency, and scaling and rotation issues.

Under certain conditions, discrimination by angle (radial menus and angular marks)

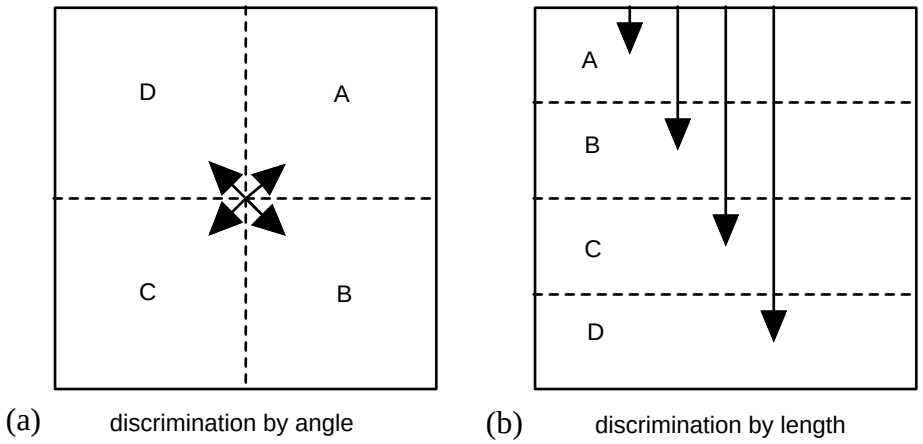


Figure 2.4: An example where discrimination by angle makes selection faster than discrimination by length. The lines with arrow heads show the movement needed to select an item. In the discrimination by angle case, selection of any item requires a movement of distance d. In the discrimination by length case, assuming all items are accessed with the same frequency and distance is equivalent to movement time, the average selection time will be 2L, where L is the height of a menu item. Assuming d is 0.5L, selection is four times faster with discrimination by angle.

allow faster selection than discrimination by length (linear menus and linear marks). First, because all the menu items are equidistant from the center of the menu in a radial menu, selection time is approximately the same for any item in the menu. In contrast, with linear menus, the first item can be selected more quickly than the last item in the menu. Figure 2.4 shows an example which compares a four-item radial menu and a four item linear menu. As described in Section 2.3.1, Callahan, Hopkins, Wieser, & Shneiderman (1988) have empirical evidence that eight-item radial menus are 15% faster and produce 42% fewer errors than eight-item linear menus. Treating selection from a radial menus as a one dimensional pointing task, and assuming that the amount of area used by a radial menu and a linear menu are the same, it can be shown that target size in a radial menu will always be larger than target size in a linear menu. For example, in Figure 2.4, the target size in the radial menu is the diagonal of an item. In contrast, target size in the linear menu is the height of an item. However, as the number of items increase in a radial menu, pointing to the narrow slices will become more difficult. To compensate for this, users will have to move farther away from the center, thus slowing their selection time. Determining the point where performance with a radial menu will degrade to the performance level of a linear menu is an open problem. Current research on two dimensional pointing (Mackenzie & Buxton, 1992) only deals with rectangular targets and therefore cannot be directly applied to radial menu slices.

There are also issues related to mark-based interfaces that make discrimination by angle preferable. Angular marks are preferred over linear marks because an angular mark can be scaled without changing its meaning (or, rather, changing the item the mark selects). In terms of a mark-based interface this means that a user is not restricted to draw the marks at a prescribed size. For example, a small "L" shaped mark would have the same meaning as a large "L" shaped mark. This is not the case with marks that are discriminated by length.

However, the meaning of angular marks changes if the mark is rotated. Rotating a horizontal to the right mark 45 degrees will cause it to be interpreted as a down to-

---

[9] It should be noted that discrimination can be performed at the reading or at the writing level (i.e., perception versus production of marks). These are significantly different problems. This dissertation examines production of angular marks. See Westheimer & McKee (1977) for a discussion of the perception of angle and length.

the-right mark by the system. In contrast, linear marks are not affected by rotation (i.e., a bull's eye menu. See Figure 1.8).

Discrimination by angle better reflects the way marks are interpreted in everyday life. Marks are generally insensitive to scaling but sensitive to rotation. For example, a small "l" has the same meaning as a large "l" but if it is rotated 90 degrees it perhaps takes on the meaning of "dash".

There is also the issue of *C:D ratio*. C:D ratio is defined as the ratio between the amount of movement of the input device (Control) and the amount of movement this imparts to the cursor (Display). On a pen-based system, the C:D ratio is constant and one to one because the cursor follows directly under the pen tip. For example, a one inch movement of the pen corresponds to a 1 inch mark. Therefore, with pen-based systems, C:D ratio is not an issue. However, with input devices that do not write directly on the display, (i.e., the mouse), C:D ratio is an issue. A one inch movement of the mouse may result in different lengths of marks on different computers if they have different C:D ratios. C:D ratios that vary depending on the speed of the movement (referred to as *cursor acceleration*) complicate this situation even further. A one inch movement made quickly can generate a much longer mark than the same movement made slowly, for example. Therefore, under these conditions, discrimination by length may be unreliable. However, discrimination by angle is not affected by varying C:D ratios. For example, a 45 degree mark is a 45 degree mark whether or not it is one or two inches long. Since it is desirable that our technique be usable with other input devices besides the pen, discrimination by angle is a better choice.

2.5.4. Control methods

Selection from a menu with a pointing device is generally accomplished by *dragging*, by *tapping*, or a combination of the two. We refer to these as the control methods. When dragging is the control method, pressing the pen down on the screen ("pen-down") displays the menu; moving the pen while it pressed against the screen ("dragging") selects different items; lifting the pen from the screen ("pen-up") confirms the selection. When menus are hierarchic, dragging into certain areas may cause submenus to be displayed for selection. When tapping is the control method, a pen-down followed quickly by a pen-up (a "tap") causes the menu to be

displayed; a “tap” over an item confirms its selection. If the menu is hierarchic, the selection will result in another menu being displayed.

Dragging is preferred because selection in menu mode must be a rehearsal of the movement needed to make the mark. Marks are created by dragging the pen across the display surface and therefore dragging is a more accurate rehearsal of marking than tapping.

Marking menus use an action called press-and-wait to allow the user to switch into menu mode. We elected to use this action for several reasons. First, it deviates very slightly from the act of marking (the wait is only 1/3 of second). Thus the principle of rehearsal is not dramatically violated. For example, an action such as holding down a special key or making a special movement to invoke the menu would be a much more dramatic violation of rehearsal. Second, when a user wants to avoid menu mode, it usually means one wants to articulate the command quickly. Press-and-wait is easily avoided by quick articulation and avoiding it also makes selection faster. Third, according to our design goals, we assume that novices are not concerned with fast selection and therefore a slight delay in selection is a minor inconvenience. However, as users become more experienced with the menus and desires faster selection, the delay may also provide incentive to use marks.

There are other reasons why delaying the pop-up of the menu is valuable: it can be distracting; it can obliterate part of the screen; and it takes time. For a novice user these may not be problem since displaying the menu is desirable. For expert users, however, a delayed menu pop-up allows the creation of marks and avoids the negative side effects of the menu's display.

2.5.5. Selection events: preview, confirm and terminate

There are several events that occur when making a selection. Selection from a menu generally involves some sort of feedback indicating which item is about to be selected, for example, an item highlights. We refer to this capability as *selection preview*. Selection also involves an action which indicates to the system that it should actually carry out the selection. We refer to this as *selection confirmation*.

In the non-hierarchic case, selection confirmation results in the termination of the entire selection process. In the hierarchic case, selection confirmation will not necessarily terminate the selection process if the item selected has a sub-menu. We

use the term *selection termination* to indicate the action that ends the entire menu selection process. In non-hierarchic case, selection confirmation and selection termination are combined in the same action.

There are many different types of input events that could be used to signal selection confirmation:

• *Pen-up/pen-down*

• *Item entry:* Item entry means selection confirmation occurs the moment the pen enters an item.

• *Boundary crossing:* Boundary crossing means that selection confirmation occurs when the pen crosses the outside border of a menu item.

• *Dwelling:* Dwelling is the act of keeping the pen pressed and not moving for a fraction of a second. A user can avoid issuing dwelling events by keeping the pen moving. Press-and-wait is an example of a dwelling event. However, we distinguish between these two events because press-and-wait signals the entry into menu mode while dwelling signals selection confirmation.

• *Events distinct from pen movement:* This includes things like a button press or an increase in pressure with a pressure sensing pen.

The type of selection confirmation event used affects other design features:

• *mimicking drawing a mark:* Since selection from a hierarchy of menu items involves a series of selection confirmations and we wish to mimic that act of making a mark, an event for selection confirmation that does not interrupt dragging must be used.

• *reselection:* In some cases, a user may desire to change the previewed selection. For example, a user may accidentally move into the wrong item then want to move to the correct item. We refer to this process as reselection. Most menu systems support reselection.

• *pairing command and parameters:* The command compass allows dragging to continue after the final selection confirmation. Dragging is then used to indicate additional parameters to the menu command just selected.

Figure 2.5 shows which selection confirmation methods support these features. Item entry is not feasible because it does not allow reselection. Boundary crossing, dwelling and events distinct from pen movements support both reselection and pairing. We discount "events distinct from pen movement" because it requires additional input sensors like pen buttons or a pressure sensing pen.

Figure 2.5 indicates that boundary crossing and dwelling are the only applicable choices. Boundary crossing is preferable because a visible boundary (i.e., the edge of a menu) gives precise information as to when selection will occur. This information is not visible if dwelling is used. Furthermore, waiting for a dwelling to occur slows interaction. It is also possible to use pen release as a confirmation method if pairing is not required and the item being selected is the last in a series of selections.

We implemented boundary crossing by having selection confirmation occur when the user crossed over the outer edge of a menu item. Specifically, selection previewing occurred as long as the user stayed within the circle of the menu. Selection confirmation occurred when the user moved outside the circle. We discovered, in practice, that boundary crossing created a problem. As a user moves away from the center of the menu to confirm an item, the item's sub-menu pops up when the outer boundary is crossed. Unless a user moves very slowly, one is still moving when the sub-menu appears. This results in one of the items in the sub-menu being selected immediately. If the user is moving fast, the boundary point for the sub-menu may have already been crossed and this results in an erroneous selection confirmation. Even if the boundary point was not crossed, this overshooting in the sub-menu causes reselection to be the first action to occur each time a sub-menu is popped up. This means that users are not rehearsing the movement of drawing a mark, but are rather making a movement which involves reselection. This approach was therefore unacceptable.

To solve this problem, we used a hybrid approach which combines boundary crossing and dwelling. The approach works as follows. As long as the pointer is within some distance from the center of menu, a dwelling event is ignored. Selection preview and reselection are therefore possible without the threat of an accidental dwelling occurring. Once the boundary is crossed, selection preview and reselection are still possible but, if the user dwells, the selected item is confirmed and its sub-menu appears. This allowed users to use coarser movements to make selections without fear of overshooting and selecting from sub-menus.

Dwelling is also consistent with press-and-wait. In both these activities, keeping the pen pressed against the display and holding it triggers the display of a menu.

A selection can also be confirmed without dwelling by releasing the pen at any point in the hierarchy of a menu. This allows any item in the hierarchy to be selected and also signals selection termination.

2.5.6. Mark ambiguities

The current design presents a dilemma if we consider using marks to make selections from hierarchies of menus. The idea behind using marks for selection is

Selection confirmation event	**allows mimicking marking?**	**allows reselection?**	**allows pairing?**
pen release	no*	yes	no
item entry	yes	no	yes
boundary crossing	yes	yes	yes
dwelling	yes	yes	yes
events distinct from pen movement	yes	yes	yes

(* yes in the non-hierarchic case)

(as long as the pointer is kept moving)

Figure 2.5: Different selection confirmation methods characteristics.

that selection will be fast and fluid. This implies that we do not desire or expect a user to "include" dwellings when making selections using marks. This would be unnatural and slow the marking process.

A problem can occur if dwellings are not included when making marks. Consider a selection from a hierarchy that is two levels deep. Suppose the user makes a straight line mark. Does the mark correspond to a selection from the parent menu or the child menu? Figure 2.6 shows the problem. If dwellings no longer occur we cannot disambiguate the selection. If we base the interpretation on boundary crossing, then the mark is unambiguous. Unfortunately, this makes the size of a mark affect its interpretation (i.e., the marks cannot be scaled).

One solution to this problem is called *no category selections.* It is based on the observation that items which have subitems are generally categories of commands, not commands themselves, and selecting a category is not a meaningful operation. For example, when using linear hierarchic menus on the Macintosh, selecting the "font" category leads to a menu of commands that change the font. Selecting "font" by itself (i.e., releasing the mouse button when "font" is selected) performs no operation. Therefore we assume that there is no need to select a category. Thus, we can consider any straight line to be a selection into a submenu (case (b) in Figure 2.6). Note that this permits selection of certain menu items that are embedded in submenus by drawing a short straight mark. We recommend designers put the most popular item in a category in this position to promote efficiency.

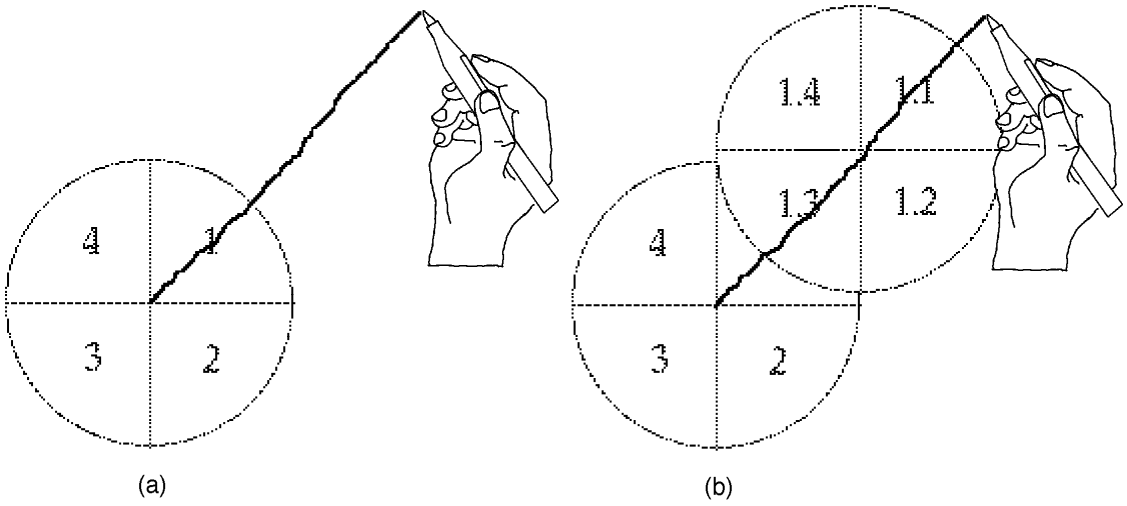


Figure 2.6: Ambiguity in selecting from a hierarchy of menu items two levels deep using a mark. Overlaid grayed menu show possible interpretations. In (a), the interpretation is the selection of item 1. However, (b) is another interpretation according to boundary crossing rules (the selection of item 1.1). Interpretation by boundary crossing is sensitive to the size of marks.

No category selections breaks down when the depth of the hierarchy is greater than two. Suppose a user makes a "^" mark as shown in Figure 2.7 (a). The start of the mark and the change in direction within the mark indicate two points of menu selection. However, what indicates selection from the third level of menu? Figure 2.7 shows this problem. Once again, boundary crossing can be applied to derive an unambiguous set of menu selections but this results in unscalable marks.

There are several solutions to this problem which preserve scaling. The first solution, referred to as the *no-oping* (from the phrase "no operation"), is to simply not permit a series of menu selections that result in a straight line. One way of doing this involves making the item in the child menu that "lines up" with the selection angle of the parent a null operation. This ensures that the beginning of a selection of a non-null item from a child menu is indicated by a change in angle. Unfortunately, this "wastes" a useful sector in a menu.

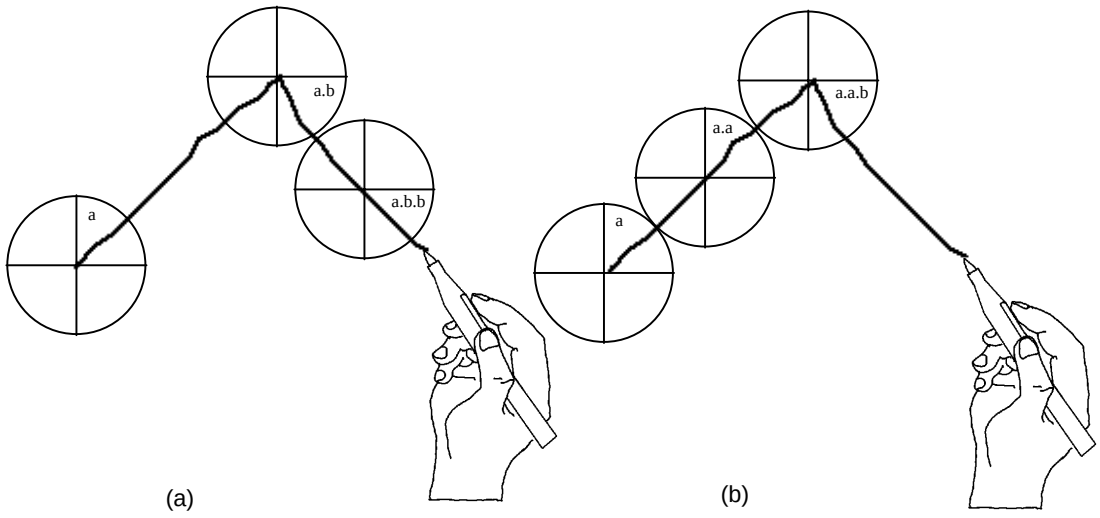


Figure 2.7: Possible interpretations of mark when selecting from hierarchies greater that two levels deep. The straight line sections of the mark have no artifacts to indicate whether the selection at that point is being made from the parent or from the child.

A second solution is *axis-shifting*. This involves rotating child menus such that no item appears at the same angle as an item in the parent menu. Figure 2.8 shows an example of this technique. Axis-shifting involves aligning the boundary between two items in the child menu with the selection angle of the parent item. This ensures that the beginning of a selection from child menu is indicated by a change in angle. Axis-shifting avoids the wasted sectors that occur with no-oping.

This discussion has presented four solutions to hierarchic menu design which are intended to produce an unambiguous vocabulary of marks. The four solutions are: boundary crossing, no category selections, no-oping, and axis-shifting. The aspects of the design that are affected by these solutions are: the ability to select any item within the hierarchy, the ability to have mark interpretation independent of the size of a mark, the ability to select leaf items with a single straight line, and the ability to have all items in a menu active. These aspects may also vary relative to the depth of the menu. Figure 2.9 summarizes this design space.

A solution can be chosen based on the demands of the menu. If menus are only one or two levels deep and menu categories do not need to be selected, then no category selections will work. Boundary crossing and axis-shifting are suitable when hierarchies are more than two levels deep and category menu items need to be

selected. Boundary crossing is also an acceptable solution if category items need to be selected and mark scaling is not an issue.

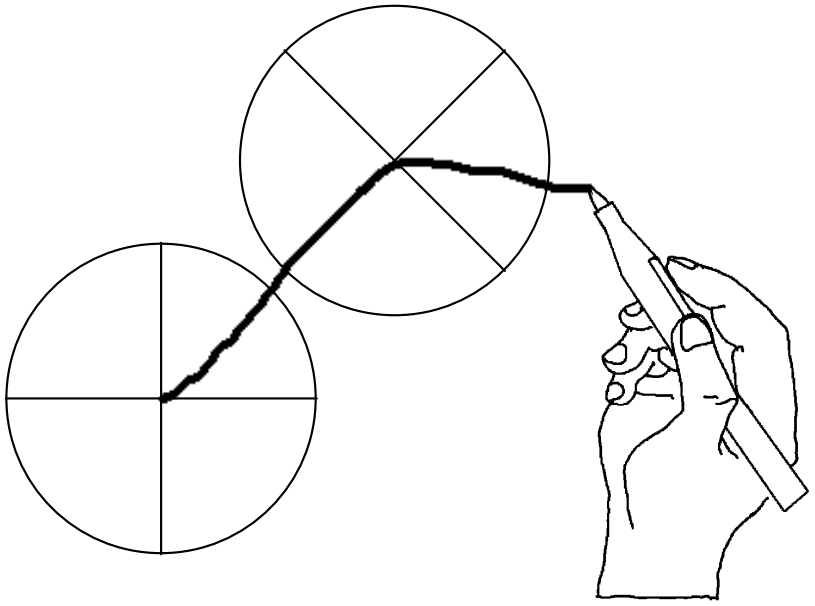


Figure 2.8: Axis shifting rotates a child menu such that child menu items do not appear on the same angle as the parent menu item. This results in a mark language where selection confirmations are indicated by changes in angle. With this scheme marks can be drawn at any size.

Policy	no depth limit?	select any item?	marks scalable?	allows "straight lining"	all items active?
boundary crossing	Yes	Yes	No	Yes	Yes
no-oping	Yes	Yes	Yes	No	No
no category selections	No (2)	No (except in 1 deep case)	Yes	Yes	Yes
axis-shifting	Yes	Yes	Yes	No	Yes

Figure 2.9: Policies that avoid ambiguous interpretation of marking menu marks.

2.5.7. Display methods

There are several design options which concern how menus are displayed:

• *Menu trail* refers to leaving parent menus displayed as a user descends a hierarchy of menu items.

• *Menu overlap* refers to displaying child menus over the top of parent menus.

These methods become important when backing up in a hierarchy of menus.

2.5.8. Backing-up the hierarchy

The ability to back-up in a hierarchy of menus is useful for browsing menu items and correcting mistakes. Backing-up can be one of three types: back-up only to the parent menu, back-up to any ancestor menu, back-up to any ancestor menu item. Backing-up can be accomplished in several ways. Pointing to an item can trigger a back-up to the item, or an explicit action can trigger a back-up (i.e., tapping the pen triggers a back-up to the parent menu). A combination of these two methods can be used (i.e., tapping on an item to back-up to it). Lifting the pen is already used to indicate selection termination, so the back-up technique is restricted to pointing while the pen is being dragged.

Backing-up brings the roles of menu trail and menu overlap into play. Pointing to the item in order to back-up to it requires that item be displayed on the screen. Therefore a menu trail must be provided. However, child menu items may cover up parent items making it impossible to point to "covered" items. The design must ensure that parent items are not covered up.

Design requirements dictate that backing-up in marking menus operates like backing-up in traditional drag-through hierarchical menus: to back-up to a parent menu item, a user points to it; the system then closes the currently displayed child menu and displays the child menu of the parent item. We can adopt this scheme for marking menus but it reduces the advantage of radial menu selection. Figure 2.10 shows the problem that occurs. A selection from a child menu may result in pointing to a parent menu item and this causes an unintended back-up. A prototype implementation of marking menus revealed this to be a real problem. The problem could be avoided if a user is "careful", but this tends to slow users down.

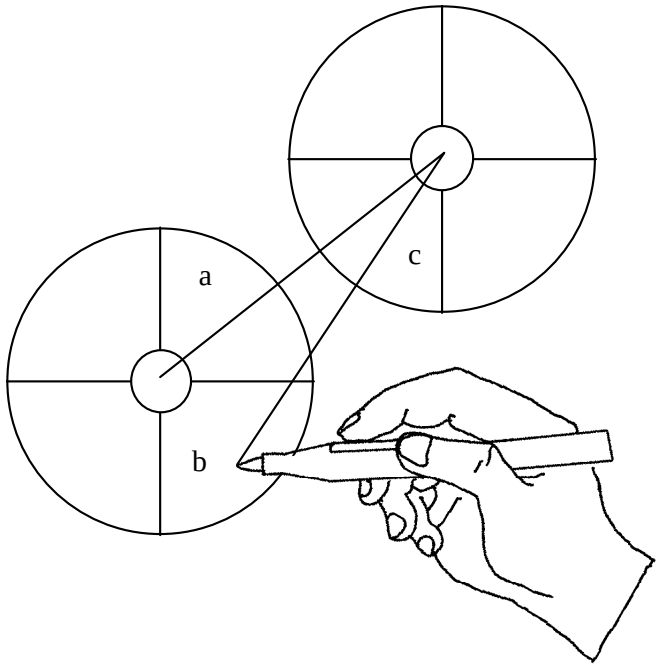


Figure 2.10: A problem with the backing up by pointing to a parent item. Is the user selecting item a.c or backing up to item b?

To solve this problem, we could restrict marking menus to operate like linear menus where selection occurs only if the user is pointing inside a menu item. This has two

major disadvantages. First, it selection sensitive to the length of strokes, and second, it massively reduces item size from a sector of the entire screen to the small sector of the menu.

The solution is to reduce the size of the back-up targets. This is done by restricting the back-up targets to the center hole of the parent menus. This drastically reduces the probability of accidentally pointing to a back-up target. Furthermore, we constrain the user to dwell on a center before back-up takes place. This allows the user to "pass through" centers without backup occurring. Figure 2.11 shows this back-up scheme.

This approach has the restriction of only allowing back-up to parent menus. Backing up to a parent menu and displaying another one of the child menus cannot be combined in the same operation. Some hierarchic linear menus allow this. However, this restriction permits fast and unconstrained selection when moving forward in the hierarchy, while still allowing back-up.

This back-up scheme has several more advantages. First, one can back-up to any parent menu, grandparent menu, etc. Second, menu overlap can occur just as long as menu centers do not get covered. Finally, because backing-up actually returns the cursor to parent menus, rather than redisplaying parent menu at the cursor location, this reduces the chances of menus "walking off" the screen (this problem is further discussed in Section 6.2.3).

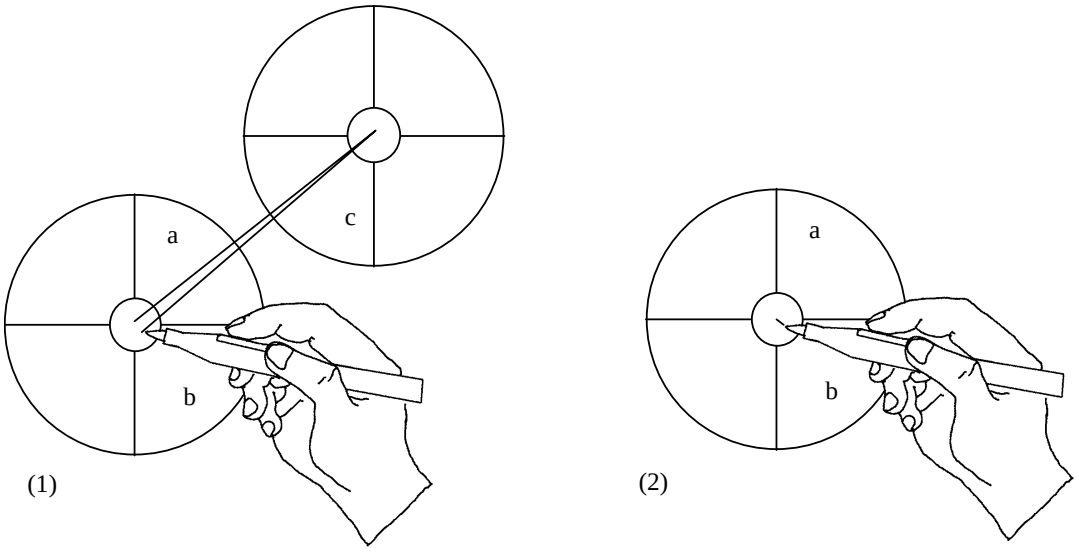


Figure 2.11: Backing-up in hierarchic marking menus. In (1) the user moves into the center of a parent menu and dwells momentarily. In (2) the system senses the dwelling and backs-up to the parent menu by removing the child of item a. Selection may then continue from the parent.

2.5.9. Aborting selection

Most menu systems have a way of specifying a null selection. Generally this is accomplished by selecting outside a menu item. As explained previously, marking menus allow selection to occur outside the item to make selection easier. To circumvent this problem, the center hole of a menu is used to indicate no selection. Lifting the pen within the center hole results in the menu selection being aborted.

A mark may be also be aborted. This involves either lifting the pen before the mark is complete or turning the mark into an uninterpretable scrawl while drawing it.

2.5.10. Graphic designs and layout

During everyday use of marking menus we observed some problems with a "pie" graphical representation. First, as the number of items in the menu increases and the length of labels increases, the size of the pie grows rapidly. This creates several problems. First, having large areas of the screen display and undisplay is visually annoying. Second, a large menu occludes too much of the screen. In many situations, a menu associated with a graphical object must be popped up over the

object. The problem is that displaying the menu completely hides the object. This results in the context of the selection being lost. Third, large menus take time to display and undisplay. In most systems, the image "underneath " a menu is saved before a menu is displayed, and restored when a menu is undisplayed. When a menu is very large, these operations take considerable amounts of time because large sections of memory are being copied to and from the display. Also, algorithms for sizing and laying out labels within the pie of the menu can be quite complex. This makes the implementation of menu layout procedures complex. Complex computations may also delay the display of menus.

To solve these problems we designed an alternate graphic layout for marking menus called "label"[10]. Figure 2.12 shows an example. This alternate design has several advantages over a pie representation. First, it reduces the amount of screen that changes when a marking menu is displayed and undisplayed, and therefore, it reduces visual annoyance. Second, it occludes less of the screen than a pie representation because only the menu center and labels are opaque. Thus more of the context underneath a menu can be seen. This design also reduces the amount of memory that must be copied to and from the display, and hence it reduces the amount of time needed to display a menu.

Another issue of graphical layout is the problem of displaying menus near an edge or corner of the screen. Pie menu systems deal with this issue by using a technique called "cursor warping". Unfortunately, cursor warping is not suitable for pen-based systems. In Chapter 6, we further discuss this issue and describe an alternative to cursor warping.

Although not shown in Figure 2.12, marking menus have many standard features found in traditional menus. For example, marking menus allow grayed-out and checked items. Also, if an item has a submenu, a small circle appears to the right of the label. The intention is that this circle represents the center hole of the submenu. We also found it valuable to hide the labels of parent menus, thus reducing screen clutter. The only portion of a parent menu that is displayed is the center hole (so a user can point to it to back-up). We have also experimented with transparent menus

---

[10] We acknowledge Mark Tapia for his assistance in designing and implementing the alternate graphical layout for marking menus

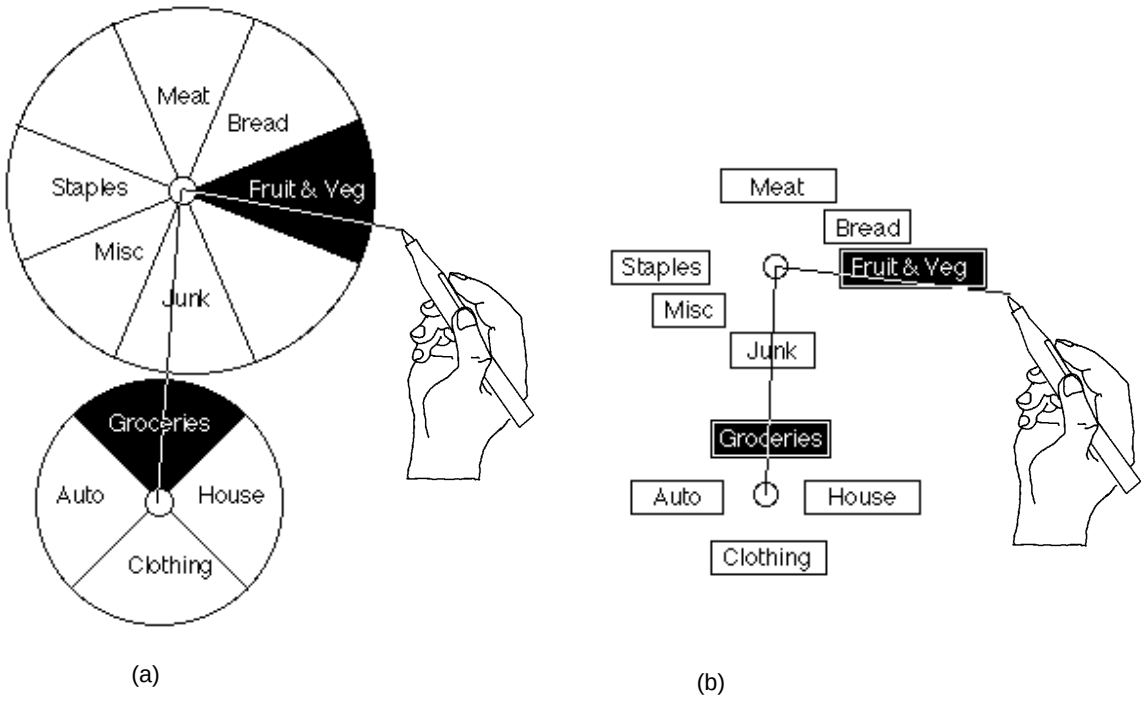


Figure 2.12: An alternate graphic representation for a radial menu "label". Rather than displaying "pie" shapes (a), only the labels and center are displayed (b). The menu then occludes less of display and can be displayed faster.

and graying out parent menus but a full discussion of these experiments is beyond the scope of this dissertation.

2.5.11. Summary of design

The previous sections described and discussed various design features and options of marking menus. We now summarize the features and indicate which design options we elected to use.

Marking menus use discrimination by angle. Selection previewing in menu mode is supported by dragging the pen into an item, and the item being highlighted. Selection confirmation is indicated by a combination of boundary crossing and dwelling. Selection termination is indicated by pen up.

To avoid mark ambiguities, we recommend three possible strategies: no-oping, no category selections and axis-shifting. If menus require only a few items, no-oping may be a suitable solution. If menus are only two levels deep and category selection is not required, no category selection is a suitable solution. If menus require many

menu items, and are more than two levels deep, axis-shifting must be used. In practice, we used no category selection in many situations.

Making a selection in menu mode leaves a menu trail but only the center of parent menu is displayed. We found in practice this reduces the visual clutter the would be caused by the display of inactive parent menu items. Menus are allowed to overlap, but because only the center of parent menu is displayed, this generally does not cause visual confusion.

In menu mode, selection can be aborted by terminating the selection while pointing to the center hole of a menu. In mark mode, selection can be aborted by turning the mark into a "scribble".

If a user dwells while drawing a mark, the system indicates the menu items that would be selected by the mark by displaying the menus "along" the mark. The system then goes into menu mode. This process, called mark confirmation, can be used to verify the items that are about to be selected by a mark or a portion of a mark.

Marking menus can be displayed in either a "pie" representation or a "label" representation. A "label" representation is suitable when there is a need to minimize the amount of screen occluded by the display of the menu.

2.6. SUMMARY

The success of an interaction technique depends not only on its acceptance by users but also on its acceptance by interface designers and implementors. An "industrial strength" interaction technique must not only be effective for a user, but also have the ability to co-exist with other interaction techniques, other paradigms, and differing features of the software and hardware. Because of these demands, as in many other interaction techniques, our motivation and design behind marking menus is complex. What appears on the surface as a simple interaction technique is actually based on many different motivations and has many design subtleties and details.

In this chapter we defined marking menus and described the various motivations for developing and evaluating them. These included providing marks for functions which have no intuitive mark, supporting unfolding interface paradigms,

simplifying mark recognition, maintaining compatibility with existing interfaces, and supporting both novice and expert users. We are also motivated to study marking menus as a way to evaluate the design principles they are based on.

We then outlined the issues involved in evaluating marking menus and proposed an initial design. The major parameters to be evaluated concern the question of how much functionality can be loaded on a marking menu. Essentially our research focus is on establishing the limitations of marking menus so interface designers who are utilizing marking menus can design accordingly. The remaining chapters explore the limitations and characteristics of the design.

Chapter 3: An empirical evaluation of non-hierarchic marking menus

This chapter addresses basic questions about marking menu design variables: how many items can marking menus contain; what kinds of input devices can be used in conjunction with marking menus; how quickly can users learn the associations between items and marks; how much is performance degraded by not using the menu; and whether there is any advantage in using an ink-trail. This chapter describes an experiment which addresses these questions. The approach is to pose specific hypotheses about the relationship between important design variables and performance, and then to test these hypotheses in the context of a controlled experiment. The results of the experiment are then interpreted to provide answers to the basic questions posed above.

In this experiment we limit our investigation to non-hierarchic marking menus. We do this for several reasons. First, this experiment serves as a feasibility test of non-hierarchic marking menus. If non-hierarchic marking menus prove feasible, then an investigation of hierarchic marking menus is warranted. Second, we feel that the characteristics of non-hierarchic marking menus must be understood before we can begin to investigate hierarchic marking menus. Our findings on non-hierarchic marking menus can then be used to refine our design and evaluation of hierarchic marking menus. Third, this experiment addresses many factors. To include the additional factor of hierarchic structuring would make the experiment too large and impractical.

To date there is little research applicable to our investigation. Callahan, Hopkins, Weiser, and Shneiderman (1988) investigated target seek time and error rates for 8-item pie menus, but concentrated on comparing them to linear menus. In particular

they were interested in what kind of information is best represented in pie menu format. Section 2.3.1 described their results.

Our experiment focuses on selecting from marking menus using marks. To address the questions posed at the start of this chapter, the experiment examines the effect that the number of items in a menu, choice of input device, amount of practice, and presence or absence of an ink-trail or menu, has on response time and error rate.

3.1. THE EXPERIMENT

3.1.1. Design

In this experiment, we varied the number of items per menu and input device for three groups of subjects and asked them to select target items as quickly as possible from a series of simple pie menus. One group selected target items from fully visible or "exposed" menus (Exposed group). Since there is little cognitive load involved in finding the target item from menus which are always present, we felt that this group would reveal differences in articulation performance due to input device and number of items in a menu.

Two other groups selected items from menus which were not visible ("hidden" menus). In one group, the cursor left an ink-trail during selection (Marking group), and in the other, it did not (Hidden group). The two hidden menu groups were intended to uncover cognitive aspects of performance. Hiding the menus would require the added cognitive load of either remembering the location of the target item by remembering or mentally constructing the menu, or by remembering the association between marks and the commands they invoke through repeated practice. Comparing use of an ink-trail with no ink-trail was intended to reveal the extent to which supporting the metaphor of marking and providing additional visual feedback affects performance. The Exposed group provided a baseline to measure the amount that performance degraded when selecting from hidden menus.

3.1.2. Hypotheses

We formed the following specific hypotheses to address the questions posed at the start of this chapter:

How much is performance degraded by not using the menu?

Hypothesis 1. Exposed menus will yield faster response times and lower error rates than the two hidden menu groups. However, performance for the two hidden groups will be similar to the Exposed group when the number of items per menu is small. When the number of items is large, there will be greater differences in performance for hidden versus exposed menus. This prediction is based on the assumption that the association between marks and items is acquired quickly when there are very few items. As the number of menu items increases, the association between marks and items takes longer to acquire, and mentally reconstructing menus in order to infer the correct mark becomes more difficult.

How many items can marking menus contain?

Hypothesis 2. For exposed menus, response time and number of errors will monotonically increase as the number of items per menu increases. This is because we assume that performance on exposed menus is mainly limited by the ease of articulation of menu selection, as opposed to ease of remembering or inferring the menu layout. We know that performance time and errors monotonically increase as target size decreases, all else being equal (Fitts, 1954).

Hypothesis 3. For hidden menus (Marking and Hidden groups), response time will not solely be a function of number of items per menu. Instead, menu layouts that are easily inferred or that are familiar will tend to facilitate the cognitive processes involved. We predict that menus containing eight items can be more easily mentally represented than those containing seven items, for example. Similarly, a menu containing twelve items is familiar since it is similar to a clock face, and thus we predict it is more easily mentally represented than a menu containing eleven items.

What kinds of input devices can be used in conjunction with marking menus?

Hypothesis 4. The stylus will outperform the mouse both in terms of response time and errors. The mouse will outperform the trackball. This prediction is based on previous work (Mackenzie, Sellen, & Buxton, 1991) comparing these devices in a Fitts' law task (i.e., a task involving fast, repeated movement between two targets in one dimension).

Hypothesis 5. Device differences will not interact with hidden or exposed menus, or the presence or absence of marks. Differences in performance due to device will

not depend on whether the menus are hidden or exposed, or whether or not marks are used. The rationale for this is that we assume performance differences stemming from different devices are mostly a function of articulation rather than cognition. We also assume that the articulatory requirements of the task are relatively constant across groups.

Is there any advantage in using an ink-trail?

Hypothesis 6. Users will make straighter strokes in the Marking group. We based this prediction on the assumption that visual feedback is provided in the Marking group and also that hidden menus support the "marking" metaphor as opposed to the "menu selection" metaphor.

How quickly can users learn the associations between items and marks?

Hypothesis 7. Performance on hidden menus (Marking and Hidden groups) will improve steadily across trials. Performance with exposed menus will remain fairly constant across trials. This prediction is based on belief that articulation of selection (or simply executing the response) will not dramatically increase with practice since it is a very simple action. Performance on hidden menus, however, involves the additional cognitive process of recalling the location of menu items. We believe this process will be subject to more dramatic learning effects over time.

3.1.3. Method

Subjects. Thirty-six right-handed subjects were randomly assigned to one of three groups (Exposed, Hidden, and Marking groups). All but one had considerable experience using a mouse. Only one subject had experience using a trackball. None of the subjects had experience with a stylus.

Equipment. The task was performed on a Macintosh IIX computer. The standard Macintosh mouse was used and set to the smallest C:D ratio. The trackball used was a Kensington *TurboMouse*, also set to the smallest C:D ratio. The stylus was a Wacom tablet and pressure-sensitive stylus (an absolute device). The C:D ratio used was approximately one-to-one.

Task. Subjects used each of three input devices to select target "slices" from a series of pie menus as quickly and as accurately as possible. The pies contained either 4, 5, 7, 8, 11, or 12 slices. All pie menus contained numbered segments, always beginning

with a "1" immediately adjacent and to the right of the top segment. The other slices were labeled in clockwise order with the maximum number at the top (see Figure 3.1 (a)). The diameter of all pie menus was 6.5 cm., and Geneva 14 point bold font was used to label the slices.

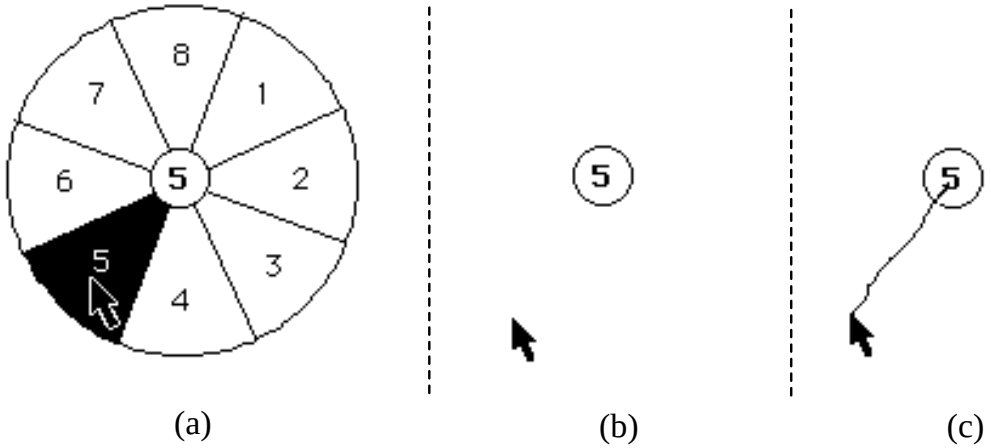


Figure 3.1: Selecting item 5 from an eight-item pie menu (a) in the Exposed group, (b) in the Hidden group, and (c) in the Marking group.

In designing this experiment, a great deal of time was spent discussing what kind of items should be displayed in the pie menus. Menus in real computer applications usually contain meaningful items, but the order in which they appear is not easily inferred. The numbered menus we used, on the other hand, used ordered, meaningless labels. We wanted to approximate the case of an expert user who is familiar with the menu layout. We decided to reduce as much as possible the learning time associated with memorizing the items. Our focus was on the articulation of actions, and the cognitive processes involved in mentally representing or mentally constructing menu layout. Since Callahan et al. (1988) have shown that performance varies depending on the kinds of items represented, using the same kind of items for all menus (numbered items) was an attempt to eliminate this effect. Thus our comparisons between menus with different numbers of items would be more accurate. We acknowledge that both the choice of menu items and their mapping within a menu may have a significant effect on performance. These factors are outside the scope of this investigation.

In the Exposed menu group, the entire menu was presented on each trial (Figure 3.1 (a)). The target number corresponding to the slice to be selected was presented when the subject located the cursor within the center circle of the pie menu and either pressed down and held the mouse or trackball button, or pressed down and maintained pressure on the stylus. The subject's task was then to maintain pressure and move in the direction of the target slice. Menu slices would highlight as the cursor moved over them, indicating to the subject a potential selection. A slice would remain highlighted even if the cursor went outside the outer perimeter of the pie. Releasing the button, or pressure, signaled to the system that the highlighted slice was selected. After the selection was made, the menu would "gray out" displaying the menu with the slice selected for a period of 1 second. If an incorrect slice was selected, the Macintosh would beep on release. This marked the end of a trial.

In the Hidden menu group, the task was essentially the same, except that during selection, only the central circle of the pie menu would be visible (Figure 3.1 (b)). After confirming the selection, subjects would receive the same grayed-out feedback as in the Exposed group, indicating which response had been made, and whether or not it had been correct. The Marking group was almost identical to the Hidden group, except that the movement of the cursor with the button depressed left an ink-trail (Figure 3.1 (c)) .

After each trial, subjects received a running score, presented in the lower right-hand corner of the screen. A minimum of 10 points could be obtained for each correct response, with more points scored as response time became shorter. However, subjects were penalized 20 points for errors.[11] At the end of each block of trials, each subject's current performance was shown in relation to the best score obtained by other subjects in the same conditions. The scoring criterion was the same for all groups.

Design and Procedure. One third (twelve) of the subjects were randomly assigned to the Exposed group, one third to the Hidden group, and one third to the Marking

---

[11] This scoring scheme was arrived at by experimenting with different scoring schemes on pilot subjects. We found that the choosen scheme emphasized both accuracy and speed. On average, subject scores were positive and they found this encouraging and fair.

group. Every subject used each of the three input devices (mouse, trackball and stylus). Trials were blocked by device and order of device was counterbalanced.

For each device, *all groups* began by practicing on exposed menus for a total of six trials for each of six different menus, containing either 4, 5, 7, 8, 11 or 12 items. During practice, number of items per menu was blocked and presented in random order. This practice period was intended to acquaint subjects with the feel of the particular input device they were about to use. It also provided an opportunity for subjects to familiarize themselves with the layout of the menus before beginning the timed trials.

Subjects in the Exposed group then moved on to the timed trials, while subjects in the Hidden and Marking groups received a further set of practice trials designed to acquaint them with the "feel" of hidden menus. For this practice session, menus containing both three and six items were used (six trials each) since 3-item or 6-item menus were never used in the actual timed trials. This was a deliberate attempt to equalize exposure to the menus of interest in the three groups.

For the timed portion of the experiment, trials were again blocked by number of items (4, 5, 7, 8, 11, or 12). The order in which the number of items appeared was randomly permuted for each subject. Each subject began a particular block by first studying the menu layout for 6 seconds. They then received a total of 40 trials for each different menu with a short break at intervals of ten trials. Targets were drawn randomly from a uniform distribution with replacement, with the added constraint that no target could be repeated on consecutive trials.

In summary, each subject performed 40 trials on each of the six menus (menus consisting of 4, 5, 7, 8, 11, and 12 items) and using all three devices, resulting in a total of 720 scores per subject. Each group consisted of twelve subjects which resulted in 8640 scores per group. The three different groups provided a total of 25920 scores for the experiment.

3.2. RESULTS AND DISCUSSION

The main dependent variables of interest were response time and number of errors. Response time was defined as the total time from presentation of the target number

to confirmation of the selection for error-free trials. An error was defined as an incorrect selection. The means for each group are shown in Figure 3.2.

3.2.1. Effects due to number of items per menu

As expected, increasing the number of items per menu significantly increased both response time ($F(5,55) = 388.4$, $p < .001$) and errors ($F(5,55) = 382.8$, $p < .001$).[12] There were overall performance differences among the groups in terms of errors ($F(2,22) = 21.97$, $p < .001$) but not in terms of response time. However, these main effects are not particularly meaningful because differences among groups depend on the number of items per menu (see Figure 3.3). That is, there was a significant interaction between group and number of items per menu both in terms of response time ($F(10,110) = 3.5$, $p < .001$) and errors ($F(10,110) = 64.7$, $p < .001$).

These results address the first three hypotheses:

(1) As predicted by Hypothesis 1, mean response time was consistently lower in the Exposed group versus the Hidden and Marking groups as the number of items increased. This is supported by the significant interaction between group and number of items per menu (reported above), and by specific comparison tests. No difference was found between the two hidden groups and the Exposed group for menus containing four items. However, for menus containing five items, response times were significantly slower for hidden menus compared to Exposed ($F(1,110) = 6.5$, $p < .001$). The two hidden groups were no different from each other in terms of errors (post hoc comparison of error means, Tukey HSD, $\alpha = .05$), but both produced significantly more errors than the Exposed menu group.

(2) Our second hypothesis predicted that in the Exposed group, response time and errors would monotonically increase as a function of number of items per menu. In the case of errors, this relationship seems to hold. However, this must be qualified by the fact that errors were infrequent and thus floor effects may obscure the true shape of the function.

---

[12] Throughout this disseration we use the F-statistic to evaluate the equality of population means. See Appendix A for an explanation.

Group	Mean RT in sec. (SD)	Mean Number of Errors in 40 Trials (SD)	Mean Percentage Errors
Exposed	*0.98 (0.23)*	*0.64 (1.00)*	*1.6%*
Hidden	*1.10 (0.31)*	*3.27 (3.57)*	*8.2%*
Marking	*1.10 (0.31)*	*3.76 (3.67)*	*9.4%*

Figure 3.2: Mean response time and number of errors for each experimental group.

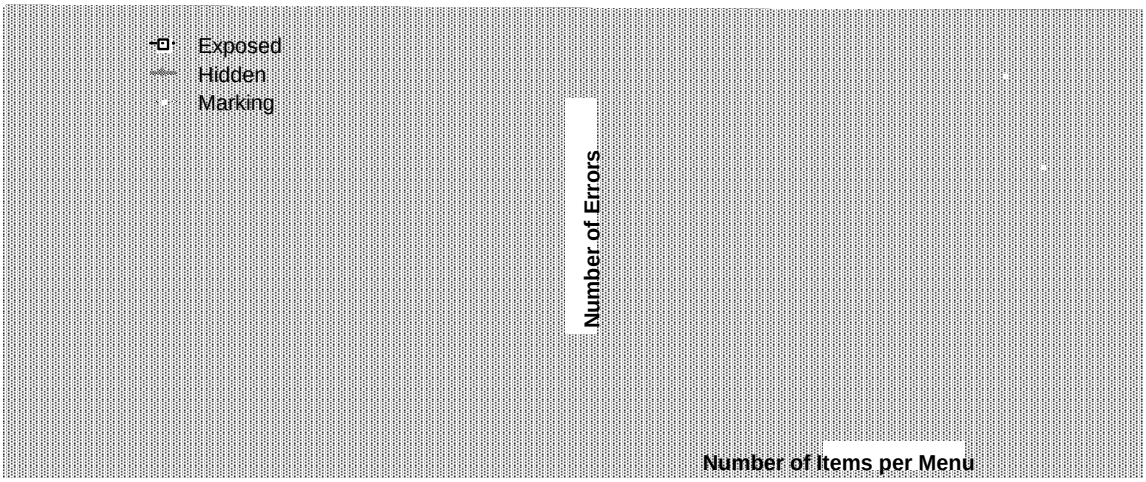


Figure 3.3: Response time and average number of errors (of a total of 40 trials) as a function of number of items per menu and group.

Response time also increased monotonically except for menus containing twelve items. Specific comparisons at the .05 level confirm significant increases in response time from four to five items per menu ($F(1,55) = 16.8$, $p < .001$), and from seven to eight items per menu ($F(1,55) = 7.4$, $p < .01$), but no differences between eleven and twelve items per menu. One possibility is that familiarity with the "clock face" layout may have reduced the time for visual search, thereby reducing overall response time. Another possibility is that this could be a case of diminishing effects. Adding an extra item to a menu containing four items represents a 20% increase in

the number of items, whereas, adding an extra menu item to one which contains eleven represents only an 8% increase in number of items.

(3) The pattern of results predicted by Hypothesis 3 is also supported: when menus were hidden, some kinds of menus were easier to evoke or reconstruct from memory than others. This was not purely a function of number of items per menu. The characteristic curve that emerges (Figure 3.3) shows that performance in general does tend to degrade as the number of items per menu increases, but that certain numbers of items do not follow this pattern (i.e., eight and twelve items).

This hypothesis is also confirmed by a series of specific comparisons showing no differences in either hidden menu group for seven versus eight items per menu. Further, performance on menus of twelve items was faster than on menus of eleven items for the Hidden group ($F(1,55) = 11.25$, $p < .001$) and was more accurate than on menus of eleven items in both groups (Hidden, $F(1,55) = 50.96$, $p < .001$; Marking $F(1,55) = 13.51$, $p < .001$). By contrast, for both groups, tests show menus of four items yielded faster response times than menus of five items (Hidden, $F(1,55) = 4.05$, $p < .05$; Marking $F(1,55) = 9.00$, $p < .05$).

The results show that menus containing twelve items in particular may have facilitated performance. Many subjects mentioned that the metaphor of a clock face helped them to select the target item because it could be brought readily to mind. Thus it seems reasonable to suggest that it is the cognitive bottleneck, or the difficulty of evoking the mapping between target and action, that limits performance.

3.2.2. Device effects

As predicted by Hypothesis 4, subjects performed better with a stylus and a mouse than they did with a trackball. Response time ($F(2,22) = 9.64$, $p < .001$) and errors ($F(2,22) = 11.29$, $p < .001$) were both affected by the type of input device subjects used. Pairwise comparisons (Tukey HSD test, $\alpha = .05$) showed the trackball was both significantly slower and gave rise to more errors than the stylus or mouse. However, contrary to our expectations, there was no difference in mean response time or errors between the stylus and mouse.

Initial analyses supported Hypothesis 5 where we predicted that the effect of input device would not depend on whether or not the menus were exposed, or whether or

not there was an ink-trail. Input device did not interact with group, either in terms of response time or errors.[13] However, on closer examination, a more interesting result emerged.

We discovered that in the Marking group, the stylus was significantly faster than both the trackball and mouse with no difference between the trackball and mouse (Figure 3.4). In the Exposed group, the mouse and stylus were faster than the trackball, with no difference between the mouse and stylus. These discoveries were based on separate analyses of variance for each of the three groups on the response time data. There were significant differences among devices in the Exposed ($F(2,22) = 10.44$, $p < .001$) and Marking groups ($F(2,22) = 8.32$, $p < .002$), but not in the Hidden group. Tukey tests revealed the superiority of the stylus in the Marking group and the inferiority of the trackball in the Exposed group. No significant interactions between device and number of items were found in any of the three groups. Given these results we cautiously reject Hypothesis 5.

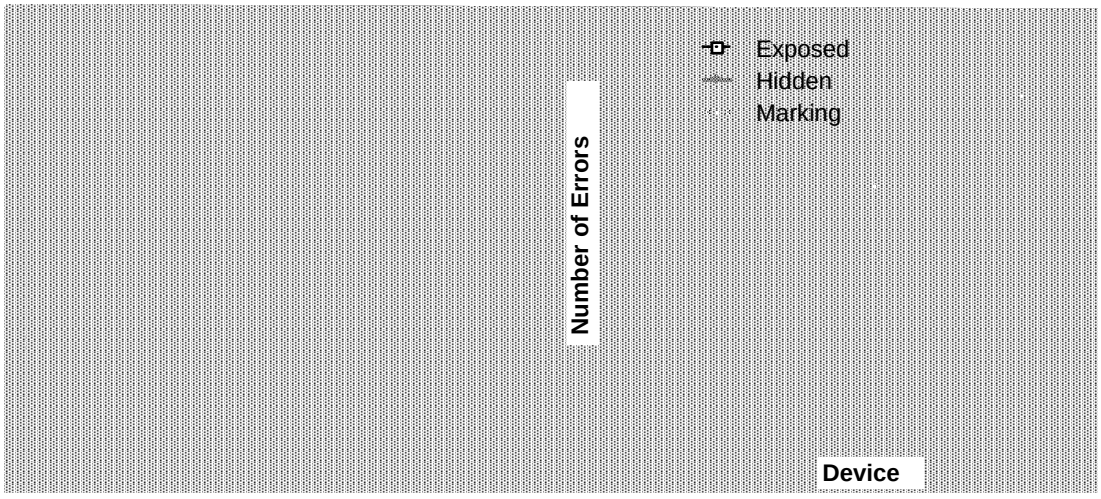


Figure 3.4: Response time and average number of errors (of a total of 40 trials) as a function of device and group.

There may be two reasons for the superiority of the stylus when marks are added to selection from hidden menus. First, it is often difficult to perceive when enough

[13] There were also no significant interactions between number of items per menu and device, nor significant three-way interactions (group by number of items by device).

pressure is being applied to the stylus to make a selection. Thus, providing visual feedback when this state is maintained may be important to realize the full potential of this device. Second, providing an ink-trail is consistent with the metaphor of marking with a pen, which may improve performance. Alternatively, failing to support the pen metaphor by not providing the ink trial (Hidden group) may violate users' expectations and thus negatively affect performance.

Separate analyses of the error data within each group further supported the inferiority of the trackball. The trackball was found to be the source of significant device differences in the Exposed ($F(2,22) = 9.92$, $p < .001$)[14] and Marking groups ($F(2,22) = 9.92$, $p < .001$). Pairwise comparisons in the Exposed and Marking groups showed differences between the trackball and the other two devices, and no difference between mouse and stylus.

The finding that the trackball was no more slower or error prone than the mouse and stylus in the Hidden group may be due to the fact that in both the Exposed and Marking groups, visual feedback emphasized the difficulty of articulating the actions of the trackball thereby causing performance to be worse. In the Exposed case, sectors were highlighted as they were selected and it is possible that the trackball caused a great deal of reselection. In the Marking case, users reported that the ink-trail was disturbing in conjunction with the trackball because the paths looked erratic and inaccurate.

3.2.3. Mark analysis

We were interested in seeing if subjects used straight marks when making selections. This was important to discover because, if menu selection tended to be done in some manner other than a straight mark, we could not claim that users rehearse this physical movement when selecting from menus. Thus we would not expect as much transfer of skill between making menu selections and making marks. Another reason we were interested in seeing if subjects used straight marks was related to using marking menus in applications that recognize other marks beside those used in menu selection. Unlike conventional menu selection which is based

[14] Both a significant device by menu size interaction ($F(10,110) = 2.47$, $p < .011$) and floor effects should make us cautious in interpreting the main effect of device in the Exposed group. However, the fact that the trackball produces consistently more errors on average across menu size, supports the claim that the trackball is outperformed by stylus and mouse.

only on the last location of the cursor, mark recognition systems take the entire shape of the stroke into account. For example, suppose the system also recognizes the symbol "C". A very crooked mark intended to make a selection from a hidden menu might be interpreted as an "C". The success of recognition depends to some extent on knowing the shapes of the strokes that users tend to create. To address these issues we recorded and displayed the path data for users' individual marks. Figure 3.5 shows a typical example.

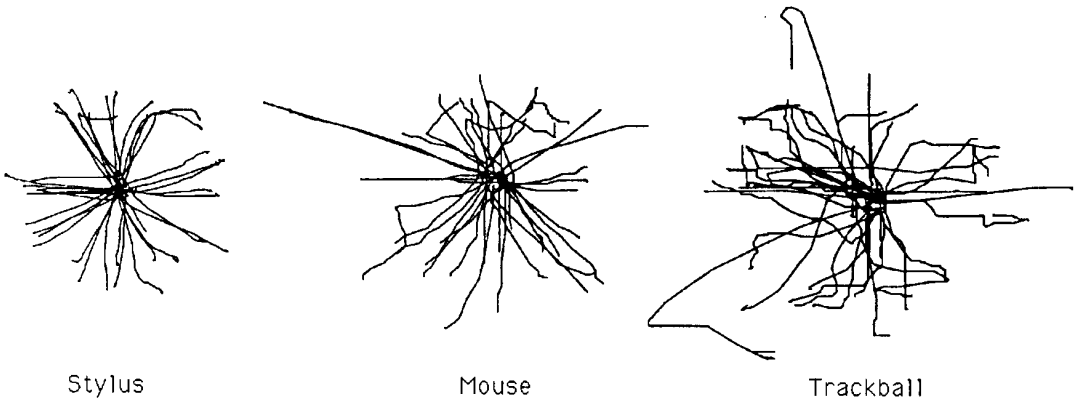


Figure 3.5: The marks a subject used in selecting from a hidden twelve-item menu.

Subjects made approximately straight marks. No alternate strategies such as starting at the top item and then moving to the correct item were observed. However, there was evidence of reselection from time to time, where subjects would begin a straight mark and then change direction in order to select something different.

Surprisingly, we observed reselection even in the Hidden and Marking groups. This was especially unexpected in the Marking group since we felt the idea of drawing a mark does not naturally suggest the possibility of reselection. Hence, we reject Hypothesis 6. It was clear though, that training the subjects in the hidden groups on exposed menus first made this option apparent. Clearly many of the subjects in the Marking group were not thinking of the task as making marks *per se*, but of making selections from menus that they had to imagine. This brings into question our *a priori* assumption that the Marking group was using a marking metaphor, while the Hidden group was using a menu selection metaphor. It may explain why very few behavioral differences were found between the two groups.

Reselection in the hidden groups most likely occurred when subjects began a selection in error but detected and corrected the error before confirming the selection. This was even observed in the "easy" four-slice menu, which supports the assumption that many of these reselections are due to detected mental slips as opposed to problems in articulation. There was also evidence of "fine tuning" in the hidden cases, where subjects first moved directly to an approximate area of the screen, and then appeared to adjust between two adjacent sectors.

Strokes produced with the trackball appeared more jagged and less controlled than those made with the mouse or stylus. This is consistent with the statistical results showing that the trackball tends to be slower and less accurate than the stylus or mouse. For four-item menus, most subjects made straighter marks with the stylus than the mouse. The presence or absence of an ink-trail did not appear to make any discernible difference to stroke shape.

3.2.4. Learning effects

The forty trials for each different menu were divided into eight consecutive blocks. Response time and mean errors were calculated for each five-trial block in order to look more closely at learning effects. Overall, there was a small but steady decrease in response time over trials which was statistically significant ($F(7,77) = 5.79$, $p < .001$). Error rate also showed signs of improving with number of trials ($F(7,77) = 10.52$, $p < .001$).

We have claimed that the major factor limiting performance on exposed menus is the physical accuracy required for the action of selection. The results support this claim. In the case of hidden menus, results support the claim that the factor limiting performance is cognitive. In other words, the time it takes to remember or infer the correct mental representation becomes the overriding factor determining performance. Thus, performance in the Exposed group can serve as a baseline measure that users should approach as they become expert.

Hypothesis 7 states that the cognitive component is the component most affected by learning, as opposed to the articulatory component. Thus, we expect a steady improvement in performance in the two hidden groups, as opposed to fairly constant performance in the Exposed group over time.

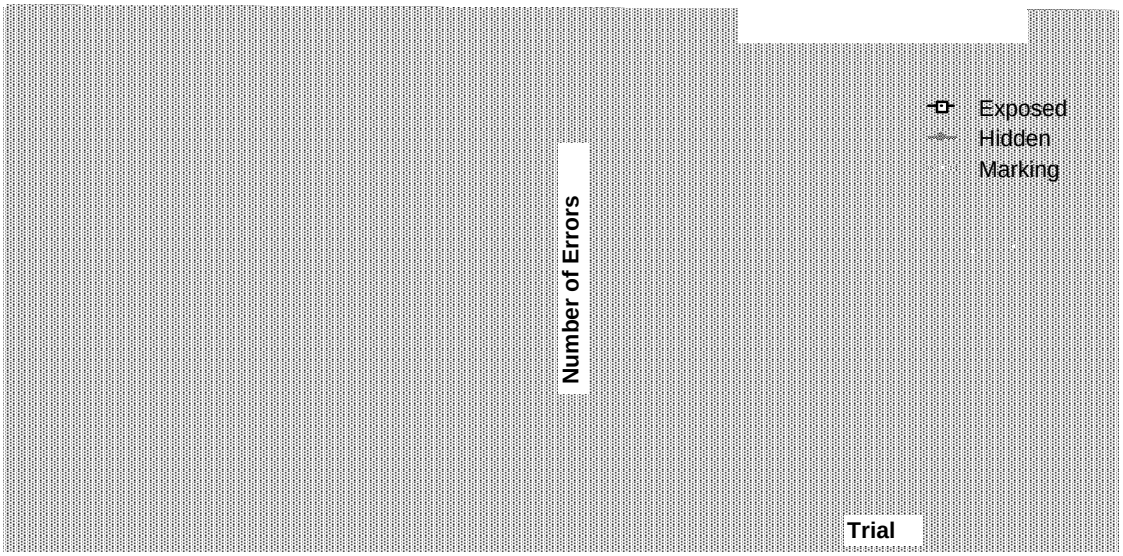


Figure 3.6: Group effects in terms of response time and number of errors in five trial intervals.

As is shown in Figure 3.6, response time in the hidden groups appears to improve across trials while the curve for the Exposed group is fairly flat. Errors also remain relatively constant for the Exposed group over trials, while decreasing on average for the two hidden groups. Support for Hypothesis 7 is found in a significant group by trial interaction for response time ($F(14,154) = 2.90$, $p < .001$) and errors ($F(14,154) = 3.15$, $p < .001$).

As a final point, it follows from the above reasoning that we would expect no significant interaction of input device by trial, since type of input device would presumably have the greatest impact on the articulation as opposed to the cognitive component of performance. The fact that no significant interaction of device by trial was found is consistent with expectation.

3.3. CONCLUSIONS

Relative to our seven hypotheses, the results and their implications for design can be summarized as follows:

Hypothesis 1. As predicted, when menus have many items, hiding menus from users both slows their performance and increases their error rate. As number of items per menu increases, added to the problems of articulation is the difficulty of successfully mentally reconstructing the menu layout or remembering the necessary

strokes to make menu selections. However, when the number of items is small (only four), there is little or no performance difference, even early in practice.

Design Implications. For ordered sets of commands, users should be as fast and error-free in making marks as in selecting from a visible pie menu of up to four slices. If the commands are not ordered, then it may take more time to acquire the skill. However, command semantics can be exploited. For example, "Open" and "Close" can be positioned opposite to each other, as can "Cut" and Paste". This may speed the learning process and allow users to mark ahead faster. In addition, the most frequently used commands form a very small set, and thus we can be optimistic that these can be invoked successfully with marking menus.

Hypothesis 2 and 3. For exposed menus, the results showed performance declines steadily as the number of items increases. This is probably due to two factors: (1) the increasing reaction time to visually search and choose among alternatives, and (2), the increasing difficulty of articulating the action as targets become smaller.

These results agree with other results concerning the effect of the number of items on performance. Perlman conducted an experiment in which subjects made selections from exposed linear menus (Perlman, 1984). Menus containing 5, 10, 15 and 20 items were used. The menus contained ordered numbers from 1 to 20. Beside each item was a randomly chosen left or right arrow character. The task was to find a target item in the menu and indicate it by pressing the corresponding left or right arrow-key. It was found that the number of items in a menu had a linear effect on the time it takes to find an item. These results agree with our results for exposed menus.

Performance on hidden menus in this experiment was different, however. Instead of a result showing monotonically increasing response times and error rates as a function of number of items, even numbers of items (four, eight, or twelve) appeared to facilitate performance. Not surprisingly perhaps, four-item menus yielded significantly faster and more accurate performance than five-item menus. However, performance on eight-item menus was no worse than performance on menus with one less item. Subjects also reported that the eight-item menu was easy to learn because they could easily mentally subdivide the pie and infer the position of the target slice. Most dramatic was the finding that a twelve-item menu actually yielded faster and more accurate performance than a menu containing only eleven

items. We speculate that this difference may be enhanced by familiarity with circles subdivided into twelve sectors, such as in clock faces.

Design Implications. When menus are hidden, overcoming the difficulty of learning and using mental representations of menus can be facilitated by using layouts which exploit known metaphors, or which are easily subdivided. Using an even number of items or laying out items at the points of a compass or hour positions of clock can be used to counteract the increased difficulty of having many items in a menu. The ease with which subjects learned and performed with the twelve-item menu is testimony to the strength of a good metaphor. One could imagine a user remembering a command location or mark by mapping it to an hour/hand position: "undo is at three o'clock".

Hypothesis 4 and 5. The stylus and mouse outperformed the trackball both in terms of response time and errors. Analysis of the paths showed that paths made with the trackball were more jagged and less controlled than those made with the mouse or stylus. The stylus and mouse yielded similar performance, with the exception that the stylus was significantly faster than the mouse when an ink-trail was present.

Design Implications. The results speak strongly against using a trackball for marking menus. Further, subjects' comments suggest that the combination of trackball and ink trial was especially bad. One subject complained of being disturbed by the messy ink-trail left when using a trackball. It seems that the visual feedback provided by the ink-trail only served to emphasize the inadequacy of the paths made by this device.

The performance similarity of the mouse and stylus suggests that either may be appropriate devices for this kind of mechanism. Two cautionary notes should be made, however. First, it is likely that the ink-trail added important feedback to tell the user when the appropriate amount of pressure was being applied to the stylus. This suggests that another kind of stylus (i.e. one with audio or tactile feedback to indicate a "button-click") might have fared better against the mouse in all groups. It also reveals a design deficiency of the stylus that could easily be overcome. Second, while the mouse and stylus yielded similar performance, observation of people using the mouse to make marks other than straight strokes suggests that the mouse may be inferior to the stylus in other situations.

Hypothesis 6. Subjects made essentially straight strokes. However, there was evidence of reselection (where subjects would begin a straight stroke and then change stroke direction in order to select something different) even in the hidden groups. This casts doubt on our initial assumption that subjects in the Marking group would begin to think of the task as making marks, instead of making menu selections. Instead, it suggests that they thought of the task in terms of making selections from the exposed menus they were trained on, which now happened to be hidden. Marks themselves do not afford reselection, whereas pie menus do.

The fact that the marking metaphor was not supported as strongly as we hypothesized may account for the fact that no major differences were found between the Hidden and Marking groups. For example, the presence or absence of an ink-trail did not appear to make any discernible difference to stroke shape.

Design Implications. Since users tended to make straight strokes we are optimistic that users are rehearsing the physical movement required to make marks as they perform menu selection. This bodes well for learning. There was some evidence of non-straight strokes which appeared to be reselection in the Marking group but it was not overwhelming. Perhaps in the context of a mark recognition system a user will learn that reselection results in a mark that cannot be recognized and that reselection is not possible when using a mark.

Hypothesis 7. Performance across trials was uniform for exposed menus but underwent steady and significant improvement across trials for hidden menus (both groups). We argue that the performance limiting factor for exposed menus is the difficulty of articulating selection actions, whereas in the hidden groups the limiting factor is the time it takes to evoke or construct the correct mental representation. Articulation skills were acquired fairly rapidly and reached stable performance. Thus performance in the Exposed group provides a baseline measure that users of hidden menus approach.

Design Implications. The substantial improvement for hidden menus over only 40 trials suggests that if the menus contain meaningful and frequently used commands, users will acquire the necessary skills quickly and easily. Both response time and error rates can be expected to rapidly improve with time. The question of *how much* practice is necessary for hidden menu performance to equal exposed menu performance, and how that varies with number of items per menu is an issue for

further research and analysis. Meanwhile, we can be confident that small numbers of items will enable users to quickly begin marking ahead.

3.4. SUMMARY

This chapter investigated basic questions concerning design variables of marking menus: how many items can marking menus contain; what kinds of input devices can be used in conjunction with marking menus; how quickly do users learn the associations between items and marks; how much is performance degraded by not using the menu; is there any advantage in using an ink-trail. An experiment addressed these questions by varying the number of items per menu and input device for three groups of subjects, and asking them to select target items as quickly as possible from a series of simple pie menus. One group selected from menus that were visible at all times, another group selected from menus that were hidden, and the final group selected from menus that were also hidden, but had the additional visual feedback of a cursor ink-trail. The differences in group conditions were intended to separate articulation and cognitive aspects. The experiment compared selection times and error rates. In addition, learning effects were analyzed.

The results of the experiment indicate that non-hierarchic marking menus, or specifically the action of using a mark to select from a menu, is a useful idea. Our results indicate that: (1) four, eight and twelve items menus are suitable for marks; (2) if that number of items is kept low (e.g., four, eight and twelve), users will be able to use marks very early in practice; (3) higher numbers of items are possible but require more practice; (4) for non-hierarchic menus, users will perform as well with the mouse as they would with the stylus/tablet. Using a trackball, however, will be slower and more error-prone than using a mouse or stylus/tablet.

In terms of using marking menus in an application, the results indicate that a designer should attempt to use four, eight or twelve item menus. For example, if seven commands are to be placed in a menu, the designer should use an eight-item menu and leave one item blank or duplicate one of the more popular commands in the extra item. Although this experiment did not address this issue, it may also be also be advantageous to maintain consistent subdivisions for menu items. For example, use four and eight item menus (items on 45 angles) but not twelve item menus (items on 30 angles).

The results are encouraging because there are many applications where menus which have a small number of items could be effective. For example, *Microsoft Word* has seven groups of function icons that appear in the "ribbon" and "ruler" display area. These icons could be grouped into seven marking menus containing four or less items. Each group of icons could be replaced by a single icon which when pressed displays a four-item marking menu. The elimination of icons would allow space to display more text, or other or larger function icons (larger icons make pointing to them easier). The graphics editor in Microsoft Word already has tool pallet icons that work this way but uses pop-up linear menus. The popular Macintosh drawing program called *Canvas* also uses a similar scheme. Many of the menus that pop up from tool pallets icons in Canvas have twelve or fewer items.

While there are many situations where menus with twelve items or less may be sufficient, there are also many situations where menus contain more that twelve items. For example, font menus, large color pallets and paragraph style menus commonly contain more than twelve items. Chapter 5 shows that hierarchic marking menus make it possible to use a mark to select from a large number of items.

Given the results of this experiment, we can now apply them to the design of hierarchic marking menus. We recommend that hierarchic marks contain only menus with even numbers of items and the number of items be less than twelve. Because the poor performance of the trackball in this experiment, it would not be suitable for hierarchic marking menus. Also it would be worthwhile to see if the mouse performs as well as the stylus on "zig-zag" marks. Chapter 5 applies these design recommendations and evaluates hierarchic marking menus.

Despite the value of such controlled studies, there are a number of questions which can only be answered by careful design and implementation of marking menus in real applications. How long will it take for users to start using marks? How intensely will users use marks? What are the issues involved in integrating such a mechanism into a larger, more complex interface? Chapter 4 addresses these types of questions by means of a case study of user behavior using a marking menu for a real task.

Chapter 4: A case study of marking menus

The previous chapter has developed an empirical understanding of non-hierarchic marking menus. From this understanding, guidelines for designing marking menus and interfaces that use marking were generated. In this chapter we report on a study which applies those guidelines to the design of marking menus in an application and we evaluate user behavior while operating this application. The application was designed to solve a real world task and was used in accomplishing real work for a project not related to this thesis. The intention was to gain insight on integrating marking menus with other interface components and to find out how well marking menus perform in everyday practical work situations.

4.1. DESCRIPTION OF THE TEST APPLICATION

A conversation analysis/editor program, named *ConEd*, developed at University of Toronto, was used as a test application for marking menus (Sellen, 1992). By digitizing audio from a conversation among four people, data were collected concerning who is speaking and when. The conversation analysis/editor program is then used to display this data in a "piano roll" like representation. The program runs on a Macintosh computer. Figure 4.1 shows a typical display of the data window. The y-axis represents the four participants in the conversation, and the x-axis represents time. A black rectangle indicates that a particular person is speaking for a duration of time (this is referred to as an event). The window can be scrolled to reveal different moments in the conversation. Besides displaying the data, the application can be synchronized to a video recording of the conversation. As the video plays, the application moves a horizontal bar across the window to indicate

the current location in the conversation. If the bar moves past the right side of the display, the application automatically scrolls to the next section of conversation.

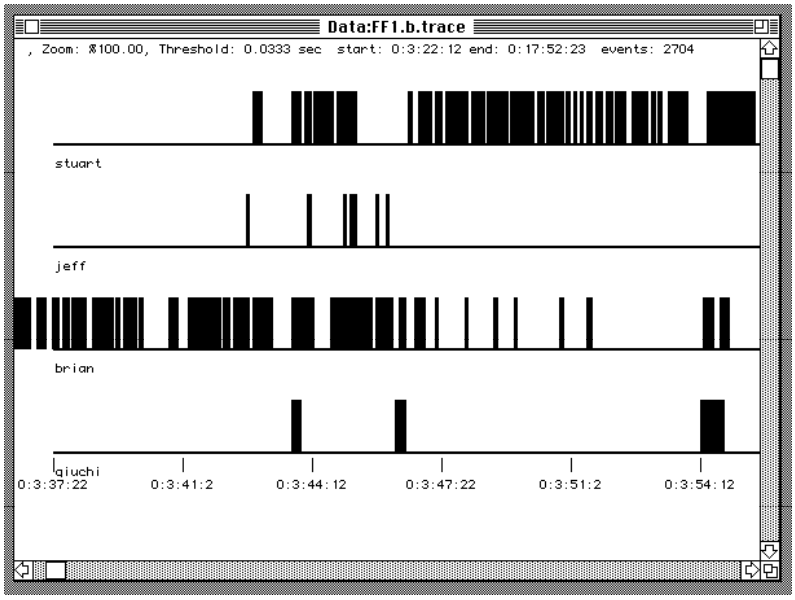


Figure 4.1: The "piano roll" representation of speaker versus time in ConEd.

Data can be edited as well as viewed with this application. Such things as coughs and extraneous noises need to be deleted. Other pieces of conversation, such as laughter, must be tagged for later analysis. Very often events must be added or extended because the automated speaker tracking system was not accurate enough.

Typically, a user sits in front of the Macintosh and video monitor, watching the video and editing events in real-time. Most of the time, a user operated the video transport with the left hand and the mouse with the right hand.

A marking menu triggers the six most frequently used commands, which consisted of commands that coded and edited the blocks of speech. The amount of coding and editing required was extremely high. Over 18 hours of operation, the two users performed 5,237 selections.

4.2. HOW MARKING MENUS WERE USED

4.2.1. The design

Figure 4.2 shows the marking menu used in ConEd. This menu can be popped up by pressing-and-waiting with the mouse in the "piano roll" window. Alternatively, a mark can be made to select the command. A user can issue six commands using this menu: laugh, delete, add, fill-in, ignore, and extend.

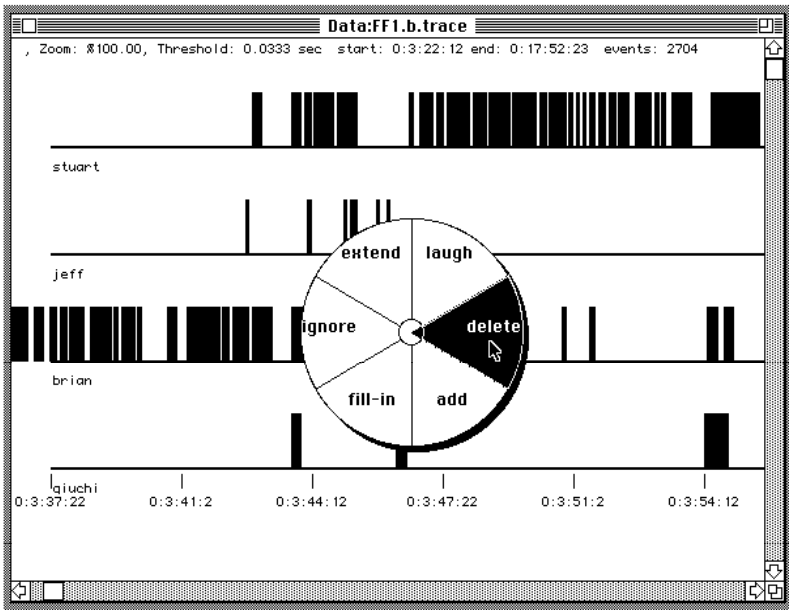


Figure 4.2: The six most frequently used editing commands are placed in a marking menu in ConEd.

Delete: The "delete" command deletes events. If the starting point of the delete selection/mark is made over an event, then that event is deleted. If the starting point is not over an event, then the events lying between the starting and ending points of the selection/mark are deleted. See Figure 4.3.

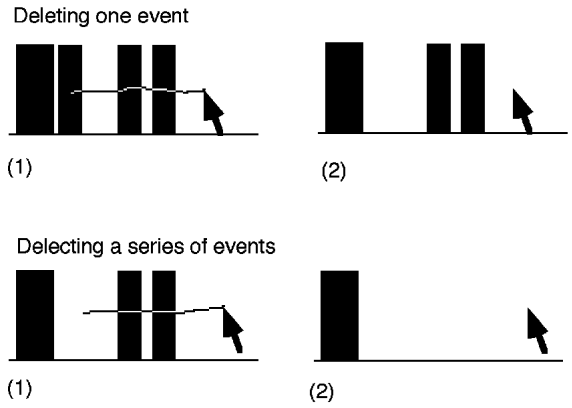


Figure 4.3: Events can be deleted one at time, by pointing to the event, or in a series by drawing over a series events.

Add: "Add" allows new events to be added. The starting point of the add selection/mark defines the beginning of a new event. The starting point of the following add selection/mark defines the end point of the new event and causes it to be displayed. If add is performed over an existing event, it is disregarded. See Figure 4.4.

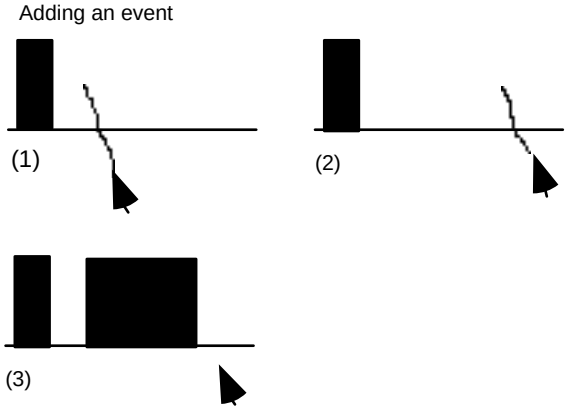


Figure 4.4: Events are added by specifying a starting point followed by and endpoint.

Extend: "Extend" elongates an event. The starting point of the extend selection/mark defines the length of the elongation. Either the start or the end of an event can be extended. If the selection/mark is made between two events, the event whose starting or ending point is closest to the starting point of the selection/mark is elongated. If extend is started over an event, it is ignored. See Figure 4.5.

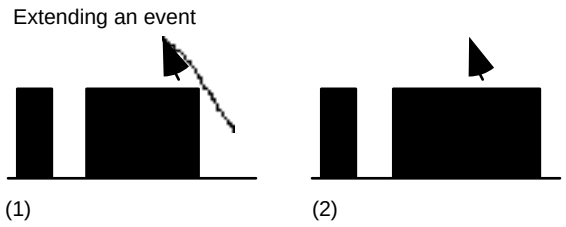


Figure 4.5: Events can be extended by pointing to the location of the extension.

Fill-in: "Fill-in" allows a gap of silence between two events to be filled. The two events are replaced by one long event. The starting point of the selection/mark indicates the gap to be filled. If Fill-in is ignored if started over an event. See Figure 4.6.

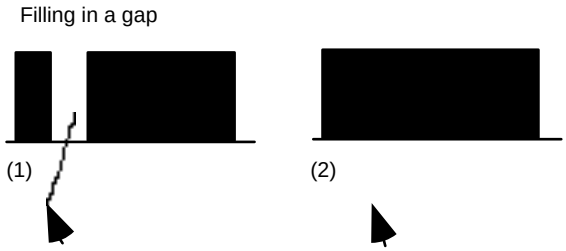


Figure 4.6: Gaps between events can be filled in by pointing to the gaps.

Ignore and Laugh: "Ignore" and "Laugh" allow events to be coded as special types. For example, speaking events generated by laughter must be tagged so they can be excluded from analysis of the conversation. Back-channel events (i.e., someone saying "uh huh" or "yes" but not trying to interrupt while another person is talking) must also be tagged. The starting point of the ignore or laugh selection/mark defines the event being coded. Either command is disregarded if not started over an event. See Figure 4.7 and 4.8.

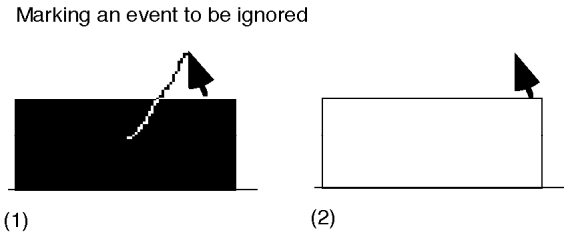


Figure 4.7: An event can be marked to be ignored by pointing to it.

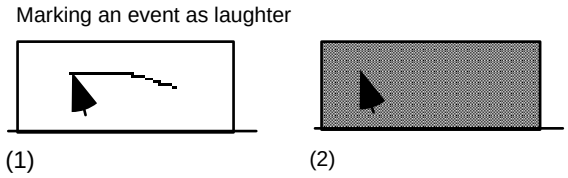


Figure 4.8: An event can be marked as laughter by pointing to it.

4.2.2. Discussion of design

Menu item choice

ConEd has more commands than the six contained in the marking menu. There are several reasons for placing this particular set of commands in a marking menu. First, the experiment in Chapter 3 showed that even numbers of items, up to twelve, enhance marking performance. Hence, six is within this range. Second, a requirements analysis told us that these six commands are the most frequently used. This implied several things. First, it would be advantageous if these commands could be invoked quickly. Therefore, marks would be suitable for these commands since marks can be issued very quickly. Second, these commands would be good candidates for marking menus because using the commands frequently would help a user memorize the associations between marks and commands. This, in turn, would lead to users using the marks.

Spatial aspects

Use of end points: While the marks used in marking menus are very simple, other features of a mark besides its angle can be used. The starting and ending points of a mark are obvious candidates. Features of a mark have been used in a similar manner by previous researchers (Coleman, 1969; Rhyne, 1987).

A requirements analysis revealed that the most frequent operations would involve selecting an event and applying an operation to it. Thus, marking menus were used in an object oriented manner—the starting point of the selection/mark indicates the object of the command. Note that this is not always the case. For example, the extend command does not point to an event to be extended but to the location of the extension. The particular event to be extended is inferred by the system. However, we found that this inconsistency caused no problems for the user.

The combination of pointing and marking produces the feeling of directness one gets when pointing and moving in objects in direct manipulation interfaces. When using marks in ConEd, there is no sensation of explicitly making a selection before applying an operation.

Use of horizontal/vertical dimension: Spatial commonalties between the representation being edited and the direction of menu items can be used to determine the assignment of directions to commands. For example, horizontal and vertical aspects of the marks can be exploited. Specifically, the direction of a mark means the objects along that direction can be selected using the mark. The delete command is an example of this. Preliminary design testing indicated that deleting a series of horizontal events was a very frequent operation. This meant putting the delete command at a horizontal menu position would allow deletion of several events in a row. This "trick" was found to be very useful.

Spatial commonalties can also be used to provide mnemonics to help recall the mark associated with a command. The add and extend commands are examples of this. Both these commands require a vertical time location value. A common way to indicate location along the horizontal is by a vertical "tick". This serves as a mnemonic for the marks associated with these commands.

Temporal aspects

Time versus space pointing: There are many temporal aspects of a mark that can be used. For example, the speed of drawing (i.e., fast or slow, fast at the start then slow at the end) or the time when a drawing occurred can be used. The aspect we exploited is time-based drawing. Specifically, the add command has two modes of operation. The first mode has been described already—the starting location of the mark is used to define the start or end of the event being added. However, if ConEd is synchronized to the playback of a video tape of the conversation, the start or end

point of an event is defined by the current playback location of the video, not by the spatial position of the mark. This is analogous to indicating a point in time by saying "... now" instead of pointing "here". However, users did find that adding events while the tape was playing was difficult.

Inverting semantics of menu items

ConEd's marking menu permits a unique method for undoing. Commands can be undone in ConEd in the standard Macintosh manner (i.e., by pressing the "undo" key or selecting "undo" from the Edit menu). The limitation of this approach is that only the most recent edit can be undone. However, the laugh and ignore commands can also be undone by repeating the laugh or ignore command on the same event. The first laugh mark turns an event into a laugh event. A second laugh mark toggles the event back to a normal event. Therefore, even if these types of edits are not the most recent, they still can be undone.

Toggling the way the laugh and ignore commands work is an example of inverting a menu item semantics. In this case, once a function in a menu is invoked, it is replaced by the corresponding inverse function. Hence, the semantics are "inverted". For example, selecting "open" will invoke the open function and replace the "open" menu item with "close". There are several reasons why inverting semantics are important to marking menus. First, inverting semantics allows extra functions to be associated with a menu without increasing the number of items in a menu. This helps keep the number of items in a marking menu small, which in turn makes marking easier. Second, inverting semantics provides a mnemonic to help recall the association between mark and function. For example, if one remembers the mark associated with "open" then one can recall the mark associated with "close", because the two functions are the inverse of each other.

The role of command feedback

There are several ways that a user receives command feedback using marking menus in ConEd. When using the menu, the user knows which command is about to be executed because the name of the command appears highlighted in the menu. When marking, a user can either recall the mark/command correspondence or watch the results of drawing the mark. We have observed that, as users gain more experience with marking menus, they graduate from watching the menus and

marks, to watching the results of their actions to determine if they have selected the correct command.

Context also plays an important role in determining the command a mark triggers when semantic inversion is being used. For example, events that were marked as "laugh" events appeared in a gray color. This feedback provides essential information to the user that a "laugh" mark on this event was not actually a laugh command but a command to "unlaugh" the event.

In ConEd, a marking menu interaction combines object selection and command application. Typically, in mouse-based direct manipulation systems, these two actions are distinct. For example, a user selects an object by pointing to it; the object then appears "selected"; next, a command is applied to the object by selecting from a menu. When using the marking menu in ConEd objects never appear "selected". It is interesting to note that none of the users ever reported missing it. We can speculate the reason for this is that the combination of selection and marking is intuitive (i.e., emulates our experiences with pen and paper), and the result of a command appeared quickly enough that the starting point of the mark was still in visual image storage.

4.3. ANALYSIS OF USE

The behavior of two users using ConEd over an extended period of time was studied. Both users were employed to edit conversation data. The edited data was used in a research project which was independent of this research thesis. Therefore, a user's main motivation was not to use marking menus, but to complete the task of editing and coding the data. The amount of data to be edited was extremely large and therefore the users were mainly interested in performing the edits as quickly as possible.

The first user (user A) was an experienced Macintosh user and was also familiar with video technology. User A was also familiar with the intentions of the conversation analysis experiment. Given this profile, user A could be considered an expert, although unfamiliar at the start of the study with marking menus. The second user (user B) could be considered a novice. While user B did have some computer experience, it was mainly with the MS-DOS environment, not the Macintosh. Therefore, user B not only had to learn how marking menus worked,

she also had to learn the many details of the Macintosh interface, and the correct way to edit the conversation data.

It was explained to both users how the conversation data was to be edited. The goal of editing was to ensure that the data matched the conversation patterns on the video tape. Users edited the conversation patterns using ConEd and then checked their work by playing back the video tape and comparing the audio of the conversation with the data in ConEd. This process was very interactive. The user played the video and watched the conversation data "playback" on ConEd. When the user saw a piece of data that did not match the audio on the video tape, the user edited the data, then rewound and replayed the video tape and data to ensure the edit was correct.

Each user had the interface to ConEd explained to them and some example edits were performed for their benefit. In particular, the commands in the marking menu were carefully explained and demonstrated. The menu and mark mode was explained and demonstrated, as well as the ability to reselect menu items or confirm a mark. We then verified that the user understood the marking menu by having them perform a few edits using the menu and marks.

Data on user behavior was gathered by recording information about a marking menu selection every time a selection was performed. The information included the time the selection was made, the user's name, the item selected, the mode used to select the item (menu or mark), the length of time the selection took, and the path of the mark or the series of reselections from the menu. A user only needed to register his or her name at the start of an editing session. The rest of the trace data was accumulated transparently.

User A edited for a total of 8.55 hours over approximately six days. User B edited for 10.1 hours over a 29 day period. Most editing sessions lasted one to two hours.

After completing the task, the users were asked to fill out a questionnaire on their experiences using marking menus. The intention of the survey was to reveal users' perception of marking menus and gauge their level of satisfaction.

4.3.1. Issues of use and hypotheses

The main goal for tracing menu usage was to understand how users behave when using marking menus. Specifically, we wanted to find out whether or not in a real

work situation users would evolve from using the menus to using marks and the characteristics of this evolution. In Chapter 2, we described the design of marking menus and how it embodied several assumptions concerning user behavior. The assumptions are that, first, a user will begin by using the menu but with experience the user will evolve to using marks, and second, as part of this evolution, users will make use of intermediate modes of selection (i.e., mark-confirmation and reselection). We wanted to discover whether or not user behavior reflected this in order to prove our assumptions about the novice to expert transition, and to verify that these intermediate modes are actually needed in the marking menu design.

With these goals in mind, we formed the following hypotheses about user behavior with the marking menu in ConEd:

(1) Menu mode will dominate a user's behavior at first. However, with experience, mark mode will dominate.

(2) The more frequently a command is executed the more likely it is to be invoked by a mark.

(3) Users will make use of mark-confirmation and reselection but with experience this behavior will disappear.

The following hypotheses test our assumptions concerning the differences between novice and expert behavior. Specifically, expert behavior will demonstrate faster selection times and more efficient movement than novice behavior.

(4) Time to select from the menu, even with the wait delay subtracted, will be greater than time to make a mark.

(5) With experience, the average length of a mark and time required to make a mark will become smaller.

4.3.2. Results

We analyzed the data from the two users separately for several reasons. First, we were concerned with individual differences. Combining the data would have masked these differences. Second, this study was not a controlled experiment. The data being edited varied, as did the amount of time and number of sessions the users worked. Thus, there was no logical way to merge the users' trace data.

Finally, our two users were very different in attitude and expertise, and therefore combining the trace data would have been inappropriate.

Menu versus mark usage

Hypothesis (1) was shown to be true. Figure 4.9 shows the percentage of times a mark was used to make a selection (as opposed to using the menu to make a selection) versus the total number of selections performed. Over time, marking dominated as the preferred mode of selection. For user A, out of a total of 3,013 selections 6.6% used the menu. For user B, out of a total of 1,945 selections, 45% used the menu.

There are several interesting observations concerning the usage of marks over time. First, when users returned to using ConEd after a lay-off period, the percentage of marking dropped. Figure 4.10 shows that several long lay-offs from ConEd occurred during the study. Note the correspondence between periods of inactivity and dips in mark usage. This indicates that mark/command associations were forgotten when not practiced. However, the amount of fading reduced with the amount of experience (i.e., the dips in Figure 4.9 become less pronounced with experience). Second, note how user B's mark usage rises dramatically at approximately 650 selections. We believe the reason this happened was because user B was a very cautious and inexperienced user. For user B, every command was a new experience. For example, user B needed help opening, saving, and closing files. User B commented that it took her several hours to get comfortable with the video machine and the Macintosh interface before she could begin to think about using marks.

Hypothesis (2) claims that the more frequently a command is used, the more likely it will be invoked by marking. This is based on the assumption that frequent use demands fast interaction and this motivates a user to learn the association between mark and command. Some commands were used more frequently than others. The horizontal axes in the graphs in Figure 4.11 shows this. Hypothesis (2) is shown to be true by a strong correlation between the frequency at which a command was used, and the frequency at which that command was invoked by a mark. Figure

4.11 shows a linear relationship between frequency of command and frequency of marking (for user A, $r^2 = 0.81$, $p<.05$; for user B, $r^2 = 0.88$, $p<.05$)[15].

[15] Note that the add command was not used in this analysis because it appeared to be an outlier point. Its frequency of marking was much lower than the rest of the commands. Our users reported that the add command didn't work correctly all the time. Therefore we assume that users were not as confident about using a marking for the add command as they were for the other commands and hence the outlying mark frequency.

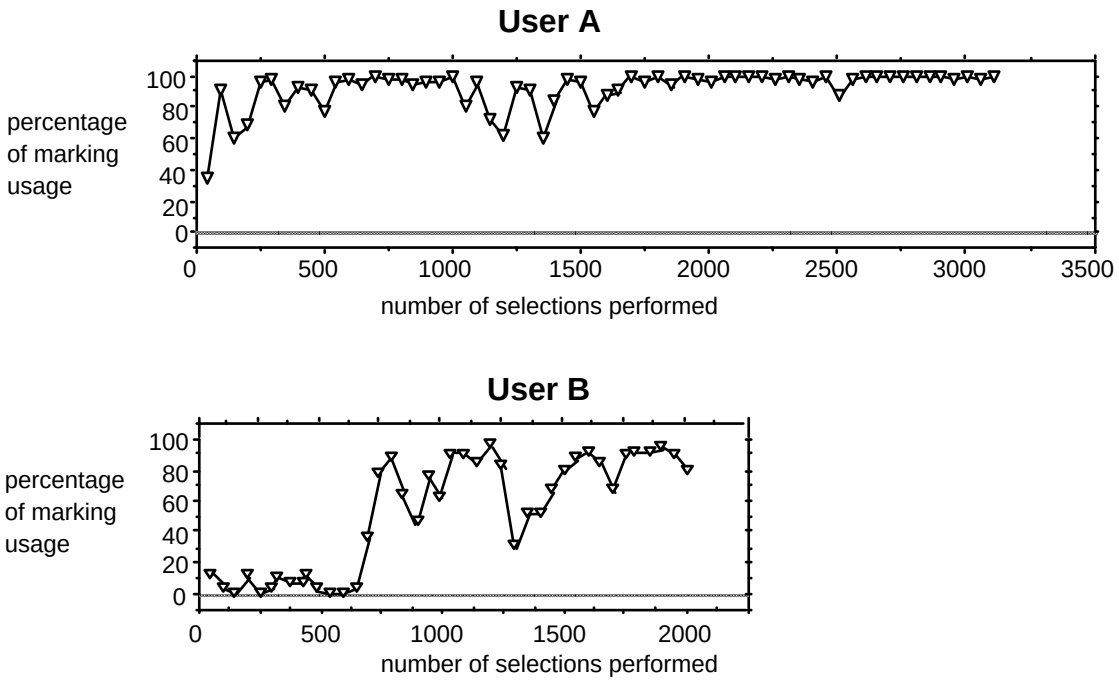


Figure 4.9: With experience, marking becomes the dominate method for selecting a command. Each data point is the percentage of times a mark was used in 50 selections.

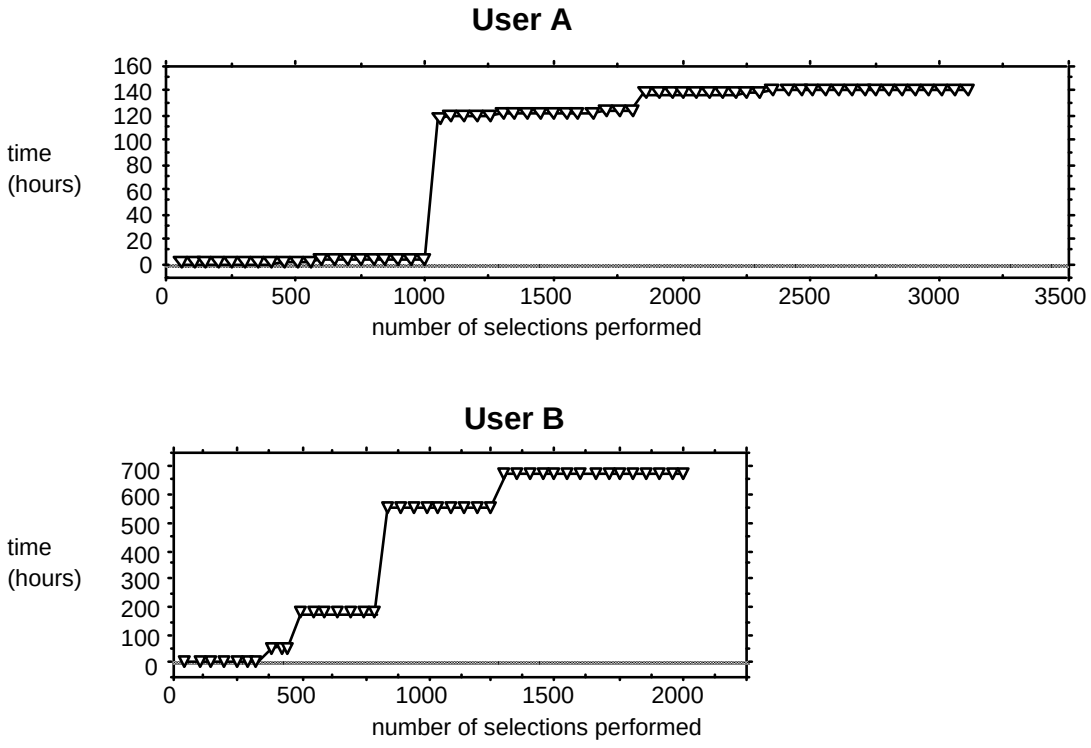


Figure 4.10: Usage of ConEd spanned many days with "lay-offs" between sessions. Steps in the graph represent layoff periods. Dips in the graphs in Figure 4.9 correspond to lay-offs. After a layoff, a user had to resort to the menu to reacquaint oneself with the marks (especially user B).

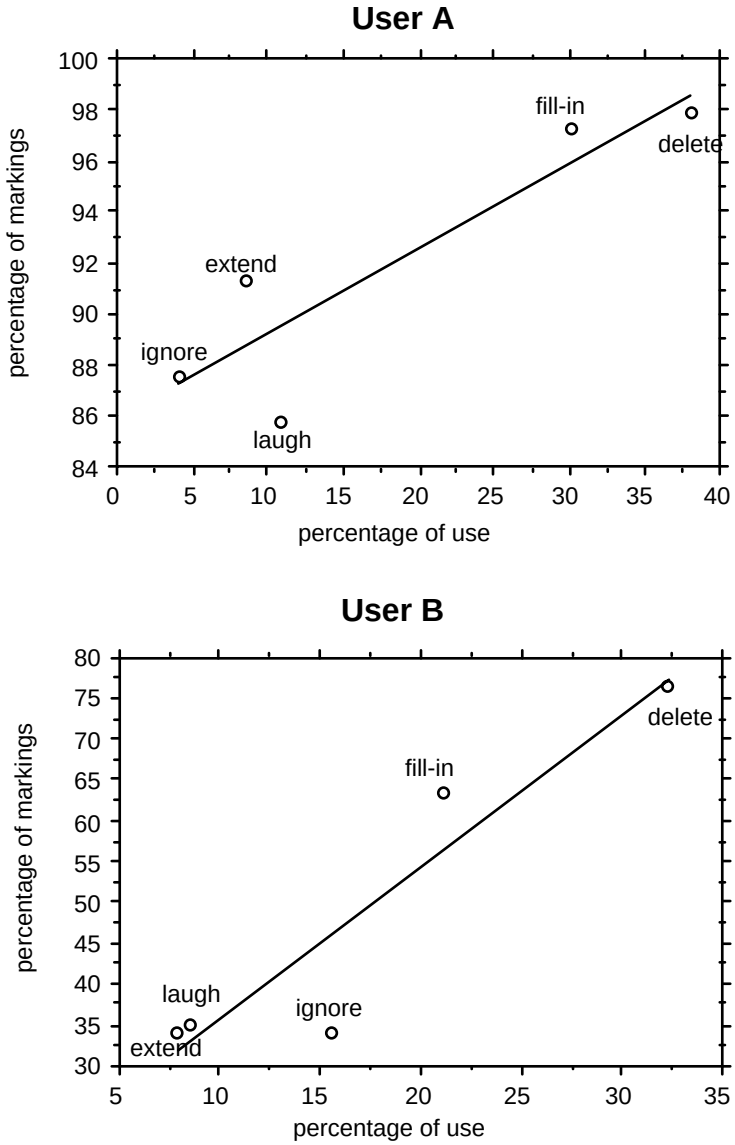


Figure 4.11: The more frequently a command is invoked the more likely it is to be invoked by a mark. The vertical axes show the percentage of times a mark was used to invoke a particular command. The horizontal axes show the percentage of times a particular command was invoked using either a mark or the menu.

Mark confirmation and reselection

As predicted by hypothesis (3), users did make use of the ability to confirm a mark and reselect from a menu but with experience this behavior disappeared. We draw evidence for this from Figure 4.12 as follows. Figure 4.12 (a) plots three types of behavior when using the marking menu:

• *mark*: a selection is made by making a mark;

• *mark-confirm*: a selection is made by making a mark but waiting at the end of the mark, thus popping up the menu to confirm the mark selects the correct item;

• *mark-corrected*: a selection is made in the same manner as "mark-confirm" but after popping up the menu another item is reselected.

We conjecture that these three behaviors are indicative of a user's skill in making accurate marks. Mark is the most skilled behavior. In this case, a user is so skilled at making a mark that no feedback is needed before confirming the selection. Mark-confirm is the next level of skilled behavior. In this case, a user has enough skill to make the correct mark but not the confidence to invoke it without checking it against the menu. Mark-corrected is a third level of skilled behavior. In this case, a user has made a mark, checked it against the menu and has corrected the mark using reselection.

Figure 4.12 shows several things. First, mark-confirm and mark-corrected behavior did occur and therefore this functionality is used and needed. Second, this behavior occurs during the transition from using the menu to drawing marks. Third, when used, this type of behavior occurred less than ten percent of the time.

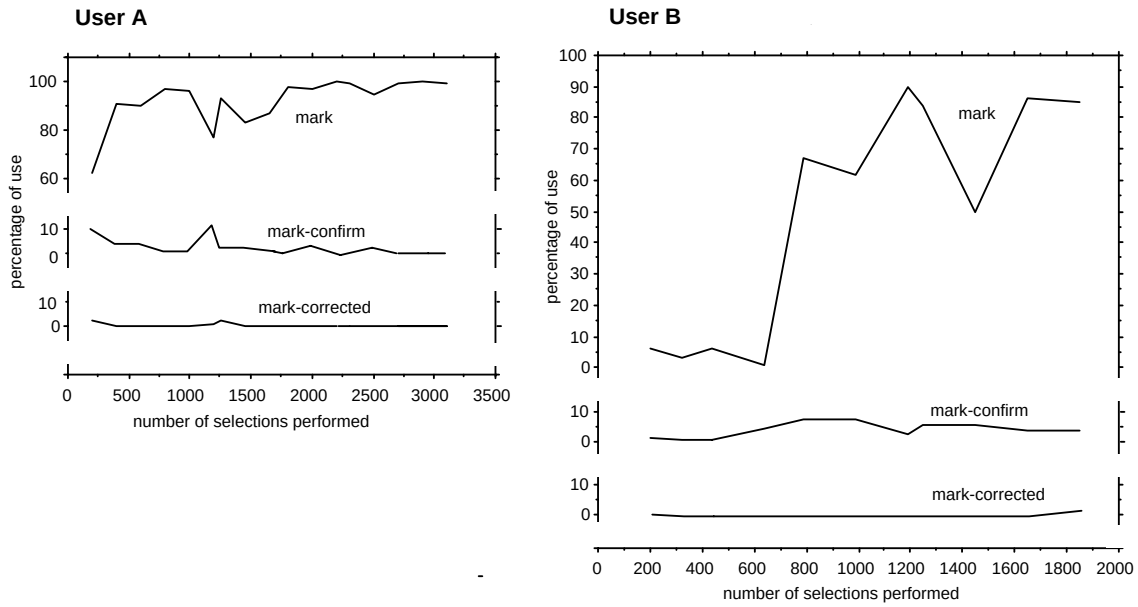


Figure 4.12: Users made use of the ability to confirm the selection a mark would make before committing to it. However, with experience this behavior disappears. Measures were averaged every 200 selections.

Reselection

Another topic of interest was whether or not users reselected when using menu mode. Figure 4.13 shows that reselection occurred less than ten percent of the time. User A demonstrated that with experience reselection disappears. However, user B did not exhibit this behavior. This is evidence that the reselection function in a radial menu is needed.

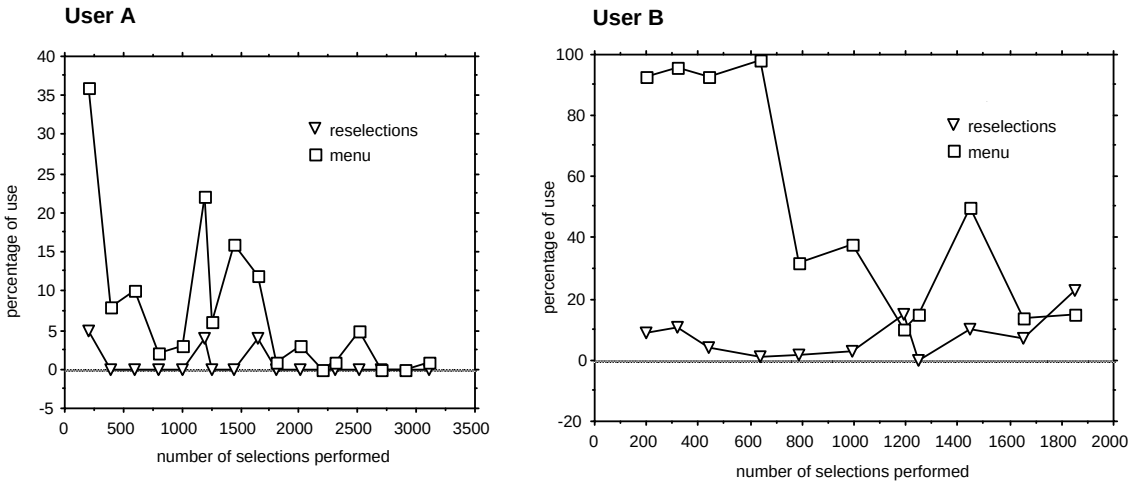


Figure 4.13: Both users utilized reselection in menu mode. While user A's use of reselection diminished with time, user B utilized reselection even after substantial experience. Measures were averaged every 200 selections.

Selection time and length of mark

Selection time is defined as the time elapsed from the moment the mouse button is pressed down to invoke a marking menu, to the moment the button is released, completing the selection from the menu. This measurement applies to either a menu or mark mode. The selection time, for both users, was substantially faster in mark mode than in menu mode. Figure 4.14 shows these differences. For user A, a mark was seven times faster than using the menu. For user B, a mark was four times faster.

Even though menu and mark mode require the same type of movement, using the menu is slower than making the mark. There are several reasons why. First, a user must press-and-wait to pop up the menu. This delay was set to 0.33 seconds. However, as the fourth column in Figure 4.14 shows, even with this delay subtracted from the menu selection time, a mark is still much faster (i.e., user A is 4.2 times faster, user B is 3.0 times faster). The user most likely waits for the menu to appear on the screen. Displaying the menu takes the system about 0.15 seconds. The user must then react to the display of the menu (simple reactions of this type take no more than 0.4 seconds, according to Card, Moran, & Newell, 1983). However, when making a mark, the user does not have to wait for a menu to display and react to its display. Thus, a mark will always be faster than menu selection, even if press-and-wait was not required to trigger the menu. Figure 4.15 graphically shows this. The

	average time to perform a selection (seconds)		
	mark	menu	menu - delay
User A	0.18 ± 0.004	1.097 ± 0.042	0.763
User B	0.404 ± 0.01	1.543 ± 0.052	1.209

Figure 4.14: On average, marks were much faster than using the menu. For user A, a mark was seven times faster than using the menu. For user B, a mark was four times faster. Confidence intervals are at 95%.

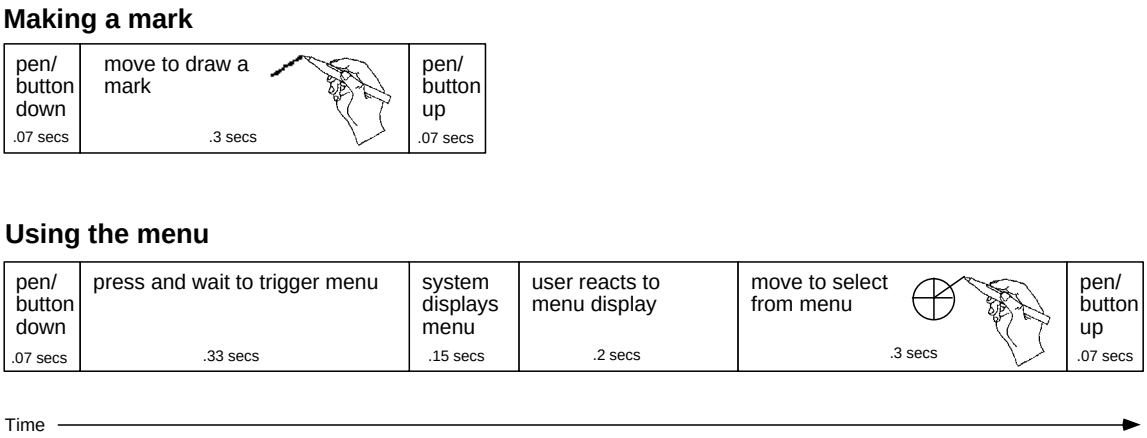


Figure 4.15: Why a mark is faster than using a menu. The typical durations of various events that take place when making a selection are depicted. Even if press-and-wait was eliminated from menu selection it would still take longer than making a mark because of the additional events.

fourth column of Figure 4.14 provides evidence of this. This supports hypothesis (4).

Selection time, using a mark, decreased with practice, however the decrease was very small. In view of the very fast times for marking performance, this is good news, since this means that, even early in practice, novice performance was very similar to expert performance. The decrease in selection time was less than 0.1 seconds. For this analysis we used the Power Law of Practice (performance time declines linearly with practice if plotted in log-log coordinates (Snoddy, 1926)). Linear relationships were found for both users (an analysis of variance of linear

regression used; for user A, $F(1, 1654) = 166.5$, $p<0.0001$; for user B, $F(1, 541) = 23.03$, $p <0.0001$).[16]

The average length of a mark decreased slightly with practice for user B, but not for user A (an analysis of variance of linear regression used; $F(1, 2813) = 10.82$, $p<0.01$). The average length of a mark was approximately one inch. The delete mark was excluded from this analysis because its length was used to indicate a range of events.

Given these results for mark time and length we accept hypothesis (5)–mark time decreases with practice, but only in the case of user B is there support for the hypothesis that mark length also decreases with practice.

Users' perceptions

Both users were given a questionnaire after performing the editing task. The intention of the questionnaire was to discover if a user's perception of marking menus matched their behavior and also to allow us to obtain information not captured in the trace data.

An important parameter not captured in the trace data was selection errors. The reason for this is that prior to a selection we did not know what item a user intended to select. Therefore, when a selection was made, we could not tell whether or not the user had successfully invoked the desired selection. Since users should be the judges of what acceptable error rates are, we simply asked them how many errors they made with the marking menu: no errors, few but acceptable, or too many? Both users reported "few errors but acceptable".

Users perceived marking menus as a tool that helped them get the task completed quickly. Both users reported that their performance with the marking menu was "fast". User B, the less confident user, however, reported she didn't have enough regular experience with the marking menu to be completely comfortable drawing marks.

---

16 Linear relationships were determined by estimating a regression line using an analysis of variance approach (see Appendix A further explanation).

Both users confirmed the differences we found in performance between menu and mark mode. The trace data indicates that using a mark was substantially faster than using the menu. Both users reported a mark was "much faster" than using a menu.

We were also interested in how users recalled the relationship between command and mark. We suggested to both users three methods they could have used to recall mark/command associations. The first is by recollecting the spatial layout of the menu. The second is by rote–"this mark produces this command". The third method is the situation where one is so skilled at performing the mark/command that one is not aware of performing an explicit association—one just "does" the correct mark.[17] User A reported using the second technique, except in the case of "delete" for which he used the third method. User B reported using the first technique. If we assume that the three methods represent various stages of increasing practice, we can conjecture that user A was farther along in expertise and practice than user B. Our data shows this to be true (i.e., user A performed 1,068 more selections than user B).

Marking menus versus linear menus

The results from this study allow us to build on the comparison between marking menus and linear menus discussed in Chapter 2. When a user is familiar with the layout of a menu, selection from a radial menu will be faster than selection from a linear menu. Callahan et. al., (1989) provide empirical evidence of this for eight-item menus. It is possible that a linear menu may be more suitable when there an natural linear ordering to the menu items and a user must search the menu for an item before making a selection. Alternatively, a radial menu may be more suitable when there is a natural radial ordering of menu items. However, as shown by both Card (1982), and McDonald, Stone, & Liebelt (1983), the effects of organization disappear with practice. Callahan et. al., (1989) provide evidence that, for eight-item menus, even when menu items have a natural linear ordering, selection using a radial menu is still faster and less error-prone than selection using a linear menu.

Drawing from data in an experiment by Nilsen (1991), we can directly compare six-item marks and six-item pop-up linear menus. In Nilsen's experiment, a selection

---

[17] Recall may also be by rote in this case, but, since recall is so quick, users may perceive it differently.

from a six-item linear menu required on average 0.79 seconds. In our study, user A and user B required, on average, 0.18 and 0.40 seconds respectively to perform a selection using marks. Furthermore, in Nilsen's experiment the subjects' only task was to select from a linear menu. Therefore, one would expect selection speed to be artificially fast. In our study, in contrast, the users were performing selections in the context of other real world tasks.

The fact that radial menus are faster to select from than linear menus is not the complete story. Selection using a mark is faster than selection via a radial menu. This case study has shown marks to be substantially faster than selection from a radial menu, even if press-and-wait time is factored out. The reason for this is, when selecting using a menu, a user must react to the display of the menu before selecting. However, making a mark involves no reaction time. Hence selection with the mark is faster by design. Obviously faster selection with a mark comes at the price of higher error rates, especially when menus become dense. But the results from this chapter, Chapter 3, and Chapter 5 indicate that menus of breadths four, six and eight have acceptable error rates.

Thus, we can conclude that if menus contain an even numbers of items and less than ten of them, and users frequently use the menus, marking menus will have a distinct advantage over linear menus. Data from this chapter tells us that using the marks will be approximately 3.5 times faster than selecting from a radial menu. We conjecture this speed-up figure would be greater if compared to linear menus.

As a practical example of the impact of this speed-up, we can consider the performance of another real user using ConEd.[18] This user performed 16,026 selections during 36 hours of work. Her average time to select using a mark was 0.23 seconds. Her average time to select using the menu was 1.48 seconds. If the task had been done exclusively with a radial menu that did not use a press-and-wait delay of 0.33 seconds, the average time to select from a menu would have been 1.07 seconds, and 16,026 selections would have required 17,099 seconds in total. However, when using the marking menu, the menu was used for 185 selections and marks were used for 15,841 selections. Thus, menu selections required $185 \times 1.48 =$

[18] A third user used ConEd extensively over a long period of time but she was not included in this study because she assisted in the design of the marking menu used in ConEd and ConEd itself. Therefore, we felt she would not be an unbiased user of marking menus.

274 seconds. Selections made with a mark required 15,841 × 0.23 = 3627 seconds. This results in the 16,026 selections requiring 3901 seconds in total. Thus using a marking menu, as opposed to a radial menu that popped up immediately, saved the user 17,099 - 3,901 = 13,198 seconds or 3.66 hours.

4.4. SUMMARY

This chapter has described a case study which served two purposes. First, the case study involved designing an application that used a marking menu. From this exercise we gained insights on design. Second, data on two users' behaviors using this application to perform a real task was collected and analyzed. Information was collected on a user's performance using a marking menu every time a selection was performed. This information consisted of selection time, selection method, item selected, time of selection, and cursor movement. Analysis of this information allowed us to verify whether or not our assumptions about user behavior, which are embodied in the design of marking menus, are true.

This study demonstrated several things:

- A marking menu was a very effective interaction technique in this setting. Its effectiveness was contingent on applying the technique to an appropriate setting—specifically, using a marking menu to invoke a few commands that are used frequently, and require a screen location as a command parameter. Also, despite the simplicity of the mark, features of the mark, such as the start and end points, and the orientation of the mark, can be used to make interactions more efficient and easier to learn.

- A user's skill with marking menus definitely increases with use. A user begins by using the menu, but, with practice, graduates to making marks. Users reported that marking was relatively error free and empirical data showed marking was substantially faster than using the menu.

- The various modes of a marking menu (menu, mark, mark-confirmation, and reselection) are utilized by users and reflect levels of skills. In addition, when a user's skill depreciates during a long lay-off period, the user utilizes these modes to reacquire skills. We conclude that these features are a necessary part of the design,

and furthermore, interfaces which supply mutually exclusive novice and expert modes are inappropriate when a user's level of skill depreciates.

- In this setting a mark is a very fast way to invoke a command, and users, very early in practice, become skilled at making marks. Evidence of this is that selection time was much faster in mark mode than in menu mode, and did not decrease substantially with practice. This same data indicates that even if the delay time is removed from a menu selection time, menu selection is still slower than marking. This may be due to a user simply moving slower when using the menu. In theory, however, even if there was not press-and-wait delay, and the user moved as quickly in menu mode as they do in mark mode, the user would still be delayed by, first, having to wait for the system to display the menu, and, second, by their own reaction time to its display. Hence, within the limitations on the number of items in a menu described in Chapter 3, we conclude that a mark will always be faster than a menu that immediately pops up. This, of course, is dependent on the user recalling the menu layout.

We can expect hierarchic marking menus to exhibit the same performance properties as non-hierarchic marking menus, since selection from a hierarchic marking menu consists of a series of selections from non-hierarchic menus. Chapter 5 establishes the breadths and depths of hierarchy at which we can expect these properties to hold true.

Chapter 5: An empirical evaluation of hierarchic marking menus

This chapter reports on an experiment to investigate the characteristics of human performance with hierarchic marking menus. Performance using a hierarchic marking menu is affected by the number of items in each level of the hierarchy and the depth of hierarchy. This chapter reports on an experiment which systematically varied these parameters to determine the conditions under which using a mark to select an item becomes too slow or prone to errors. Increasing depth and breadth tends to degrade performance. Thus the intention of this experiment was to find an practical upper bound for these parameters. Understanding of the role of depth and breadth helps us address the types of questions one asks when designing hierarchic marking menus for an interface:

Q1: Can users use hierarchic marks? Chapters 3 and 4 have shown non-hierarchic marking menus to be useful. (Hopkins, 1991) describes how hierarchic pie-menus can be useful. Thus we can expect hierarchic marking menus, even without using marks, to also be useful. However, the question remains: Is it possible to use a mark to quickly and reliably select from a hierarchic radial menu?

Q2: How deep can one go using a mark? Just how "expert" can users become? Can an experienced user use a mark to select from a menu which has three levels of hierarchy and twelve items at each level? By discovering the limitations of the technique we are able to predict which menu configurations, with enough practice, will lead to reliable selections using marks, and which menu configurations, regardless of the amount of practice, will never permit reliable selections using marks. Also, will some items be easier to select regardless of depth? For example, it

seems easier to select items that are on the up, down, left and right axes even if the menus are cluttered and deep.

Q3: Is breadth better than depth? Will wide and shallow menu structures be easier to access with marks than thin and deep ones? Traditional menu designs have breadth/depth tradeoffs (Kiger, 1984). What sort of tradeoff exists for marking menus?

Q4: Will mixing menu breadths result in poorer performance? The experiment on non-hierarchic marking menus described in Chapter 3 has shown that the number of items in a menu and the layout of those items in the menu affects subjects' performance when using marks. Specifically, menus with 2, 4, 6, 8 and 12 items work well for marks. What will be the effect of selecting from menu configurations where number of items in a menu varies from sub-menu to sub-menu?

Q5: Will the pen be better than the mouse for hierarchic marking menu marks? The experiment in Chapter 3 compared making selections from non-hierarchic marking menus using a stylus/tablet, a trackball and a mouse. Subjects' performance was poorest with the trackball while performance with the stylus/tablet and mouse was approximately equal. However, hierarchic marking menus require more complex marks. Will the mouse prove inadequate?

We are also concerned with some pedagogical issues which help us design human-computer interactions. Buxton has described the notion of *chunking* in human-computer dialogs (Buxton, 1986). For example, when using a mark to specify a "move" command, one can issue the command verb, source and destination all in one mark or "chunk". This notion is related to the concept of a "motor program" in motor control studies. A motor program is "a set of muscle movements structured before a movement begins, which allows the entire sequence to be carried out uninfluenced by peripheral feedback" (Keele, 1968).

Some systems or interaction techniques allow chunking to take place while others don't. In some systems a user can articulate a series of operations without having to wait for the system to finish each operation. This allows these commands to be chunked. For example, a user quickly clicks on three graphical buttons without having to wait for each button to complete its operation. In this case, the user may perceive the three clicks as a chunk. If the user was restricted to wait for each button to complete its operation before clicking on the next button, the user may not

perceive the three operations as chunk. Hence, this indicates that something as low-level as input event handling policies can affect user perception and behavior.

Relative to marking menus, the phenomenon of chunking occurs when a user, rather than articulating a selection from a hierarchic menu as a series of directional strokes separated by pauses in movement, performs the entire series of selections in one fluid movement or "chunk". We speculate that chunking is related to expertise. The more expert a user becomes with an interface the more the user chunks. This experiment provides the opportunity to investigate this phenomenon.

5.1. THE EXPERIMENT

5.1.1. Design

In order to determine the limits of performance, we needed to simulate expert behavior. We defined expert behavior as the situation where the user is completely familiar with the contents and layout of the menu and can easily recall the mark needed to select a menu item. To make subjects "completely familiar" with the menu layouts we chose menu items whose layout could be easy memorized. We tested menus with four, eight and twelve items. For menus of four items, the labels were laid out like the four points of a compass: "N", "E", "S" and "W". This type of menu we referred to as a "compass4". Similarly, a "compass8" menu had these four directions plus "NE", "SE", "SW" and "NW". Menus with twelve items, referred to as a "clock" menus, were labeled like the hours on a clock.

Will users of real applications ever be as familiar with menus as they are with a clock or compass? We believe the answer is yes, and base this on three pieces of evidence. First, our own behavioral study of users using a marking menu in a real application (Chapter 4) shows, with practice, they used marks without the aid of menus over ninety percent of the time. Other researchers have reported this type of familiarity with pie menus (Hopkins, 1991). Second, Card (1982), and McDonald, Stone, & Liebelt (1983) report that effects of menu organization disappear with practice. In other words, with practice, users memorize menu layouts and navigate directly to the desired menu item. Finally, it must be remembered that a user does not have to memorize the layout of an entire menu. For example, a hierarchic

marking menu could contain 64 items but the user might only memorize the marks needed to select the two most frequently used menu items.

The design of a trial in our experiment was as follows. A subject was completely familiar with the menu layout and the marks needed to select an item. The system would ask the subject to select a certain item using a mark (the menu could not be popped up by the subject). The subject would input the mark and the system would then record the time taken to draw the mark and whether or not the mark successfully selected the requested item. After a series of trials, we would then vary the menu configuration and input device in order to see what effect these variables had on selection performance.

The rationale for choosing menus of four, eight and twelve items was based on the results from the experiment in Chapter 3. This experiment showed that menus with even numbers of items and less that twelve items were suitable for marking. Using four, eight and twelve item menus is a deliberate attempt to explore a reasonable range of menu breadth. We would expect that performance on a menu of four items to be quite acceptable even at extreme depths. Whereas selection from a menu structure consisting of twelve-item menus which are two levels deep, seems quite treacherous.

Using a similar rationale, we chose to evaluate menu depths from one to four. A depth of one is a non-hierarchic menu which we know from the experiment in Chapter 3 produces acceptable performance. A maximum depth of four was chosen since it is in the range where we believe performance will become unacceptable.

For the sake of brevity we adopt a simple notation in the experiment. A menu structure can be described by a tuple B,D. B is the breadth of each menu in the structure and D is depth of menu structure. For example, 8,2 menu is a menu hierarchy where every menu contain eight items and the hierarchy is two level deep. An 8,2 menu contain 64 leaf menu items. When menu structures consist of different breadth at different levels we use the notation B:B:B, where B is the breadth of a menu at a certain depth. For example, a 12:8:12 menu is menu structure consisting of a top level menu of twelve items, second level menus of eight items and a third level menus of twelve items. A 8,2 menu is represented in this notation as 8:8.

In menu structures of even moderate depth and breadth the number of selectable items becomes very large. For example, in an 8,2 menu there are 64 selectable items.

As stated earlier, we wanted to simulate the case where the user was familiar with the marks being drawn. Given the practical time constrains of the experiment we could not expect subjects to become familiar all marks. Instead we decided to use only three target selections for each menu structure. A subject could then quickly become familiar with the mark needed to make the target selection with a reasonable amount of practice. In this way, the experiment addressed the question: given that the user knows the mark and is practiced at making it, will selection be quick and reliable?

The next issue concerning targets was "which three targets"? For menus of small breadth and depth this was not a major issue as one type of selection is approximately as easy to draw as another. However, in the case of menus which consist of combinations of larger breadths and depths, some selections are definitely harder that others. For example, we observed that making the selection "12-6-3-9" from a 12,4 menu was much easier that "10-11-10-11".

Our approach was to pick three targets for each menu configuration such that one was easy, one was moderately difficult, and one was difficult. Easy targets were those that had items along the vertical and horizontal axes (on-axis items). Difficult targets were those with items not on the vertical and horizontal axes (off-axis items), and little angle change between items. Finally, targets of moderate difficulty were those with a 50% mix of on-axis and off-axis items, and a 50% mix of little and large angle changes between items. It was hoped this mixture of targets would result in behavior that would be representative of an "average selection".

In the case of menus that contain only on-axis items and large angle changes, we observed, prior to the experiment, that up and to the left selections seemed to be hardest, and down to the right selections seemed to be easiest. Thus we chose hard targets and easy targets accordingly. For moderately hard targets we chose either down-and-to-the-left, or up-and-to-the-right targets.

We also had subjects perform selections from a 12:8:12 menu. This was done so we could observe the effects of combining menus of different breadths in a menu configuration.

5.1.2. Hypotheses

Nine hypotheses are proposed:

(1) Pen outperforms mouse: The subjects will perform better with the pen than with the mouse in terms of response time and errors. Once again, the experiment in Chapter 3 showed that subjects performed marginally better with the stylus/tablet than with the mouse on non-hierarchic marking menus. However as depth increases, marks become more complex to draw and therefore the pen should be a more suitable device.

(2) Increasing breadth increases response time and errors: As breadth increases, response time and error rate will increase. The experiment in Chapter 3 demonstrates this effect for non-hierarchic marking menus. Therefore, we believe this effect will apply to hierarchic marking menus as well.

(3) Increasing depth linearly increases response time: As depth increases response time will increase. We base this on the belief that marks to access deep menu configurations will require more time to draw because they will be longer.

A study by Fischman is the most relevant work to this hypothesis (Fischman, 1984). In the study, subjects used a stylus to tap on a series of metal disks (ranging from one to five disks) that were either arranged in a straight line or in a staircase pattern that required changes in direction of 90û between disks. Changes in direction and different numbers of disks in the series roughly correspond to directional movements and different depths in hierarchical marking menus. Fischman found that response time linearly increased with the number of disks, but changes in direction did not affect response time.

(4) Increasing depth increases errors: As depth increases error rate will increase. As depth increases so does the number of times a subject has to estimate at the angle of mark needed to select an item. Therefore the error rate will increase as the probability of error increases.

(5) Inaccuracies propagate: We hypothesize that the depth at which errors take place will be on average greater than half the depth of a menu structure. Informally, we claim that inaccuracies in one portion of a mark will affect the accuracy of the remaining portion of a mark. Our reasoning is as follows: a subject uses the angle of a partially drawn mark to estimate the angle for the next portion of the mark. Inaccuracy in the first portion of the mark may then propagate into the rest of the mark, eventually resulting in an error. The effect of this is that the probability of an error increases with depth. If the probability of an error was the same at every level,

an error would occur on average at half the depth of the menu structure. However, if the probability of an error increases with depth we should see errors take place on average at a depth greater than half the depth of the menu structure.

(6) Mixing menus degrades performance: Combining menus of different breadths in a menu configuration will degrade response time and increase the error rate relative to menu configurations where all menu breadths are the same. Specifically, we hypothesize that subjects will perform better on a 12:12:12 menu than on a 12:8:12 menu, even though an eight-item menu is easier to select from than a twelve-item menu. We believe it is easier for users to select items when the difference of stroke angle needed to select different items is consistent. For example, in a menu structure consisting exclusively of eight-item menus, all items are at 45 degree angles. If a twelve-item was introduced into the menu structure, some items would be at 45 degrees while others, the ones from the twelve-item menu, would be at 30 degree angles. We believe inconsistency in "item angle" will degrade performance.

(7) On-axis items enhance performance: Marks that consist of on-axis items will be faster to draw and produce fewer errors than marks that consist of off-axis marks. This hypothesis is based on prior practical experiences using hierarchic marking menus.

(8) Drawing direction affects performance: The direction of drawing will affect performance. Specifically, marks that require drawing left to right will be performed faster than marks that require drawing right to left. Other researchers have found a similar bias in directional movements (Boritz, Booth, & Cowan, 1991; Malfara & Jones, 1981; Guiard, Diaz, & Beaubaton, 1983).

(9) Subjects will chunk: The number of pauses when drawing a selection will approach zero with practice. Once a subject starts to think of selection not as a series of strokes at certain angles, but as a mark of a certain shape, the subject will draw the mark without pauses between strokes. This hypothesis was based on our own experiences using marking menus in the laboratory.

5.1.3. Method

Subjects: Twelve right handed subjects were recruited from University of Toronto. All subjects were skilled in using a mouse but had little or no experience using the pen on a pen-based computer.

Equipment: A Momenta pen-based computer development system was used. The input devices consisted of a Microsoft mouse for IBM personal computers, and a Momenta pen and digitizer. The digitizer was transparent and placed over the screen. This allowed subject to "write on the screen" with the pen. The screen was placed in front of the subject at approximately a 45 degree angle. When using the pen the hand could be rested on the screen. The mouse was placed to the right of the screen on a mouse pad. No mouse acceleration was used and the sensitivity of the mouse was set to a value of 50 in the control panel. A setting of 50 corresponds to a one to one C:D ratio.[19]

Task: A trial occurred as follows. The type of menu structure being tested appears in the top left corner of the screen. A small circle appears in the center of the screen. A subject then presses and holds the pen or mouse button over the circle. The system then displays instructions describing the target at the top center of the screen. A subject then responds by drawing a mark that is hoped to be the correct response. The system responds by displaying the selection produced by the mark. If the selection did not match the target, the system beeps to indicate an error. The system then displays each menu in the current menu structure at its appropriate location along the mark and indicates the selection from each menu. The subject's score would be shown in the lower left of the screen. Figure 5.1 shows the experimental screen at this point. If the selection is incorrect, a subject loses 100 points and the trial is recorded as an error. If the selection is correct, the subject earns points based on how quickly the response was executed. Response time is defined as the time that elapsed between the display of the target and the completion of the mark.

A subject's score (accumulated points) is displayed in the lower left corner of the screen plotted against current trial number. The graph also shows the best score for that particular pairing of menu structure and input device. This gives subjects a performance level to compete against. This helped to ensure that subjects performed the task both quickly and accurately.

A subject's progress through the trials was self paced. Subjects could pause between trials for as long as they liked. Subjects used this pause to check their score and rest.

---

[19] See Section 2.5.3 for the definitions of C:D ratio and mouse acceleration.

Most subjects paused just a few seconds. All subjects required approximately one hour and fifteen minutes to complete the experiment.

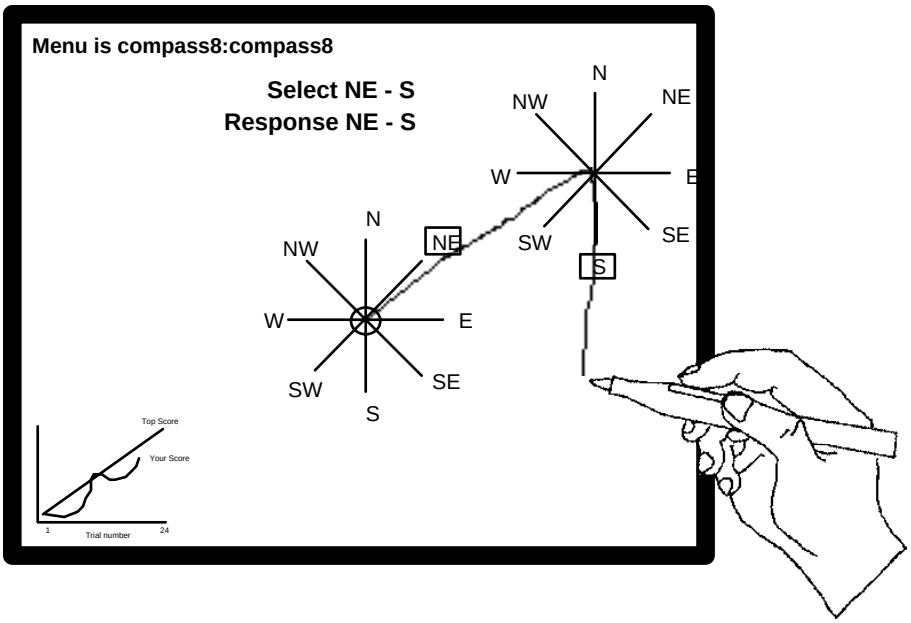


Figure 5.1: The experiment screen at the end of a trial where the target was "NE-S". After the mark is completed, the system displays the menus along the mark to indicate to the subject the accuracy of their marking.

Design: All three factors, device, breadth and depth were within-subject. Trials were blocked by input device with every subject using both the pen and the mouse. One half of the subjects began with the pen first while the other half began with the mouse. For each device, a subject was tested on the thirteen menu structures (breadths 4, 8 and 12 crossed with depths 1 to 4, plus the mixed menu structure of 12:8:12). Menu structures were presented in random order. For each menu structure a subject performed 24 trials. For the 24 trials, subjects were repeatedly asked to select one of three different targets. Each target appeared eight times in the 24 trials but the order of appearance was random.

Given this design, for each data point (a particular combination of input device and menu structure) 288 selections were collected (24 selections times 12 subjects). For the experiment, 7,488 selections were performed in total.

Before starting a block of trials for a particular menu configuration, subjects were allowed eight seconds to study the menu configuration. Before starting trials with a particular input device, a subject was given ten practice trials using the device on a 4,3 menu. This was intended to acquaint a subject with the "feel" of the input device.

It can be argued that the practice session on the 4,3 menu gave subjects an unfair advantage on this particular menu. We believe the effect was small for several reasons. First, a different set of targets was used for practice than those used in the timed trials so subjects did not become practiced at drawing the targets for the timed trials. Second, because of our choice of obvious menu labels and structure for all menus, a subject was already familiar with all of the menu structures even before practice.

5.2. RESULTS AND DISCUSSION

The main dependent variables of interest were response time and percentage of errors. Response time was defined as the time that elapsed between the display of the target and the completion of the mark. Percentage of errors was the percentage of incorrect selections out of 24 trials on a particular combination of device, breadth and depth. Figures 5.2 and 5.3 show the means tables.

Response time averaged across all subjects, breadths and depths for the pen was 1.69 seconds, while the mouse was significantly slower at 2.07 seconds ($F(1,11)=19.7$, $p < .001$). The subjects produced significantly more errors with the mouse than with the pen ($F(1, 11)=6.41$, $p < .05$). Subjects' performance with the pen was better than with the mouse both in terms of response time and percentage of errors, and therefore we accept hypothesis 1.

Breadth significantly affected both response time ($F(2,22)=91.7$, $p < .001$) and errors ($F(2,22)=130.5$, $p < .001$). Figure 5.3 shows, in general, that increasing breadth increases response time and percentage of errors. Based on these results we accept hypothesis 2.

Depth significantly affected both response time ($F(3,33)=195.4$, $p < .001$) and errors ($F(3,33)=51.5$, $p < .001$). Figure 5.3 (a) shows a linear increase in response time as depth increases. Linear regression on each device, menu breadth pair verifies this

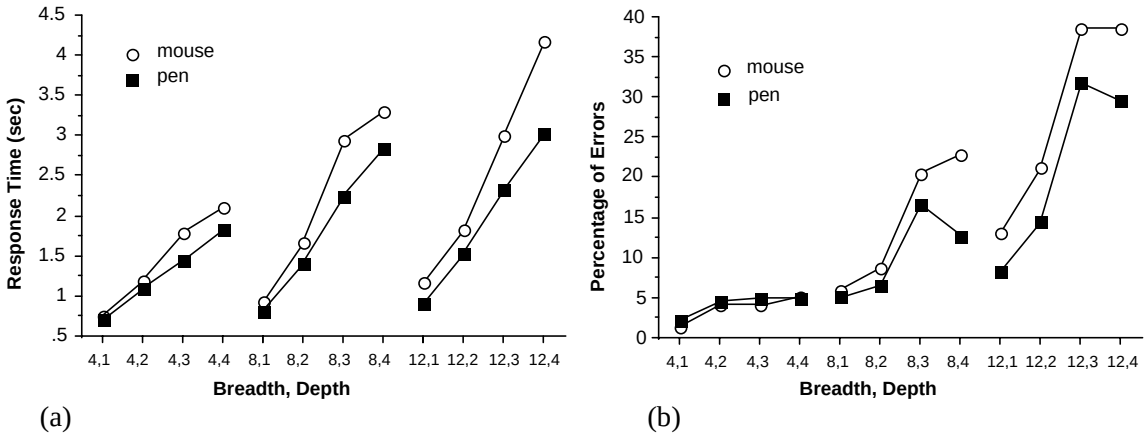
claim (for the pen: breadth four, $r^2 = 0.79$, breadth eight, $r^2 = 0.88$, breadth twelve, $r^2 = 0.82$; for the mouse: breadth four, $r^2 = 0.73$, breadth eight, $r^2 = 0.77$, breadth twelve, $r^2 = 0.67$; $p < .001$ for all values). Figure 5.3 (b) shows that as depth increased so did percentage of errors. Given these results we accept hypotheses 3 and 4.

		Depth					
Breadth	**Device**	**1**	**2**	**3**	**4**	**Total**	**mouse & pen**
4	**mouse**	.752 (.146)	1.189 (.188)	1.797 (.407)	2.102 (.445)	1.460 (.616)	1.367 (.554)
	pen	.710 (.108)	1.098 (.142)	1.451 (.275)	1.835 (.298)	1.279 (.473)	
8	**mouse**	.932 (.211)	1.665 (.543)	2.938 (.829)	3.309 (.845)	2.211 (1.159)	2.021 (1.047)
	pen	.810 (.142)	1.411 (.298)	2.258 (.420)	2.843 (.698)	1.831 (.895)	
12	**mouse**	1.170 (.289)	1.842 (.407)	3.011 (.763)	4.181 (1.363)	2.551 (1.406)	2.250 (1.208)
	pen	.915 (.236)	1.531 (.266)	2.331 (.519)	3.022 (.443)	1.950 (.888)	
Total	**mouse**	.951 (.278)	1.565 (.484)	2.582 (.877)	3.197 (1.272)	2.074 (1.194)	
	pen	.812 (.186)	1.347 (1.272)	2.013 (.572)	2.567 (.724)	1.685 (.826)	
	mouse & pen	.881 (.245)	1.456 (.415)	2.298 (.789)	2.882 (1.075)		

Figure 5.2: Means table for response time. Each entry is average response time in seconds. Standard deviation is shown in parentheses.

		Depth					
Breadth	**Device**	**1**	**2**	**3**	**4**	**Total**	**mouse & pen**
4	**mouse**	1.43 (2.81)	4.18 (4.35)	4.24 (3.59)	5.10 (4.20)	3.74 (3.92)	3.94 (4.65)
	pen	2.19 (5.09)	4.59 (4.63)	4.91 (5.97)	4.89 (5.69)	4.15 (5.32)	
8	**mouse**	5.90 (4.85)	8.82 (4.62)	20.44 (8.60)	22.98 (9.64)	14.51 (10.18)	12.42 (9.93)
	pen	5.21 (7.13)	6.64 (6.56)	16.71 (9.84)	12.77 (9.26)	10.33 (9.34)	
12	**mouse**	13.19 (6.37)	21.26 (8.79)	38.56 (14.98)	38.58 (12.06)	27.90 (15.45)	24.50 (16.26)
	pen	8.33 (8.33)	14.61 (14.91)	31.87 (14.13)	31.87 (14.13)	21.09 (16.48)	
Total	**mouse**	6.84 (6.84)	11.42 (9.51)	21.08 (17.32)	22.19 (16.52)	15.38 (14.69)	
	pen	5.24 (7.24)	8.62 (10.46)	17.83 (15.5)	15.74 (14.90)	11.86 (13.29)	
	mouse & pen	6.04 (7.04)	10.12 (10.02)	19.46 (16.24)	18.96 (15.95)		

Figure 5.3: Means table for percentage of errors. Standard deviation is shown in parentheses.



(a) (b)

Figure 5.3: Response time and percentage of errors as a function of menu breadth, depth and input device. Each data point is the average of 288 trials.

All three factors, input device, breadth and depth affected response time. Analysis of variance revealed a three way interaction between input device, breadth, and depth ($F(6,66)=3.32$, $p < .05$) affecting response time. Figure 5.3 (a) shows these relationships. As one would expect, increasing breadth and depth increases

response time, however subjects' performance degraded more quickly with the mouse than with the pen.

Both depth and breadth interacted to affect error rate ($F(6,66)=12.28$, $p < .0001$). Variance in the error data is large, so the curves in Figure 5.3 (b) must be interpreted carefully. Individual comparisons of error means revealed no significant differences for breadth four at any depth. For breadth eight and twelve, the only significant change in error rate occurred between depth two and three ($F(1, 11) = 23.85$, $p < .001$; $F(1, 11) = 60.52$, $p < .0001$). This indicates that the "rolling off" of the errors curves for breadths eight and twelve between depths two and three is not statistically significant but the increase between depths two and three is significant.

It is important to compare these errors against what we believe would be reliable menu configurations. It seems reasonable that selection from 4,2 menus would be reliable since these marks can be recognized even if drawn very inaccurately. A comparison between 4,2 and 8,2 menus reveals no significant difference. Hence, we have no evidence to claim that eight-item menus, up to two levels deep are more unreliable than 4,2 menus.

A similar comparison between the 4,2 and 12,1 menu revealed a significant difference ($F(1, 11) = 8.25$, $p < .01$). However, the 12,1 menu was not significantly different from the 8,2 menu. Continuing the comparison, we found that the 12,2 menu was significantly different from the 8,2 menu ($F(1, 11) = 21.11$, $p < .0001$). Hence, we claim that the 12,1 menu borders on being unreliable. Section 5.2 has further interpretations on these results.

Hypothesis 5 (inaccuracies propagate) was shown to be true. As depth increased, the average depth at which errors occurred became significantly greater than half the depth of the hierarchy ($F(3,33)=7.62$, $p < .001$). However, the input device had an effect on this behavior. Figure 5.4 shows the pen consistently demonstrated this effect but the mouse exhibited a more erratic behavior.

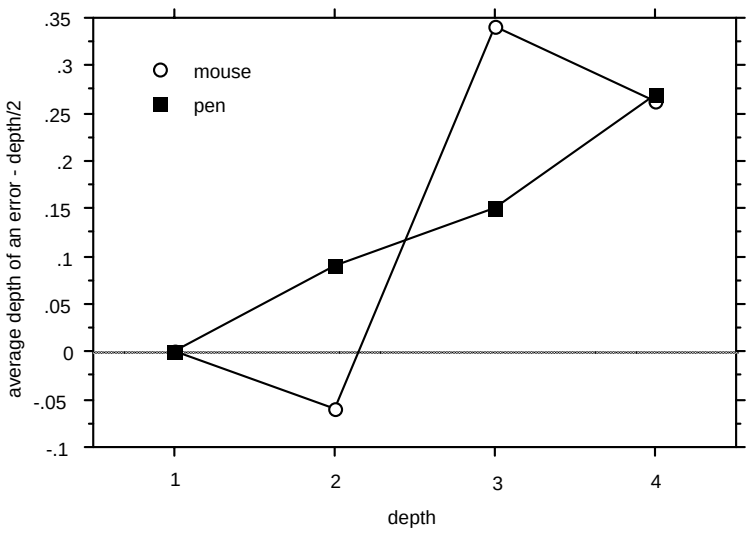


Figure 5.4: Average depth of error - depth/2 versus depth. Depth is the depth of the menu structure being selected from.

We tested the effects of mixing menu breadths in menu configurations by comparing the performance of a 12:12:12 menu with a 12:8:12 menu. We found no significant performance difference between the two menu structures. Therefore, we have no evidence that hypothesis 6 (mixing menus degrades performance) is true.

In order to test the hypothesis 7 (on-axis items enhance performance), targets for the 12,2, 12,3 and 12,4 menus were picked such that the experiment data could be divided into 3 groups. With each group we associated an "off-axis-level": a1, a2 and a3. Experimental data was placed in group a1 if the target consisted only of menu items that were on-axis, such as "12-3-9-3." Group a3 consisted of data on targets that consisted of entirely off-axis targets such as "1-2-1-2". Group a2 consisted of data on targets that were a mixture of on-axis and off-axis menu items, such as 12-7-3-9. Figure 5.5 shows that axis level had a significant effect on response time ($F(2,22)=104.84$, $p < .001$), and on percentage of errors ($F(2,22)=36.2$, $p < .001$). Figure 5.5 (a) shows how the type of device interacted with off-axis level ($F(2,22)=6.93$, $p < .05$). This indicates that subjects response time using the pen did not degrade as much as their response time with the mouse on the worse off-axis targets.

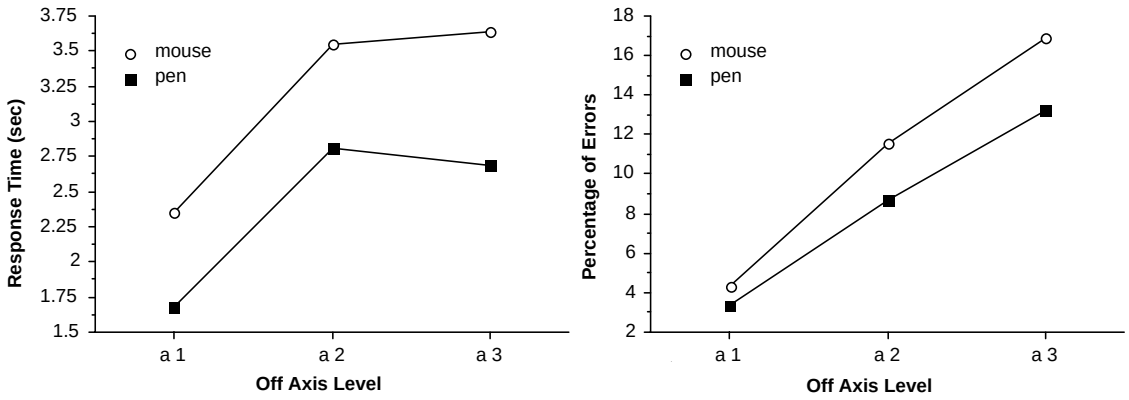


Figure 5.5: Average response time and percentage of errors for targets with an increasing number of "off-axis" items.

In order to evaluate hypothesis 8 (drawing direction affects performance), targets were picked for 4,3, 4,4, 8,3, and 8,4 menus such that mirror image pairs of targets could be compared. For example, the target N-W-N-W was compared with the target N-E-N-E. No significant different in response time was found between "left" and "right" direction targets. Therefore this experiment provides no evidence that hypothesis 8 is true.

The data was analyzed for learning effects by examining performance after every sixth trial. Figure 5.6 shows the results. Response time dropped over 24 trials ($F(3,33)=59.227$, $p<.0001$). Percentage of errors dropped as well ($F(3,33)=8.294$, $p<.0003$). This shows that not only were subjects getting faster but also producing fewer errors. No significant performance differences were found between trial 18 and trial 24. It may be possible that, because subjects were only selecting from three targets, their performance was beginning to asymptote by trial 24.

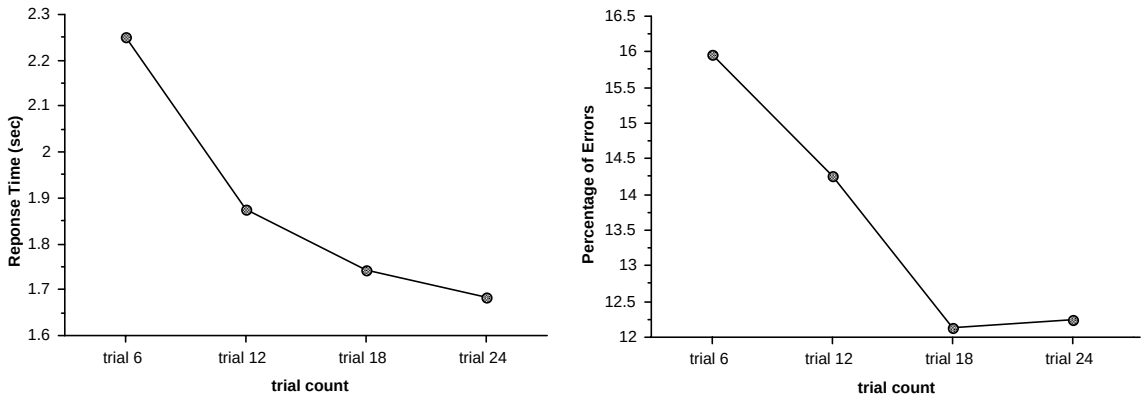


Figure 5.6: Average response time and percentage of errors after every sixth trial.

We analyzed the data for the number of pauses that occurred as a selection was being drawn. A pause was defined as the pen or mouse not moving more than five pixels for more than 1/2 of a second. Figure 5.7 shows that, as users gained experience the number of pauses dropped ($F(9,99)=38.409$, $p < .0001$). This is evidence that subjects, with experience, began to draw a mark not as a series of discrete selections but as a single mark of a certain pattern (assuming that when pauses did occur they occurred between different selections). The number of pauses did not fall all the way to zero because some of the most difficult targets required careful drawing which resulted in pauses. Given these results we accept hypothesis 9 (subjects will chunk).

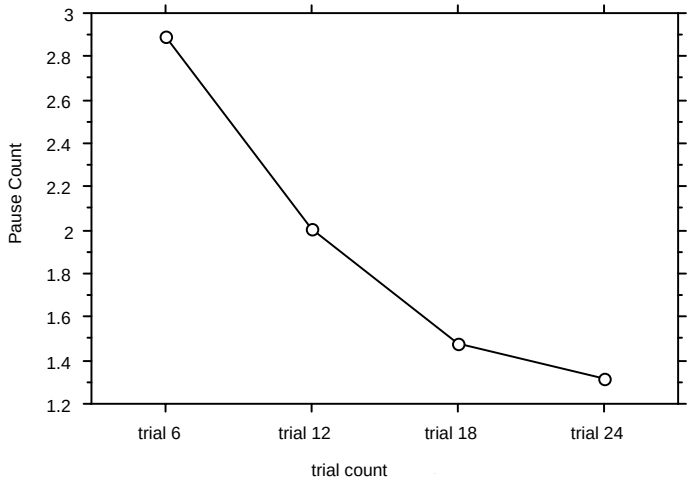


Figure 5.7: Average number of pauses counted after every six trials.

We also gave subjects a questionnaire after the experiment. This was to elucidate subjects' perception of their own performance and compare some of the experimental data with subjects' perceptions.

Eleven out the twelve subjects thought making selections with the pen was faster and more accurate than with the mouse. This agrees with the data from the experiment. Also, when asked to comment on the experiment, four subjects reported that, although their performance with the mouse was fast, they found the mouse required more effort.

We wanted subjects' opinion on the accuracy of our mark recognizer. In some cases, for example, menus which are only one level deep, recognition is simple. In this case, only the start and end points of the mark need to be examined to determine the item picked. However, at depths greater than one, submenu selections must be determined so changes in direction along a mark must be recognized. The algorithm for determining these "kinks" along a mark is complex because it has to handle dense menus and marks that are drawn sloppily. Typical of most recognition systems, occasionally what appears to the subject as a correct mark is misinterpreted by the system. However, on average, subjects claim that this happened only three percent of the time. This is an acceptable recognition rate by mark recognizer standards (Sibert, Buffa, Crane, Doster, Rhyne, & Ward, 1987). Nonetheless, after observing the type of recognition errors that occurred during the experiment, we believe the recognition rate can be further improved by a few refinements to the recognizer algorithm.

Another phenomenon that occurred in the experiment was subjects selecting the wrong direction by accident. For example, the screen would display "select N" and the subject would select south. Errors of this type are referred to as "mental slips" (Norman, 1981). These types of errors were removed from the data set before analysis because they are not caused by drawing inaccuracies. Other errors such as clear cut errors on part of the recognizer were also removed from the data. Subjects reported several causes for mental slips: "I just goofed" or "I started to draw the mark from the previous trial". Subjects, on average, claimed that mental errors occurred two percent of the time. This approximately agrees with the data: we found a one percent error rate for clear cut "mental slips". We do not feel these

errors are particular to drawing marks—mental slips are common in any human activity (Norman, 1981).

We hypothesized before the experiment that drawing marks that were predominately left to right movements would be easier, and hence faster, than right to left marks. However, our analysis of the data showed drawing direction had no significant effect on selection times. This agrees with the results of the questionnaire: six out of the twelve subjects thought left to right marks were easier to draw that right to left marks. This even split among subjects perhaps explains the non-significant effect of direction. A closer examination of the data might reveal individual effects.

5.3. CONCLUSIONS

We can now revisit the questions posed at the start of this chapter and interpret the results of this experiment.

Q1: Can users use hierarchic marks? Even if using a mark to access an item is too hard to draw or cannot be remembered, a user can perform a selection by displaying the menus. Nevertheless, since the subjects could perform some of the marks in the experiment with acceptable response times and error rates, marking is a usable method of selection.

Q2: How deep can one go using a mark? Our data indicates that increasing depth increases response time linearly. The limiting factor appears to be error rate. Error rate was found to rise significantly for menus beyond the 8,2 menu. 8,2 menus were not any more unreliable than 4,2 menus. Common sense tell us that the marks required to select from a 4,2 menus are not difficult to draw. Hence we consider menu configurations which did not significantly differ in error rate from 4,2 menus to be reliable. It seems reasonable to recommend using menus of breadth four, up to depth four, and menus of breadth eight, up to depth two. 12,1 menus border on unreliability.

Off-axis analysis indicates that the source of poor performance at higher breadths and depths is due to selecting off-axis items. Thus, when designing a wide and deep menu, the frequently used items should be placed at on-axis marks. This would

allow some items to be accessed quickly and reliably with marks, despite the breadth and depth of the menu.

What is an acceptable error rate? The answer to this question depends on the consequences of an error, the cost of undoing an error or redoing the command, and the attitude of the user. For example, there is data that indicates, in certain situations, experts produce more errors than novices (Sellen, Kurtenbach, & Buxton, 1990). The experts were skilled at error recovery and thus elected to sacrifice accuracy for fast task performance. Our experiences with marking menus with six items in a real application indicate that experts perceived selection to be error-free. Other research reports that radial menus with up to eight items produce acceptable performance (Hopkins, 1991). Marking menus present a classic time versus accuracy tradeoff. If the marking error rate is too high, a user can always use the slower but more accurate method of popping up the menus to make a selection.

Marking error rates can be compared to linear menu error rates but one must be very cautious when comparing results from different experiments and different interaction techniques. Even within the same experiment, subjects may not consistently perform at the same level of accuracy, or the experimental task may artificially inflate or deflate the error rate. We can, however, make some approximate comparisons. In a study of selection performance using pop-up hierarchic linear six-item menus of depth two, Nilsen (1991) reports error rates of 2.3%. Nilsen also reports that subjects accidentally popped up the wrong submenu on their way to making a correct selection 6.3% of the time. In another study of similar pop-up linear menus, Walker, Smelcer, & Nilsen (1991) report error rates that range from 2.0% to 12.6% for subjects selecting from nine-item menus of depth 2. These error rate figures are in the range of the error rates found in our experiment for menus of up 8,2 menus. Therefore, we can conclude, with caution, that marks, within the limits discussed above, can be as accurate as selection from linear menus. It is also critical to note that this level of accuracy is not the expense of speed. For example, in this experiment selection from 8,2 menus required on average 1.5 seconds. In Nilsen's experiment, selection from six-item linear menus of depth 2 required on average 1.8 seconds (six-item menus should be faster). We found that, comparing the data from Nilsen and Walker experiments with this experiment, for equivalent menu configurations, selection from linear menus is slower than selection using marks.

Q3: Is breadth better than depth? For menu structures that resulted in acceptable performance, breadth and depth seems to be an even tradeoff in terms of response time and errors. For example, accessing 64 items using 4,3 menus, is approximately as fast as using 8,2 menus. Both have approximately equivalent error rates. Thus, within this range of menu configurations, a designer can let the semantics of menu items dictate whether menus should be narrow and deep, or wide and shallow.

Q4: Will mixing menu breadths result in poorer performance? The experiment did not show this to be true. One possible explanation is that our menu labels strongly suggested the correct angle to draw and thus eliminated confusion. A stronger test would use less suggestive labels when mixing breadths. Our results do indicate that, when there is enough familiarity with the menus, mixing breadths is not a significant problem.

Q5: Will the pen be better than the mouse for marking menu marks? Overall, the pen proved to be more suitable. However, for small menu breadths and depths, the mouse produced approximately equivalent performance. We found this extremely encouraging because it implies that marking menus are an interaction technique that not only takes advantage of the pen but also remains compatible with the mouse. Of course, it is worthwhile to note that some subjects thought their performance with the mouse was just as good as with the pen, but that the mouse required more effort to attain this level of performance.

These conclusions should be tempered by reminding the reader that this experiment simulated an expert situation (i.e., subjects were asked only to select from three different targets, thus they quickly became "expert" at those targets). We have argued that this situation is reasonably realistic. Other realistic situations, such as the performance of users on unfamiliar hierarchic marking menus with varying targets, has yet to be explored.

5.4. SUMMARY

The chapter described an experiment to test the limitations of using marks to select from hierarchic marking menus. Subjects were asked to select from marking menus using marks only. Menus were chosen such that the subject would very quickly learn and remember the mark required to perform a given selection. The breadth and depth of these menus and the input device was then systematically varied to

elucidate the effects of these variables. Subjects' time to perform selections and error rates were collected and analyzed. Subject's perceptions were collected using a questionnaire.

The experiment revealed that error rate was the limiting factor. Menus of breadth 4, 8 and 12 were examined. Error rate became a factor when menu breadth was eight or twelve. For these breadths of menu, error rate rose significantly when depth was greater than two. For these menu structures with acceptable error rates, there appeared to be an even depth/breadth tradeoff. When menus structures contained equivalent numbers of items, subjects showed equivalent performance on both narrow, deep menus and wide, shallow menus. It was also discovered that mixing menus of different breadths in a menu structure did not adversely affect performance. Finally, we concluded that the pen is more suitable for drawing marking menu marks than the mouse, but the difference is not large.

This chapter has answered some basic questions about the design variables of hierarchic marking menus. Specifically, how deep and wide can menu structures be yet still allow a user to perform selections using marks? The following chapter takes the answers to these questions and applies them to the design of hierarchic marking menus in a pen-based application.

Chapter 6: Generalizing the concepts of marking menus

6.1. INTRODUCTION

This chapter reports on a design experiment which deals with applying the design principles of self-revelation, guidance, and rehearsal to interface design. Two issues are explored. First, we examine the ramifications of integrating an interaction technique that is based on these principles (marking menus) into a pen-based interface. We found that it is possible to integrate marking menus into an interface but several compromises needed to be made. Although these compromises change the original design of marking menus, we show that the resulting design still obeys our three design principles. Second, we examine how these design principles can be applied to other types of marks besides zig-zag marks. With this goal in mind, we developed an interaction technique that provides self-revelation, guidance, and rehearsal for these other types of marks. These experiences provide a better understanding of the role of marking menus in interface design and demonstrate the value of the design principles.

The test bed for this design experiment was a pen-based electronic whiteboard application called *Tivoli* (Pederson, McCall, Moran, & Halasz, 1993). Tivoli is intended to be used in collaborative meeting situations, much in the same way that a traditional whiteboard is used. Tivoli runs on a large vertical display, called *Liveboard* (see Figure 6.1) (Elrod et. al., 1992), that can be written on with an electronic pen (see Figure 6.2). Much like a whiteboard, several people can stand in front of a Liveboard and write, erase, gesture at, and discuss hand drawn items.

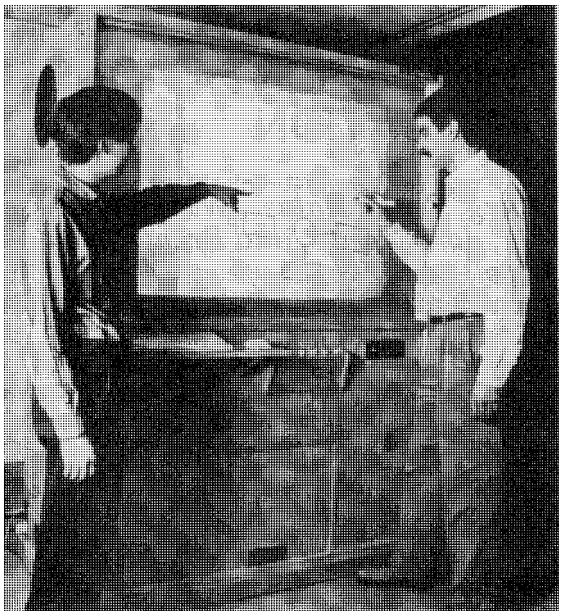


Figure 6.1: The Liveboard in use.

Tivoli, however, does more than just emulate marking on a whiteboard. Marks can be edited, stored, and retrieved. Marks are remembered by Tivoli in terms of *strokes*. A stroke is the path of the pen recorded from the moment the pen is pressed against the screen and moved, until it is released from the screen. A screenfull of strokes can be grouped into a "slide", and saved for retrieval later. Typical operations on strokes include moving or copying groups of strokes, changing the color or thickness of the pen tip, and undoing edit operations. Users draw "edit marks" to perform some of the editing operation described above. Figure 6.3 shows the types of marks used. Other operations are triggered using graphical buttons, dialog boxes, and menus.

One basic goal of our design study was to address the problem of operating extremely large displays. It is envisioned that someday the *Liveboard* display surface would be very large, and therefore, we wanted to address the problem of bringing the commands to the user as opposed to the user moving to the commands. Marking menus seemed suitable for this type of design since the menus can pop up at any location and the marks can be made at any location.[20] Furthermore, since

[20] This is not completely true. Depending on the design of the interface a user may have to be over some particular area or object on the display before a menu can be popped-up or a marking intepreted. However, the point is that pop-up menus and marks help reduce the amount of movement a user must make to invoke functions. For example, when a user wants to change pen color, traditionally one has to move from the drawing

Tivoli has many commands, we felt that hierarchical marking menus might allow access to many of these commands from a single location. The issue was whether or not we could integrate marking menus into the existing Tivoli interface design to solve some of these problems.

Another basic design goal was that Tivoli should be based on the unfolding interface paradigm described in Chapter 2. For example, for a novice Tivoli user the interface presents a limited set of functions—the type of functions one gets with an ordinary whiteboard. However, additional functions can be discovered and used with minimal instruction and experience. In effect, once a user has the "key" to unlock the hidden functionality, Tivoli can be unfolded and additional functions invoked. Using edit marks is a way to hide additional functions. The edit marks are not in themselves self-revealing, and therefore, this serves as a way to hide functions from a novice.

---

area to a color pallet and back. With a pop-up menu, this trip is avoided since the menu can be popped-up over some white space in the drawing area.

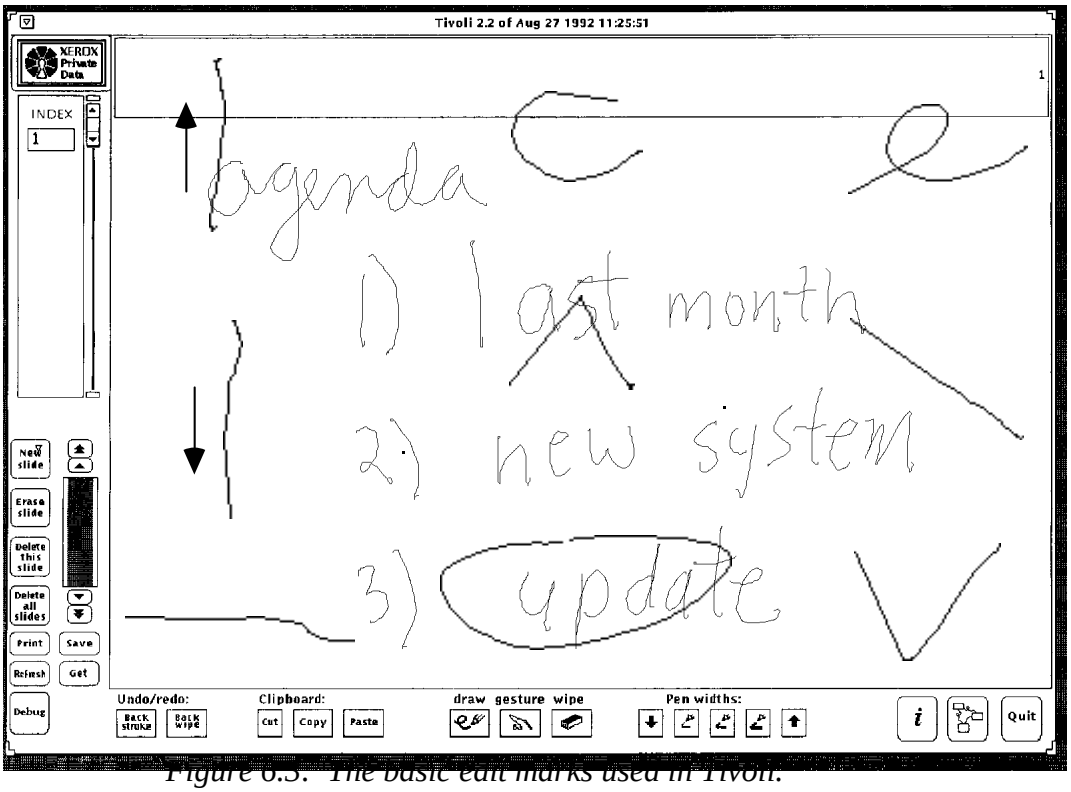


Figure 6.3: The basic edit marks used in Tivoli.

Figure 6.2: An application called Tivoli, running on Liveboard, emulates a whiteboard but also allows drawings to be edited, saved and restored.

Given these basic goals, we explored two problems. The first problem was to determine which Tivoli functions would be suitable for marking menus, and how marking menus could be integrated into the existing interface. The second problem was how to provide self-revelation, guidance, and rehearsal for the edit marks in Tivoli.

6.2. INTEGRATING MARKING MENUS INTO A PEN-BASED INTERFACE

We decided we would explore design issues by using marking menus to control pen settings in Tivoli. In Tivoli, the pen can be set to different colors and thicknesses. Originally, these settings were performed using a pallet of buttons which had an individual button for each pen thickness and color. There were several reasons why it would be advantageous to control these functions using marking menus. First, the original buttons consumed a large amount of screen space. Replacing these buttons with a marking menu would free up this screen space. Second, changing pen

settings was a frequent operation while drawing. Changing settings meant many trips to and from the button pallet. A marking menu could be made to pop up at the drawing location, thus avoiding trips to the button pallet. Third, no intuitive set of marks exist for controlling pen settings. Marking menus could provide a set of marks and a method for learning those marks.

6.2.1. Adapting to drawing and editing modes

Figure 6.4 shows the marking menu we used to control pen thickness and color. The range of items is deliberately small. We felt that, in Tivoli, users need only a few different thicknesses and colors for the pen. This is like real whiteboards, where the number of markers is limited. The menu items “inc” and “dec” allow a user to increment and decrement the pen size to get custom thicknesses. The menu appears when a user presses-and-waits with the pen anywhere in the drawing area. This allows a user to change pen settings without having to move the pen from the current drawing location.

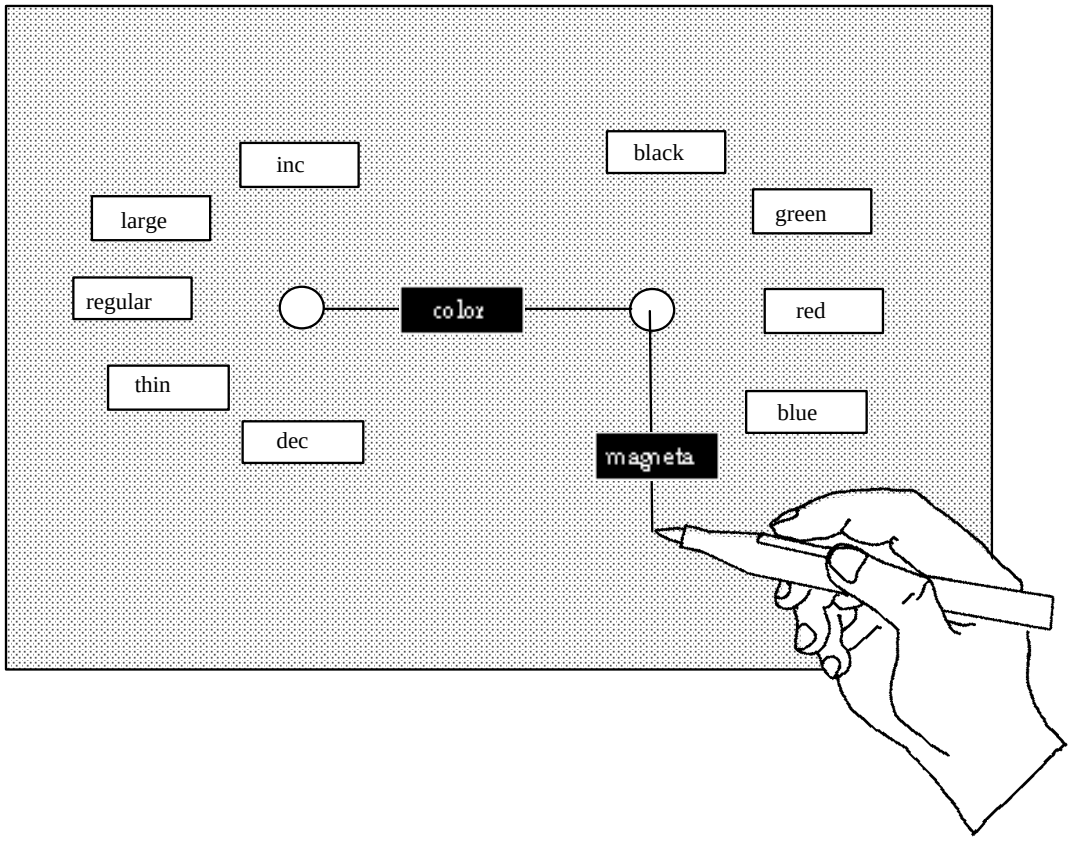


Figure 6.4: The hierarchical marking menu used to control pen settings in Tivoli. The menu can be popped up by pressing-and-waiting instead of drawing.

There is a complication with this design. Normally, a marking menu allows a user the alternative of drawing a mark to select a menu item. However, in the situation just described, Tivoli is in the "drawing mode" (i.e., all marks are interpreted as drawings, not commands). A mark is interpreted as a command in Tivoli when it is drawn while a button on the pen (the command button) is pressed. Thus, the design of the Tivoli's interface requires that menu selection marks (which are actually command marks) be performed with the command button pressed. However, this deviates from the rehearsal principle slightly: the physical action of making a selection mark is the same as selecting from the menu, but the command button must also be pressed. All the directional motions remain the same so we can be hopeful that using the menu still develops skills useful for learning and making the selection marks.

We have no empirical data to verify that, despite this deviation in rehearsal, skills developed in using the menu are still transferred to using the marks. However,

when using Tivoli ourselves, because of our experiences with the menu, we were able to recall the spatial layout of the menu, and issue marks. The role of spatial memory and physical movement memory in the transition from menus to marks is a topic for future research.

6.2.2. Avoiding ambiguity

Typically interfaces that use marks as commands identify marks by the shape of the mark or the context in which the mark is made. This discussion discriminates between marks intended for marking menu selections and other kinds of marks intended for commands. For the sake of brevity in this discussion, we will refer to these other kinds of marks as *iconic marks*, although the meanings of these marks may not be strictly based on iconic shape (see section 6.3.1 for further discussion). Also for the sake of brevity, we refer to marking menu's zig-zag marks as *menu marks*. The important point is that the potential exists for marking menu marks (menu marks) to be confused with iconic marks. Figure 6.5 shows an example of two marks for a menu structure of breadth eight and depth of two which are the same as some of the iconic marks in an early version of Tivoli.

These types of ambiguities are not peculiar to marking menu marks. Many interfaces that use marks exhibit this problem. For example, a classic problem is drawing an "O" for the letter "O" and having it confused with a small circle (where circling performs a selection). We present three strategies for overcoming this problem for marking menus, and the advantages and disadvantages of each one. We then describe how a one of these three strategies was used in Tivoli.

Avoidance

One way to avoid ambiguities between marking menu marks and iconic marks is to eliminate the ambiguous marks from the marking menu set. This can be done by avoiding the placement of menu items at locations in a menu structure that would result in ambiguous marks. These "avoided locations" can be occupied by null menu items. A mark that selects a series of null items is then considered no longer a marking menu mark, and therefore ambiguity is eliminated.

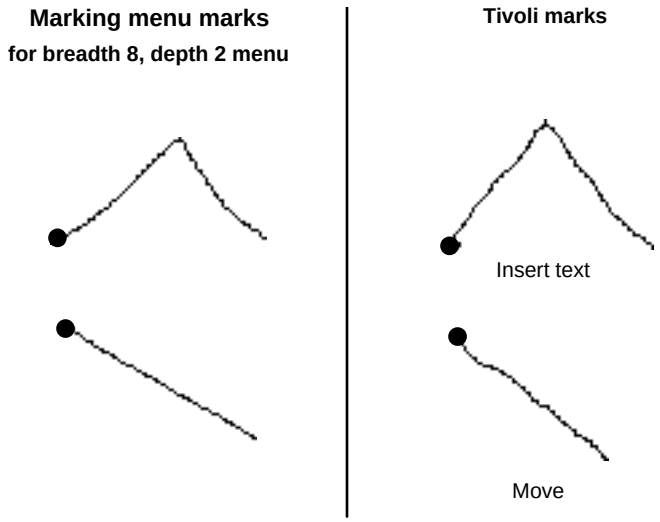


Figure 6.5: The marks used for a marking menu may conflict with other marks. The example shows two marks used for selecting from a marking menu that can be confused with edit marks in an early version of Tivoli. A dot indicates the starting point of the mark.

One drawback to this approach is that the number of items a menu can hold is reduced and "unnatural" gaps may appear in the menus. For example, suppose a menu contains an ordered set of font sizes. If one of the menu items is not used, then a gap appears between two menu items that logically should appear adjacent to one another. This may make learning the layout of the menus more difficult.

Another drawback is that eliminating a menu item from certain location forces that the item to be placed somewhere else. Menu structures can be expanded to hold displaced items either in breadth or in depth. As shown in Chapter 5, expanding in breadth or depth slow menu selection and increases errors. Furthermore, eliminating items may result in losing on-axis items, which have been shown in Chapter 5 to enhance performance. Ultimately, rearranging menus may lead to menus that appear to be oddly structured, and this results in menus that are hard to learn, slow to use and error-prone.

These drawbacks makes avoidance a poor solution. In certain restricted cases, though, it can be a simple and easy solution to implement. For example, suppose the only conflicting mark is a horizontal stroke which is to the right, and the marking menu only needs to contain six items. The simple solution is to use an

eight-item menu and the make the "right stroke" menu item and some other menu item null items, and populate the remaining menu items with the six commands. A variation on this strategy is to change the iconic marks. This, of course, avoids the problems with modifying a marking menu as described earlier, but in certain situations may cause confusion for a user when obvious or common marks are replaced by non-obvious, uncommon iconic marks.

Different context

Another design alternative is to allow iconic marks and marking menu marks of the same shape to coexist but determine their meaning by the context in which they are drawn. Two dimensions in which the context can vary are time (i.e., when the system is in a certain mode a mark has a certain meaning), and by space (i.e., a mark's meaning varies depending on the location at which it is drawn).

Distinguishing the meaning of a mark by the context of time leads to moded interfaces. An interface where a user must enter a "marking menu mode" to issue a marking menu mark seems to defeat the purpose of making a mark—a fast way to invoke a particular command. However, if the cost of switching modes is very low and properly designed (Sellen, Kurtenbach, & Buxton, 1992), this can be an effective technique. An example of low cost mode switching is a dedicated pop-up menu button on the mouse which is found in many windowing systems such as *X11* (X11, 1988) and *Open Look* (Hoeber, 1988). After developing the habit of holding down the button to pop-up and maintain a menu, a user no longer perceives using a menu as a mode. One can imagine such a similar design for marking menus where a user presses down a button on the pen or mouse to indicate to the system that the mark is intended for the marking menu. The obvious disadvantage to this scheme is that a hardware button must be dedicated strictly to a menu. Many pen-based system pens do not have buttons, or the buttons have already been assigned other functions. For example, in Tivoli the two buttons on the pen were already used for other functions. The first button is used to distinguish between drawing mode and command (edit mark) mode. The second button is used to control whether the pen is in drawing mode or erasing mode.

Another type of context that can be used to distinguish the meaning of a mark is location. For example, a stroke through a word may mean "delete the word" while the same stroke starting on a graphic may mean "move the graphic". This type of

scheme works well with object oriented direct manipulation systems, where the combination of an object and a mark can be used to distinguish a mark's meaning. Of course, distinguishing meaning by location will not work if the same location must accept two identically shaped marks.

Marking menus work very well in identification by location situations. For example, on a different project, we found an effective interaction technique can be created by embedding a marking menu in an ordinary graphical button. In effect, this extends the functionality of the button. Along these lines, we developed a system called *HyperMark* which allows marks to be used in Apple's Hypercard (Apple 1992). For example, if HyperMarks are added to a button, not only does a button react to a mouse press, but marks can also be drawn on the button which trigger other actions. This results in the interface having fewer buttons and faster interactions in some cases. In effect, HyperMarks are similar to pop-up menus where additional functions are hidden under a button until popped up. However, with HyperMarks, a user does not have to wait for a menu to pop up, visually search the menu and point to an item. Instead, a mark triggers the item directly. Our intention was to permit ordinary Hypercard users or programmers to incorporate marks into their own Hypercard stacks.

With HyperMark, different buttons accept the same mark but the interpretation of the mark is different. Figure 6.6 shows an example of different locations having different menus but reusing the same set of marks. The meaning of the marks is disambiguated by the location of the mark. We feel this is a reasonable design as long as the common commands (scroll up and scroll down, for example) are kept consistent from button to button.

The disadvantages of discriminating by location are, first, it does not eliminate the problem if the same location accepts two ambiguous commands and second, it consumes screen space. Consumption of screen space results in situations where the desired location is not displayed on the screen and must be acquired by the user. This can slow interactions and defeat the purpose of using marks.

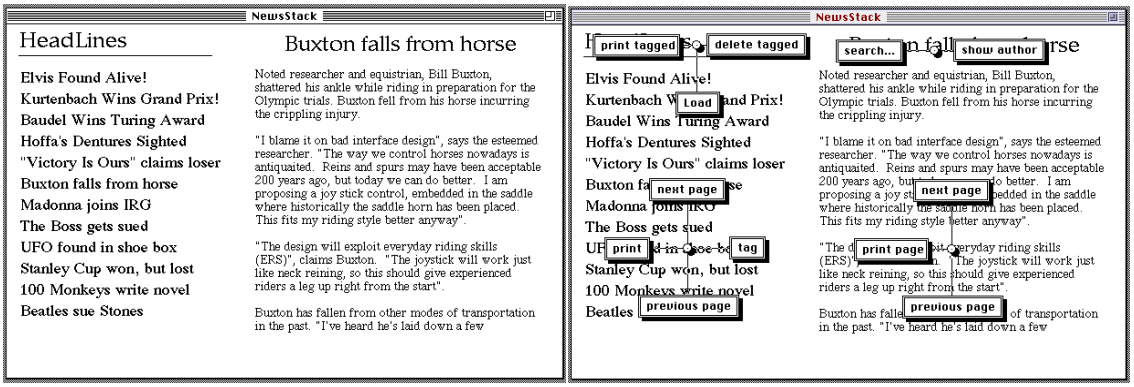


Figure 6.6: A simple news reader program in Hypercard that is controlled by marking menus. (a) shows the four major area of the screen: "Headlines", a list of articles, the title of the current article, and text of the article. (b) shows the marking menus associated with each of these areas. When marks are used to select from the menus the context (the location) of the mark contributes to its meaning.

Distinguishing tokens

Distinguishing marks by tokens involves augmenting a mark with some characteristic that disambiguates it. Augmentation can be of several forms. The shape of marks can be augmented. Alternatively, the dynamics of drawing the mark can be used to augment a mark.

Figure 6.7 shows how an augmenting "dot" at the start of a marking menu mark is used to indicate the mark is intended for a marking menu. An augmenting token, however, does not have to be at the start of the mark. The token could appear as a prefix to the mark, within the mark or as a suffix to the mark. However, if the mark is not distinguished from the start, then mark-confirmation may lead to ambiguities, since the system may identify the partially completed mark as both the start of a marking menu mark and an iconic mark.

Figure 6.7: Two marking menu marks that are augmented by a "dot" to distinguish them from other types of marks in an interface.

There are many alternatives to "dot". Any sort of token that guarantees distinction could be used. In practice, we found "dot" easy to draw and easily and reliably recognized by the system.[21] We also found that one could make an analogy between it and press-and-wait. In Tivoli, pressing-and-waiting in drawing mode popped up a marking menu to change pen settings. "Dot" could be thought of as a mark in command mode that mimicked press-and-wait, and allowed access to the pen setting menu.

Another way of distinguishing marks is by dynamics. For example, in some systems the speed at which a mark is drawn determines its meaning. For example, a slow up-stroke may mean "next page", while a quick up-stroke (a "flick-up") may mean "go to the end of the document" (Go, 1991). In Tivoli, we experimented with dynamic schemes and found several problems. First, flicks are not consistently recognized because the speed of a flick varied with direction and the user's dexterity. Also, quick movements sometimes caused the pen to skip off the display surface before the speed of a flick could be attained. Flicking was not very reliable because of these problems. We also experimented with prefix flicks and suffix flicks. Prefix flicks made drawing the remaining mark too hard: slowing the pen down after drawing the flick to draw the rest of the mark, was difficult. Alternatively, drawing the entire mark at flick speed was too hard. Suffix flicks were more reliable: we could safely draw the first part of the mark then add a "flick flourish" on the end of the mark to indicate it as a marking menu mark.

---

[21] We occasionally operated Tivoli with a mouse, although it is intended to be operated with a pen. In this case we found a "dot" very difficult to draw. Thus we would not recommend the use of the "dot" for mouse driven system that use markings.

Recognizing flicks was further complicated by limitations in the input event software. On occasion, input events are buffered. Time stamping of input events occurs after events are read from the input buffers and therefore, at times, these buffering delays confuse the flick recognition process. This problem could be overcome by immediately time stamping all events. Nevertheless, this indicates that tracking dynamics place special demands on input software.

Even if flicks could be made reliable they still present a problem: how can flicks be demonstrated to a user? The "dot" is easy to learn because a user can simply be told: "make a dot, about this big". Flicks on the other hand are dynamic in nature and are best learned by demonstration and practice. Section 6.3.2 discusses issues concerning self-revelation of mark dynamics.

To summarize, we have presented three strategies to avoid ambiguity between menu marks and iconic marks: avoidance, different context, and distinguishing tokens. Based on the various advantages and disadvantages each strategy just discussed, we elected to use a distinguishing tokens strategy in Tivoli. Specifically we used the "dot" prefix mark shown in Figure 6.7. Section 6.4 discusses our experiences with this strategy.

6.2.3. Dealing with screen limits

One problem that can occur in a pop-up menu system is that, when a menu is displayed near the edge of a screen, some portion of the menu may be clipped-off. This may make it impossible to see or select some items. We refer to this as the *screen limit* problem. Marking menus suffer from this problem because they use pop-up menus.

One possible solution to the screen limit problem is not to allow menus to be displayed too close to the edge of the screen. This implies placing menu "pop-up spots" some safe distance away from the edge of the screen. While this is a workable solution, it is not practical when menu hierarchies are deep, since pop-up spots may have to be located a large distance from the edge of the screen to keep the submenus from hitting the edge of the screen. Furthermore, it seems to be an unreasonable constraint given popular interface design. For example, most drawing programs have scrollable windows, and a user is allowed to scroll a window till menu pop-up spots are close to the edge of the screen.

Another solution to the screen limit problem is *constraining*., Most pop-up menu systems constrain menus to display entirely on the screen, even if the location from which the menu was invoked would cause some portion of it to be clipped-off. For example, the menus in Open Look use this solution (Hoeber, 1988). Constraining, however, causes problems when hierarchic menus are used. In this case, accessing a series of menus causes each menu to hit the edge of the screen. We refer to this problem as *crowding*. When crowding occurs, users end up making a series of selections from menus that are against the screen edge and this can sometimes make menu selection slow and error-prone.

Hopkins (1991) uses a constraining solution for radial menus. Since marking menus use radial menus, it is worthwhile to consider this solution. With Hopkins' radial menus (or pie menus), normally, a pie menu pops up centered around the cursor location. However, when the cursor is close to the edge of the screen, this results in some portion of the menu being clipped-off, To overcome this problem the menu is displayed not centered around the cursor, but shifted over so it is completely displayed. The cursor is then reset by the system to the center of the menu (this is referred to as "warping" the cursor). At this point, the user can make a selection in the usual way.

Problems occur with Hopkins' solution when the input device is an absolute device like the pen, and this makes it unsuitable for marking menus in Tivoli. The problem is that the system cannot change the location of pen (given the constraint that the cursor always appears under the tip of the pen). An example demonstrates this. Suppose a radial menu is popped up too close to the edge of the screen. If the menu is constrained to display completely on the screen, the pen tip is no longer in the center of the menu. The pen tip generally ends up located in one of the menu items. This immediately highlights the item. If the highlighted menu has a submenu, this menu would then be displayed. Thus, a user inadvertently descends the menu hierarchy. Even if the menu item has no submenu the user would still have to move the pen out of the menu item if the menu item was not the desired one.

We propose the following solution which permits marking menu selections near the edge of the screen when using a pen. When the pen is pressed close to the edge of the screen, the marking menu appears centered around the pen tip cursor with some portion of it clipped-off. If the clipped-off portion is large enough to obscure some menu items, another special menu item (referred to as the "pull-out" menu item)

appears on the screen (see Figure 6.8). At this point a user can select the visible menu items in the normal fashion. However, if the user moves the cursor to the pull-out menu item, the menu is redisplayed centered at the location of the pull-out item. The pull-out menu item is located far enough away from the edge of the screen so that the menu is completely visible when redisplayed. At this point the pen is located in the center of the menu and all items are accessible. This same scheme works with hierarchic menus. Every time a submenu hits the edge of the screen, a pull-out item is displayed.

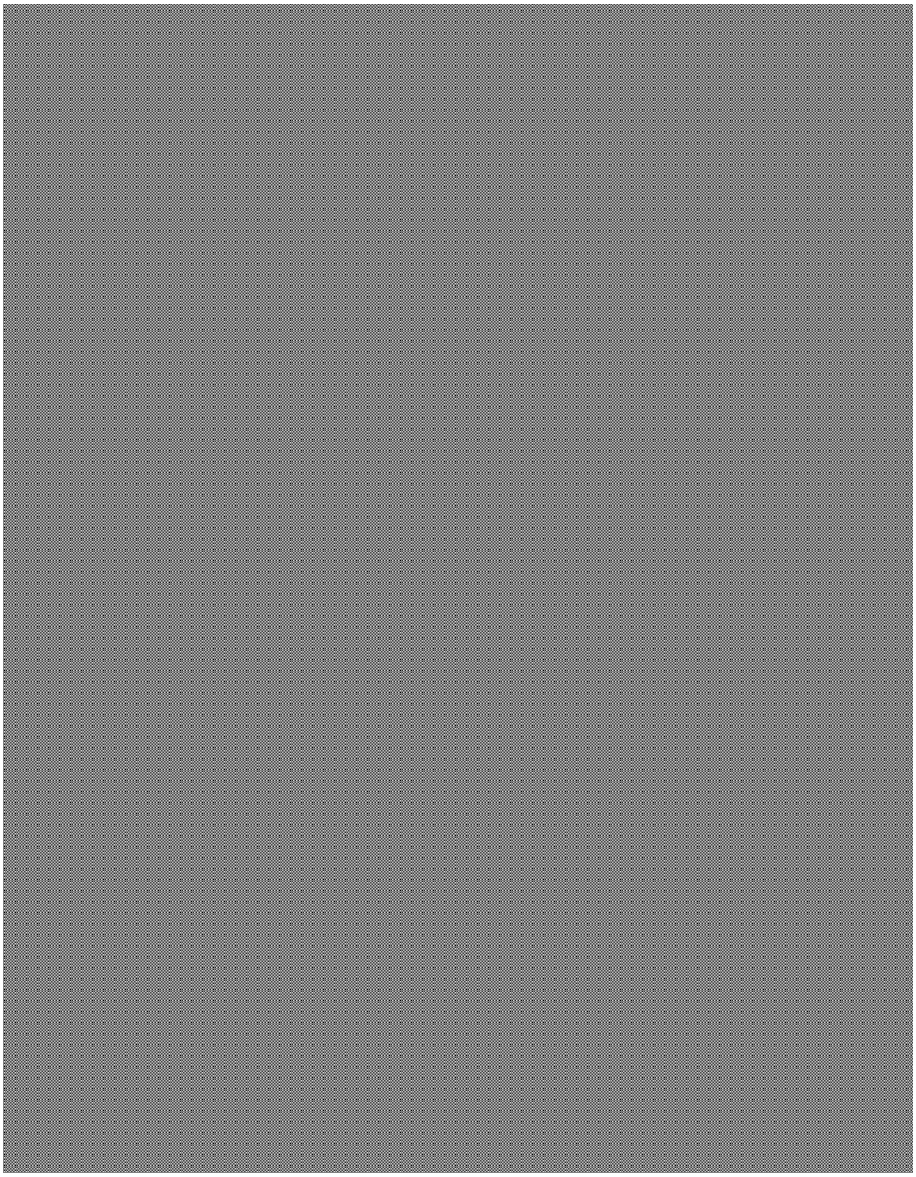


Figure 6.8: A "pull-out" menu item allows a user to access menu items that would be clipped-off by the edge of the screen. In (a) a user has displayed a marking menu but a portion of it is clipped-off by the edge of the screen. Because of this, a pull-out item appears (the gray circle). In (b) when the user drags over to the pull-out item, the menu is redisplayed so all items can be accessed.

Marks also have a screen limit problem. If one starts a mark too close to the edge of the screen one may run into the edge. As with menu mode, the input device used makes an important difference in a solution to the problem.

If a relative input device like the mouse is used, it is possible for users to draw marks "beyond" the edge of the screen. Hopkins (1991) has proposed a solution that

is suitable for marks. Hopkins' pie menus use a technique called mousing-ahead which is similar to marking but the path of the cursor leaves no ink-trail (see Section 2.3.1 for a complete explanation of mousing-ahead). Mousing-ahead is possible even when the cursor hits the edge of the screen. Although the cursor is constrained to the area of the screen, mouse movement after the cursor hits the edge of the screen is still tracked. Thus, a user can mouse-ahead beyond the edge of the screen. Applying this solution to marks, a user could draw marks beyond the edge of the screen, although some portion of the mark would not be visible. This solution is important because it preserves the principle of rehearsal. The movement to select from the menu must be the same as movement to make a mark and this happens even when menus and marks hit the edge of screen.

If the input device is a pen, drawing a mark close to the edge of the screen behaves logically: if the mark does not hit the edge of the screen, it can be performed as usual; if the mark does hit the edge of the screen, a user cannot physically draw it. This behavior mimics the way pen and paper works—if one is too close to the edge of the page one cannot draw certain marks.

We still need to, however, be able to apply marks to objects that are near the edge of screen. To do this we mimic pen and paper traditions. Generally, when something is too close to the edge the page to fit, a line is drawn from the object, out to some clear space and then an annotation is made. We propose a similar design. Suppose an object is too close to the edge of the screen for a certain mark to be made. A user can draw a line, out to some clear space on the screen, then make a "pull-out" mark, followed by the desired mark. Figure 6.9 shows this.

6.3. APPLYING THE PRINCIPLES TO ICONIC MARKINGS

Marking menus provide self-revelation, guidance, and rehearsal for "zig-zag" types of marks, specifically, the type of marks that are the byproducts of selecting from radial menus. Can a similar mechanism be provided for iconic marks? As a design experiment we decided to see if we could design mechanisms similar to marking menus but for the edit marks in Tivoli. Thus we attempted to design ways to self-reveal these marks, guide a user in making them, and have this be a rehearsal which builds skills for expert behavior.

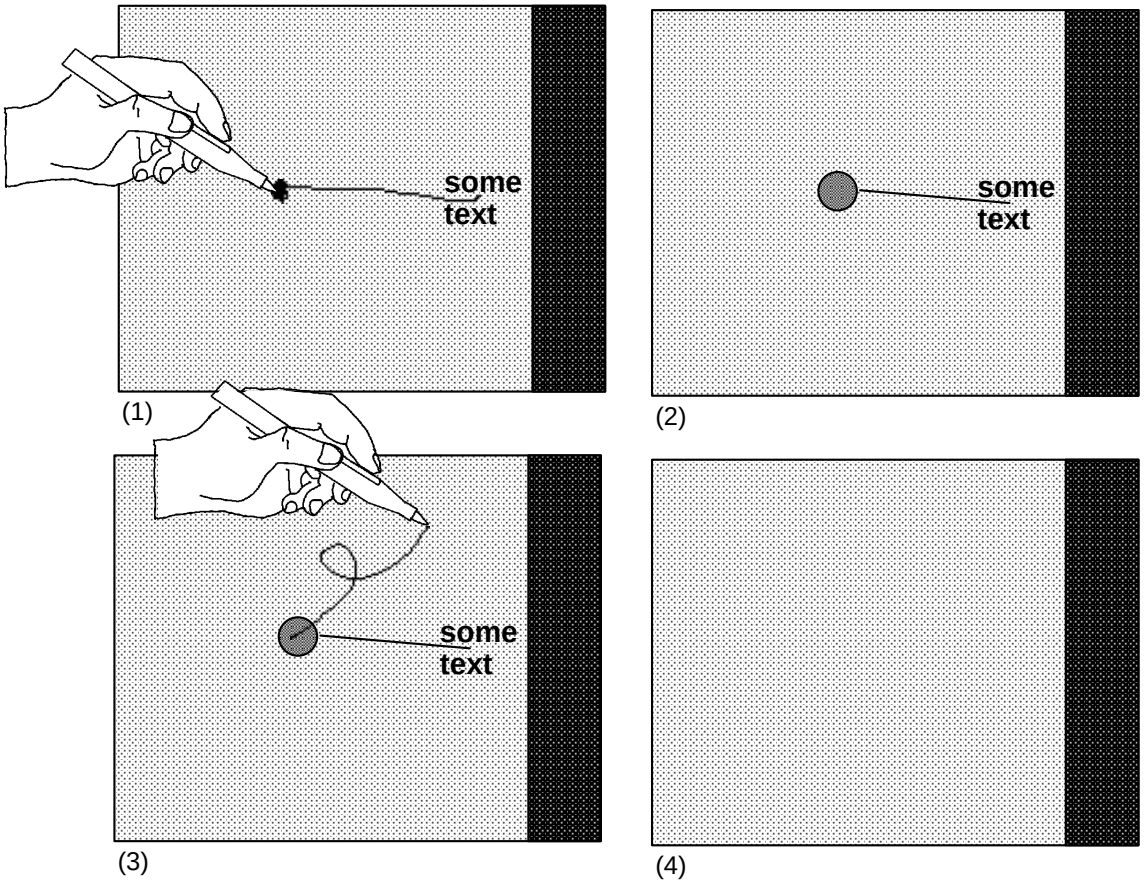


Figure 6.9: Using a "pull-out" mark to apply a mark to an object close to the screen edge. In (1), the pull-out mark is drawn (a line followed by a dot). In (2), the system has turned the mark into a pull-out object. A mark is then drawn in the pull-out object, in (3). In (4), the mark is applied to the object that is "pulled out", and it is deleted.

Another design goal was ease of programming. One of the attractions of marking menus is that an interface programmer can implement interactions which provide self-revelation, guidance, and rehearsal with something as simple as a pop-up menu subroutine call. We wanted a mechanism for iconic marks that was just as convenient to program. The idea was to avoid creating custom code to self-reveal each different type of mark.

6.3.1. Problems with the marking menu approach

Overlap

Suppose we strictly applied the marking menu design to the marks shown in Figure 6.3. In other words, display all the possible marks a user could make starting from a

certain location. Figure 6.10 shows the result of this approach. Marks overlap and can cause confusion. Part of the problem is that iconic marks are not suitable for displaying in this manner. Menu marks, however, are suitable because of their directional nature. Another problem in the example is that each entire mark is displayed. If all the marks of a hierarchical marking menu were displayed, this too, would result in overlap.

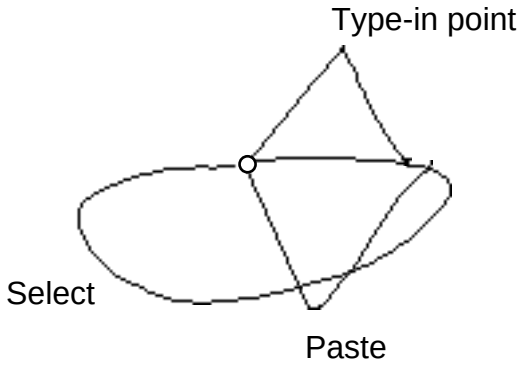


Figure 6.10: Overlap causes confusion when using the marking menu approach to self-reveal other types of marks. Here we display the commands available when starting a mark from a clear spot in the drawing region of Tivoli.

Not enough information

A display like Figure 6.10 gives little contextual information. For example, the important thing about the "Select" mark is that it should encircle objects and the shape of the circle can vary. This type of information is not shown in Figure 6.10.

The meaning of several edit marks in Tivoli is determined not only by the shape of the mark but also by the context in which the mark is made. For example, a straight line over a bullet-point moves an item in a bullet-point list, while a straight line in a margin scrolls the drawing area. These types of inconsistencies can potentially confuse the user. To avoid these problems, we wanted to provide context sensitive information about which edit marks a user can make over what objects. Informally, we wanted a user to be able to answer the question: "what marks can I draw on this object or location?". Since marking menus are sensitive to context (i.e., the contents of a menu may vary depending on where it is popped up), we hoped that some similar mechanism could be designed for iconic marks in Tivoli.

For mark sets in general, besides Tivoli's iconic mark set and the marking menu mark set, the following characteristics may contribute to a mark's meaning and this type of information therefore needs to be self-revealed.

Shape: This is the case where a particular shape is an icon for a certain command. For example the "pigtail" shape is an icon for the delete command.

Direction: Sometimes the direction of a mark affects its meaning. For example a up-stroke means "scroll up" while a down-stroke means "scroll down". The shape of the mark is basically the same but the direction or orientation of the mark has meaning.

Location of features: The location of particular features of a mark can affect its meaning. For example, the summit of the "Type-in" point mark shown in Figure 6.10, determines the exact placement of the text cursor.

Dynamics of drawing: How a mark is drawn can affect its meaning. For example, a flick could mean "scroll to the end of document", while a slow up-stroke could mean "scroll to the next page".

6.3.2. Solutions

Crib-sheets

Interactive crib-sheets self-reveal marks without the overlap problem. When the user requires help, a crib-sheet can be popped up which shows the available marks and what they mean. The user can then dismiss the crib-sheet (or "pin" it down on the side) and make a mark. In Chapter 1, two systems that use mechanisms similar to this were described. Crib-sheets can be as succinct as a simple list of named marks or as elaborate as multi-page explanations of the marks in great detail. Thus a crib-sheet could contain complete information on all the characteristics of a mark. However, since crib-sheets are for reminding and guidance, they are usually succinct.

Figure 6.11 shows the crib-sheet technique we designed for Tivoli. The design works as follows. Similar to a marking menu, if one doesn't know what marks can be applied to a certain object or location on the screen, one presses-and-waits over the object for more information, rather than marking. At this point, rather than a menu popping up as in the marking menu case, a crib-sheet is displayed. The crib-

sheet displays the names of the functions that are applicable to the object or location, and example marks. If this is enough information, a user can draw one of the marks in the crib-sheet (or take any other action) the crib-sheet automatically disappears. If the pen is released without drawing mark, the crib-sheet remains displayed until the next occurrence of a pen press followed by a pen release or a press-and-wait event.

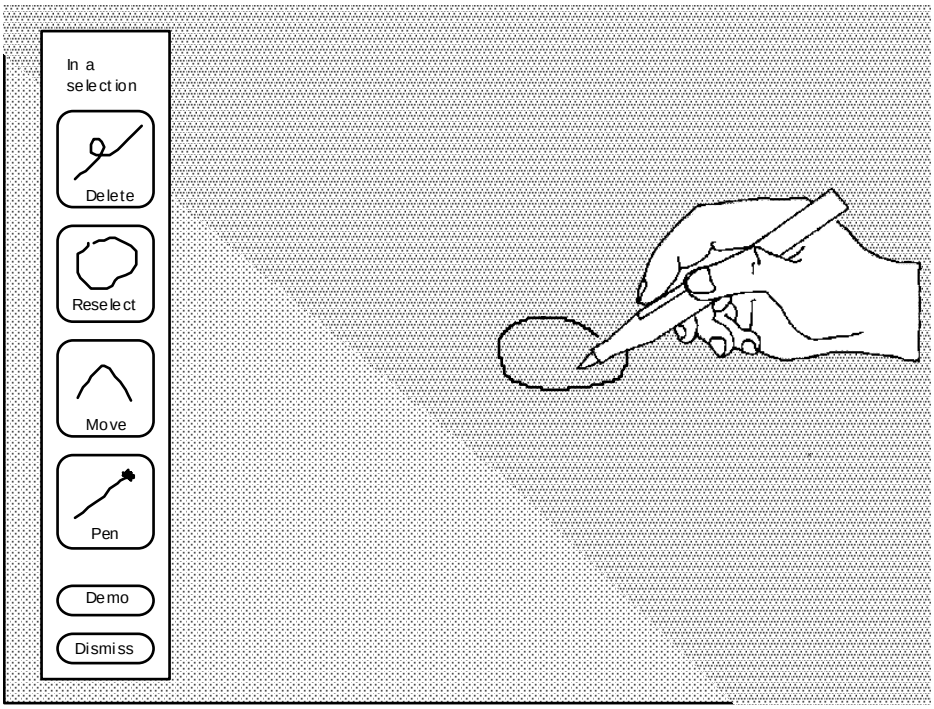


Figure 6.11: Self-revealing iconic marks in Tivoli: The user has selected the word "Tea" by circling it. To reveal what functions can be applied to the selection, the user presses-and-waits within the selection loop. A crib-sheet pops up indicating the context ("In a selection") and the available functions and their associated marks.

This design has several important features. First, the system displays the crib-sheet some distance away from the pen tip so that the crib-sheet does not occlude the context. This leaves room for a user to draw a mark. Second, the significance of the location of the pen tip is displayed at the top of the crib-sheet (i.e., in Figure 6.11 "In a selection" is displayed at the top of the crib-sheet). This is useful for revealing the meaning of different locations and objects on the screen.

This design obeys the principles of self-revelation, guidance, and rehearsal. The crib-sheet provides self-revelation, and a user can use the examples as guidance when drawing a mark. Rehearsal is enforced because a user must draw a mark to

invoke a command. For example, a user cannot press the delete button on the crib-sheet to perform a deletion. The user must draw a delete mark to perform a deletion.

Animated, annotated demonstrations

While the crib-sheet does self-reveal contextual information about marks, it still lacks certain types of information. For example, one static example of a mark relays little information about variations and features of a mark. It has been shown that people need good examples to help visualize procedures (Lieberman, 1987). Ideally a demonstration of the mark in context should be provided, similar to what one receives when an expert user demonstrates a command. The tutorial program in Windows for Pen Computing works like this. In the tutorial, a user is shown how marks are made by animated examples.

Demonstrations can be provided through animation. Baecker and Small have described how animation can assist a user, and how the animation of icons can be effective (Baecker & Small, 1990; Baecker, Small, & Mander, 1991). The idea of animated help is not new. Cullingford (1982) used "precanned" graphical animation coupled to natural language contextual messages to provide help. Feiner (1985) used graphical explanations to illustrate the problem solving process of real world physical actions. Feiner's system, however, was not sensitive to the user's current context. A research system, called *Cartoonist*, which automatically generates context sensitive animated help for direct manipulation interfaces, has been developed (Sukaviriya & Foley, 1990; Sukaviriya, 1988). The major difference between Cartoonist and the system we are about to describe is that Cartoonist is designed for direct manipulation interfaces, not mark-based interfaces. As we shall see, an animation of drawing a mark must have special features to make it meaningful and helpful. Specifically, in our system, the animation of a mark is annotated with text for explanation. Cartoonist does not support annotations. Furthermore, Cartoonist relies on an extensive knowledge base to describe the application and interface. The system we describe has a vastly simpler implementation which is compatible with existing user interface architectures.

Crib-sheets could be animated. A crib-sheet could show how to draw a mark, variations on a mark, and the various features of a mark. This certainly would help a user understand how a mark should be drawn. However, crib-sheets illustrate

marks outside of the context of the material that the user is working on, and this can make it difficult to see how the mark applies to the context. Marking menus, on the other hand, have the advantage of showing the available marks directly on top the object being worked on,.

To solve these problems we extended the function of the crib-sheet by adding animations of marks which take place in context. If the crib-sheet does not provide sufficient information, a demonstration of a mark can be triggered by pressing the "demo" button on the crib-sheet. The demonstration of the mark begins at the location originally pressed. The demonstration is an animation of the drawing of the mark which is accompanied by text describing the special features of the mark (see Figure 6.12).

There are several important aspects to this design:

• Marks are shown in context. The animation of the mark is full size, and emanates from the exact location originally pressed on by the user. A user can trace the animated mark to invoke the command.

• Variations in marks can be demonstrated by multiple animations. There is usually a variety of ways to draw mark. For example, a pigtail, signifying deletion, may be drawn in any direction, clockwise or counterclockwise, big or little. To prevent users taking a single animated example too literally, we show variations by animating multiple examples of mark. Usually, two examples seems to be enough.

• Information about features is provided by annotations. Not only is the drawing of a mark animated but the animation is annotated with text to explain features or semantics of marks (e.g., in Figure 6.12 "A pigtail deletes the *selected objects*."). In addition, features of the application can be displayed. For example, in Tivoli scrolling marks can only be drawn in the margins of the drawing area, but the borders of margins are not visible.[22] In situations like this, the animation can display these features to clarify matters. Annotations appear in sequence during the

[22] This was done to keep the drawing area uncluttered.

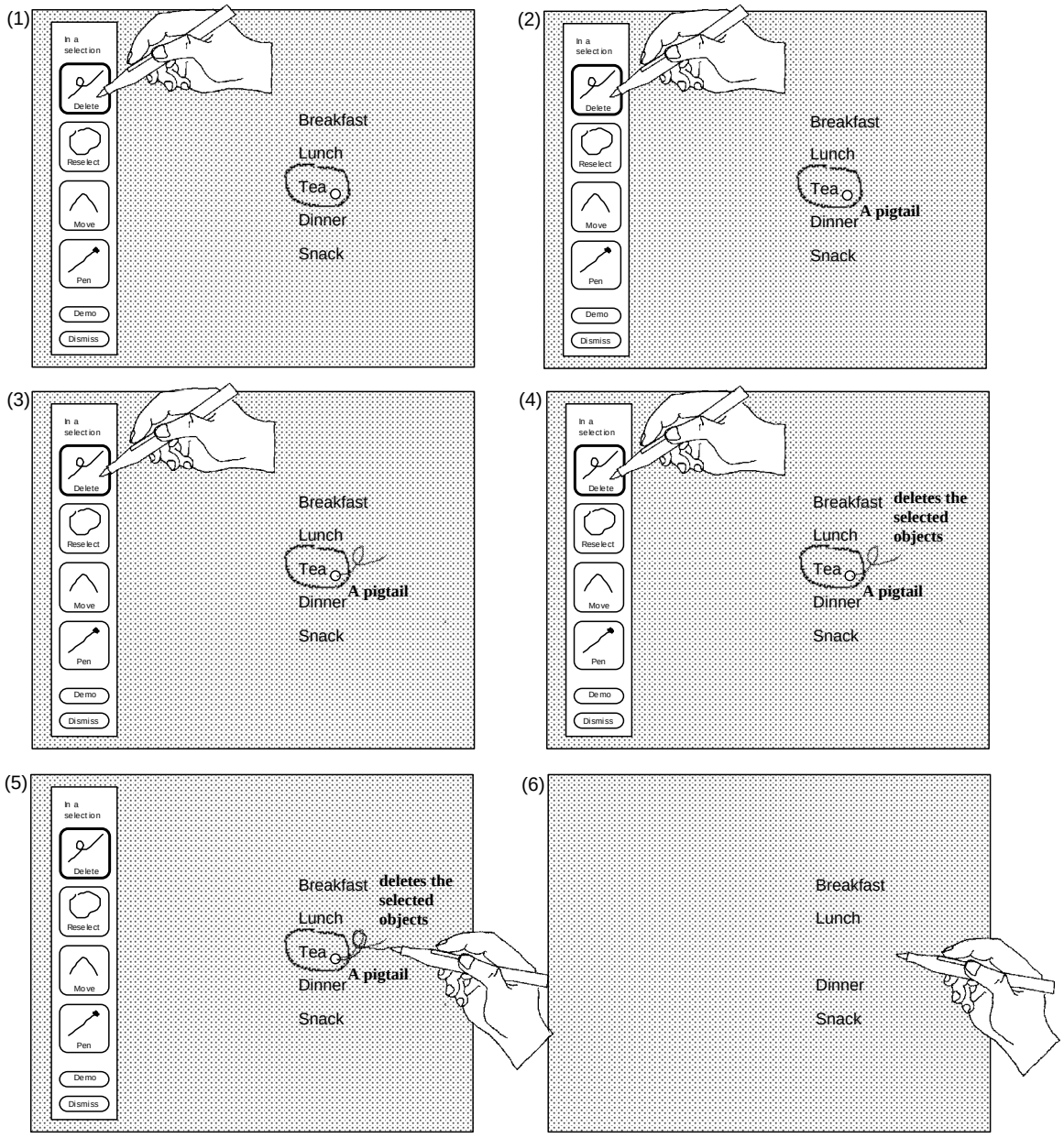


Figure 6.12: A demonstration of a particular function can be attained by pressing its icon. In (1) the user presses on the delete icon for more information. This triggers an animated demonstration of the mark with text annotation to explain its features. This is shown in (2), (3) and (4). In (5), the user traces along the example mark to invoke the function. When the pen is lifted, the action for the mark is carried out, and the crib-sheet and animation disappear (shown in (6)).

mark's animation, and they are timed to remain on the display long enough for the user to be able to read them.

• Animation can be controlled. A long series of animations takes quite a bit of time and this can be tedious for the user. By pressing a button in the crib-sheet, individual animations of the marks can be started or stopped. Pressing "Demo All" causes the system to cycle through all the animations. Pressing the "Dismiss" button stops the animation and removes the crib-sheet. As in the case of the crib-sheet by itself, the moment a user completes a mark, the crib-sheet is removed and the animation terminates.[23]

• The user is not required to make a mark from the crib-sheet. The user is free to perform any mark at any location on the screen while the animation is running. As before, the moment the user completes a mark, the animation and crib-sheet are removed. The user can also choose to not draw a mark by tapping the pen against the screen. This also removes the animation and crib-sheet.

Architecture

A goal for our crib-sheet/animation design was that it be easy for an interface programmer to use. We designed the software architecture with this in mind. To describe the characteristics of this architecture, we will describe an interactive computer system as consisting of two parts, an application module and animator module. The application allows the user to interact with a particular domain of materials by means of marks (i.e., Tivoli is the application and the materials are free-hand drawings). The animator is called by the application to show the marks to the user. The animator is generic—it can be made to work with different applications.

The design of the animator raises many specific design problems. We describe the animator by laying out the problems and describing how they are addressed.

How does the animator get invoked? This is the job of the application. As with a marking menu, the user deliberately presses-and-waits while the command button pressed. The application detects this action and then calls the animator.

---

[23] The animation actually freezes when a user begins drawing a marking so a user can trace the animated mark. The animation is removed from the screen when the user finishes drawing the mark and raises the pen.

How does the animator know which marks to animate? In order to make an application work with the animator, an application-specific Mark Animation Database (MAD) must exist. The MAD contains descriptions of examples of marks grouped by application context. When the user presses-and-waits, the application calls the animator with a description of the current context. The animator can then select the marks to be animated based on context.

How are marks and contextual features animated? In order to understand how marks are animated it is convenient to first understand the structure of MAD. Figure 6.13 shows an example of the structure of MAD. MAD consists of annotated examples of marks which are grouped by context. When the application calls the animator with a context, the examples corresponding to the context are retrieved from MAD. When a user requests a demonstration, the animator animates these examples. A mark is a sequence of x and y coordinates which is animated by incrementally displaying the mark. The marks that appear in MAD were originally drawn by hand. When animating a mark the animator uses the same drawing dynamics as the original hand-drawing (a technique developed by Baecker (1969)). In this way, dynamics of drawing can be revealed and the speed of an animation can be controlled by the constructor of the database. Annotations are labeled by where and when they should occur in the animation cycle (e.g., "start" and "end"). The pacing of the animation of text annotations is determined by length of text: after an annotation is displayed the animator pauses for an amount of time that is proportional to the length of the text before continuing with the rest of the animation. This gives a user time to read the annotation and then watch the rest of the animation.

How are variations shown? Variations are shown by animating another example of a mark. A mark in MAD can have more than one example. If an extra example is tagged as "variation", it is then included in the animation along with the original example.

How is the crib-sheet constructed? When the animator retrieves the examples from MAD, labels for the crib-sheet buttons are extracted, and example marks are shrunk down to be displayed in the buttons. We found it convenient to designate certain example marks for shrinking. Therefore, a function in MAD can contain an extra

example mark that is tagged for use as an "icon" in the crib-sheet. If no "icon" example is found, the animator shrinks the first example mark it finds.

How are application features animated? Like text annotations, application features appear in MAD. If during an animation an application feature needs to be displayed, the animator makes a call-back to the application. For example, the call-back may ask the application to display the margin boundaries of the drawing area. Therefore, a call-back protocol must exist between the application and animator.

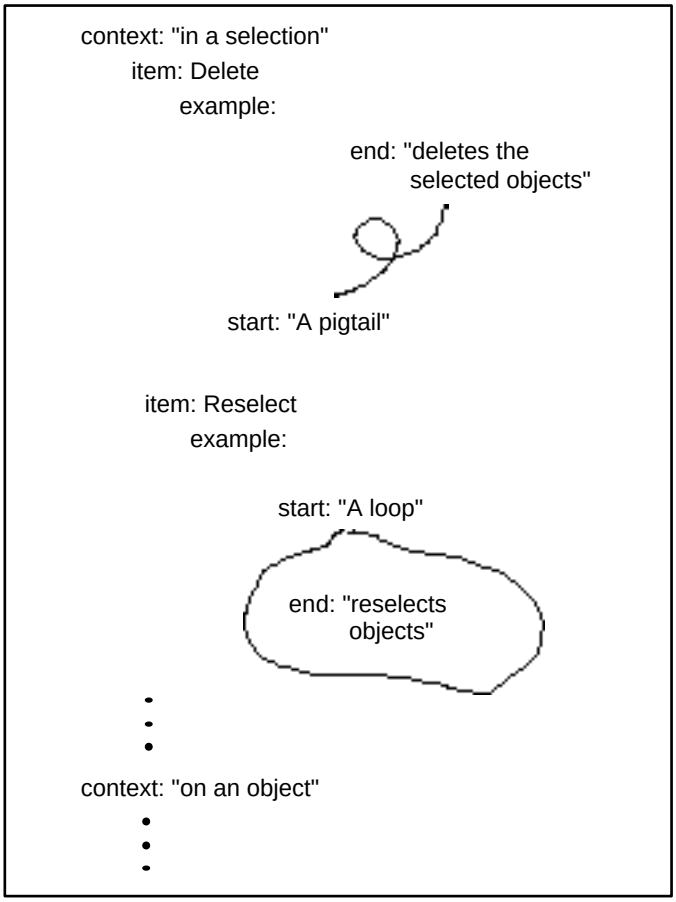


Figure 6.13: An example of the structure of the Mark Animation Database (MAD). Annotated examples of marks used for the crib-sheet and animations are grouped by context and function.

How are marks animated in constrained spaces? Assume that a user invokes the animator near the bottom of the drawing area, and that one of the possible marks at that point is a pigtail. At the bottom of the drawing area, there is no room to draw a

pigtail downwards, but there is room to draw it upwards. Thus, the animator should show only pigtails that fit in this place. The solution to this problem lies in the fact that MAD contains multiple examples of marks. When the animator retrieves examples from MAD it looks for examples that will fit in the space it is working with. Thus, MAD should be set up with many examples of each mark, so that the animator can find an example for any location. We found as little as four different examples were sufficient. In the event that an example which fits cannot be found, the animator generates and displays a "no room message (e.g., "not enough room to demo pigtail here"). This tends to only happen when there is not enough room for a user to actually draw the mark.

How is MAD constructed? MAD is constructed by drawing the examples in the form shown in Figure 6.13 and then copying these examples into MAD. For Tivoli, we constructed the examples by drawing them in Tivoli. Thus we could easily design examples that fit in constrained spaces in Tivoli by drawing them in those spaces. For example, we drew instances of pigtails that fit at the top, bottom, left and right edge of the screen. The animator does not have to be sophisticated at laying out the animations—the layouts are determined by the constructor of the examples. The animator need only check if an example will fit at a certain location. If it does not fit, it merely looks for another example.

More sophisticated features

The design for the crib-sheet/animator and MAD previously described has been implemented. Section 6.4 describes experiences using it. We now discuss future designs which are currently not implemented.

One problem with our current implementation is that, although animations do appear in context, they do not "work with" the context. For example, the animation of a loop being drawn to select objects sometimes doesn't enclose any objects. The problem is the animator has no knowledge about the application objects underlying the animation.

A more advanced version that we have not implemented extends the notion of parameterized marks to allow them to utilize application objects in the current working context. For example, assume we have a mark to move a list item. There would be two typed parameters to this mark: the list item and the location to which it is moved. In Tivoli, the list item would be a set of strokes between two "blue

lines" (like the blue lines on lined paper), and the location would be a blue line between two other list items. When the application calls the animator and tells it to animate a move-list-item mark, it would have to also give the animator some actual items and locations in the current context. The animator would then deform a move-list-item mark to fit the items and locations. Thus, the user would see a real example in the current context.

Having examples that manipulate the objects in the current context requires a much more sophisticated architecture for the animator. The animator must be able to manipulate objects in the application interface, and therefore a protocol that allows this must exist. Essentially, the distinction between the application and the animator becomes blurred in this more sophisticated scheme: the animator needs to know how to manipulate the application in the same way a user does. It must be able to identify objects and locations, construct marks and apply those marks. In addition, it needs to annotate the examples in a meaningful way. All these features require that examples in MAD be parameterizable. The design of this architecture is future research. A good starting point is to build on the work that Sukaviriya and Foley have done on the generation of parameterizable, context sensitive animated help for direct manipulation interfaces (Sukaviriya & Foley, 1990; Sukaviriya, 1988).

Integrating menu marks

As described earlier, menu marks in Tivoli are treated in the same manner as iconic marks. Specifically, menu marks will be interpreted as commands if drawn in command mode (i.e., drawn with the command button pressed down). The crib-sheet/animator provides self-revelation for all marks available in this mode including menu marks.

It would be impractical to include in the crib-sheet and animations all the marks used to access the pen settings menu. The menu is a much better mechanism for revealing this information, but is available only in drawing mode. Therefore, the crib-sheet/animator refers the user to the marking menu that is available in drawing mode. The animation of this is shown in Figure 6.14. The animation shows how to draw the dot required for a menu mark to be distinguished from other marks, and shows one example pen setting. The animation then displays a message for the user to see the marking menu available in drawing mode for more information. In this way an information link exists between the crib-sheet/animator and the marking

menu. Hence the crib-sheet provides self-revelation for the menu marks by referring the user to the marking menu.

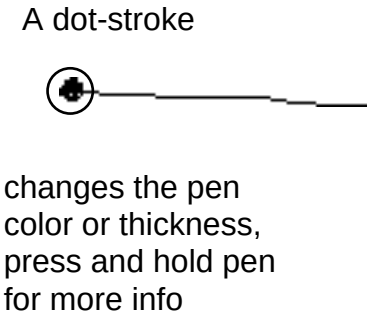


Figure 6.14: To self-reveal menu marks, the animator shows one example then refers the user to pop up the marking menu itself for more information. This avoids the problem of explaining and animating the many marks used for the marking menu.

6.4. USAGE EXPERIENCES

A large portion of the design described in this chapter has been implemented. The crib-sheet, animator, and MAD have been implemented, although the parameterized version of the animator was not implemented. Tivoli currently supports animations with multiple examples for every mark it uses. As Tivoli evolves, we expect the mark set to change. This can be supported by simply modifying MAD. The pen setting menu and marks were completely implemented. The "pull-out" menu item has yet to be implemented.

Future research will include formal user tests of our designs. It would be optimistic of us not to expect users to have problems with our system. First, there are many details that user might trip over: are the menus and buttons labeled meaningfully? Are the press-and-wait time thresholds correct? We believe the next step in user testing is to evaluate some of these details and refine the content of the animations. As Baecker, Small, & Mander (1991) point out, animations require significant development and refinement. Fortunately, our design makes this easier than a frame by frame process.

The design has been used informally by several researchers at Xerox PARC. Users appeared to be quite successful at using the marking menu, once press-and-wait was

understood. Users were also successful at selection using a mark but found recognition unreliable. We traced this unreliability to incorrectly drawn "dots" at the start of marks. We found the source was not that a user failed to draw the "dot", but that the system occasionally did not start tracking the pen till after the "dot" was drawn. This implies that the pen tracking hardware and software needed improvement.

Another problem revealed through informal use was the "right-handedness" of the marking menu. Depending on a user's handedness, some portion of the screen is occluded from view when one's arm is holding the pen against the screen. When using the marking menu, left handed users found some menu items occluded from view (they had to look "under" their arms). This implied that, like most pen-based systems, marking menus must be configurable for handedness.

Users also experimented with using the crib-sheet/animator. Initially, we found that users did not notice the crib-sheet pop up on the left side of the display. This was because users were so close to the large display that the crib-sheet popped up outside their visual focus. We then added an animation of the crib-sheet expanding from the point at which press-and-wait occurred. This helped users notice the display of the crib-sheet.

Users were also able to make use of the crib-sheet/animator after a brief demo. We found that users explored the interface by pressing-and-waiting at different spots to see what functions where available. We also observed users tracing the animated marks. The most common error involved a user pressing-and-waiting with the command button pressed, then releasing the button while watching the animation. The user would then trace the animated mark without the command button being pressed. This would result in the mark being drawn but not interpreted (i.e., the mark as drawn in drawing mode, not in command mode). We feel this type of error may disappear when a user gets into the habit of holding down the command button to issue a command. It is also possible to have the system recognize this error and advise the user to press the command button.

6.5. SUMMARY

In the beginning of this chapter we set out to integrate hierarchical marking menus into a pen-based application, and provide self-revelation, guidance, and rehearsal

for iconic marks. A design was developed and implemented to satisfy these goals. The design gives rise to many issues and conclusions:

The integration of marking menus into Tivoli reflects the situation with many applications today. Tivoli had an interface prior to our design experiment. Thus we were faced with the task of integrating marking menus with other interaction techniques. The main effect of this was that our design of marking menus had to change, not the existing interface components. This was an excellent test of the resiliency of the marking menus paradigm.

Marking menus had to be integrated with a range of interaction techniques. The interface to Tivoli not only contains edit marks but also free-hand drawing, buttons, dialog boxes, pop-up menus, mode buttons and a windowing system. Thus it was a challenge to find a spot where marking menus could fit in and be effective. The exploration also reminded us that interaction techniques cannot be added to an interface design without considering the other interaction techniques that surround it.

There were many other situations where we could have experimented with marking menus. One goal in redesigning the Tivoli interface was to reduce the number of buttons on the screen. Consolidating many buttons into a marking menu, hence, removing them from the screen, would have accomplished this. Also, using marking menus to issue commands to Tivoli objects such as list-items would have been another effective use. Time constrained us to only explore one particular usage. We thought using a marking menu to control pen settings would elucidate many design issues, since the menu marks would have to be used in the same mode as the edit marks. Nevertheless, this simple implementation gave rise to many design issues which one would encounter in a larger scale integration.

This design exploration also revealed issues concerning using marking menus in mark-based interfaces. Figure 6.15 summarizes the major design problems and the solutions we developed. Specifically, ambiguities can develop between menu marks and iconic marks. We proposed three solutions: avoidance, different context, and distinguishing token. We elected to used a distinguishing token strategy, given the way marking menus were being used in Tivoli. The other strategies, however, can be useful in other situations. Also this design exploration allowed us to use marking

Problem	Solution
Ambiguity between iconic and menu marks.	Draw a distinguishing token (a "dot") at the start of an menu mark.
Menu items clipped-off near edge of screen.	Use pull-out menu item.
Object too close to edge of screen to mark.	Use pull-out mark.
Need self-revelation for iconic marks.	Use crib-sheet/animator.
Provide guidance for iconic marks.	Draw a mark based on crib-sheet example or... Trace a mark over an example displayed by the animator.
Ensure rehearsal of iconic marks.	A mark is the only way to issue a command.
Crib-sheet/animator should be easy to program and work at any screen location.	The programmer generates multiple examples in MAD.
Getting information on marking menus marks from the crib-sheet/animator.	A crib-sheet/animator item refers user to the marking menu.

Figure 6.15: Major design problems encountered integrating marking menus into Tivoli and the solutions developed.

menus with a pen. This uncovered issues and led to developments concerning screen limits and drawing dynamics.

The crib-sheet/animator is designed to support the principles of self-revelation, guidance and rehearsal. These mechanisms do not appear and behave exactly like marking menus, and we have shown why this must be so, but we feel that the design supplies the same type of information to the user and promotes the same type of behavior.

Designing a mechanism to self-reveal iconic marks brings to light many issues concerning the self-revelation of marks. First, revelation can occur at various levels of detail. The crib-sheet is the first level: a quick glance at the icon for the mark may be sufficient for the user. An animation is the second level: it requires more time

but provides more information and explanation. Our design essentially supports a hierarchy of information where there is a time versus amount of information tradeoff.

A hierarchic view of information can also be applied to the way in which marks themselves are self-revealed. For some marks, it is sufficient just to show a static picture of the mark. For other marks an annotated animation is needed before each one can be understood. Besides an animation, some marks need to show variations. Finally some marks, like menu marks, are best self-revealed incrementally. Depending on the characteristics of a mark, there are different ways of explaining the mark. This implies our self-revelation schemes must support these different forms of explanation. Marking menus, crib-sheets, and animations are instances of different forms of explanation. A complete taxonomy of forms of explanation is future research.

While user testing is needed to refine our design, we feel that this design supports the desired type of information flow. Users can interactively obtain information on marks and this information is intended to interactively teach them how to use these marks like an expert. No mark-based system that we know of supports this type of paradigm.

Chapter 7: Conclusions

7.1. SUMMARY

This dissertation develops and evaluates an interaction technique called marking menus. Marking menus were developed based on several observations:

1) Marks can be an efficient and expressive way to issue commands, especially for pen-based computers.

2) Marks, by themselves, are not easy to use because unlike buttons, menus, and icons, they do not automatically reveal themselves to a user.

3) Therefore, marks must rely on some other interaction technique to reveal themselves to the user.

Given these observations we designed an interaction technique that combines menus and marks with the intention that using the menu helps a user learn the marks. The design of marking menus was based the design principles of self-revelation, guidance, and rehearsal. The principle of self-revelation states the system should interactively provide information about what commands are available and how to invoke those commands. The principle of guidance states that the way in which this information is provided should guide a user through invoking a command. The principle of rehearsal states that the guidance provided should be a rehearsal of act of drawing the mark associated with a command. The goal of these design principles is to help a user learn and use marks and quickly move from novice to expert.

After proposing a design for the technique based on these principles, we then evaluated the technique. The intention of the evaluation was to determine the limitations of the technique.

The first evaluation was an empirical experiment on non-hierarchic (i.e., one level) marking menus. This experiment showed that certain configurations of menu items make marking faster and less error-prone. Specifically, the experiment showed that four, eight, and twelve item menus enhance performance when marking. Also this experiment showed that subjects, on average, performed marks faster and more accurately with a mouse and stylus/tablet than with a trackball.

The second evaluation was a practical case study of two users' behaviors using a six-item marking menu for a real-life editing task. From this study we observed several things. First, with practice, users learn to use the marks and tend towards using the marks 100% of the time. Second, users utilized the features of the technique that were designed to aid in learning the marks (i.e., reselection and mark-confirmation). Third, using a mark in this situation was on average 3.5 times faster than selection using the menu.

A third evaluation was an empirical experiment examining the effect of menu breadth and depth on users' performance when selecting from hierarchic marking menus using marks. We found as breadth and depth of a menu structure increases, subject performance slows and the number of incorrect selections increases. Error rate appears to the limiting factor when selecting using marks. The experiment examined menus of breadth four, eight, and twelve, and menu depths from one to four. A significant change in error rate occurred when menu depth was greater than two and breadth was eight or twelve. The results suggest that marks can be used to reliably select from four-item menus up to four levels deep, or from eight-item menus up to two levels deep. This experiment also examined the effect of using a pen or a mouse. We found that subjects, on average, performed better with the pen than with the mouse. However, the difference in performance was not large. This indicated that the mouse would be an acceptable input device for hierarchic marking menus.

A final design study examined generalizing the design concepts of marking menus. Marking menus are an interaction technique that provides self-revelation, guidance, and rehearsal for a particular class of marks (i.e., straight lines and zig-zag marks).

We developed an interaction technique that provides self-revelation, guidance, and rehearsal for more general classes of marks. We also showed why the technique must differ from marking menus, and described an efficient means of implementing the technique.

7.2. CONTRIBUTIONS

The contributions of this work can be divided into two categories: contributions concerning marking menus specifically, and contributions concerning larger issues of human computer interaction.

7.2.1. Marking menus

The design of marking menus is a contribution in itself because of several design features. These features were described in detail in Section 2.2. The following is a summary of the design features that make marking menus a valuable and unique interaction technique. Marking menus:

- Allow menu selection acceleration without a keyboard.
- Permit acceleration on all menu items.
- Minimize the difference between the menu selection and accelerated selection.
- Permit pointing and menu selection acceleration with the same input device.
- Utilize marks that are easy and fast to draw.
- Use a spatial method for learning and remembering the association between menu items and marks.
- Are implementable as a "plug-in" software module.

The empirical studies and case studies in this work have contributed in:

Proving that users behave with marking menus as predicted. The design of marking menus features three modes of interaction: menu mode, mark confirmation mode, and mark mode. The case study in Chapter 4 has shown that users utilize all

three modes in the transition from novices, who use menus, to experts, who use marks. The case study also showed that users performed marks as quickly as keypresses. An equivalent interaction implemented with accelerator keys would have required pointing with the mouse *and* pressing an accelerator key. Hence we can conjecture that interaction was faster with marks than with accelerator keys in this setting.

Increasing our understanding of the limitations of marking menus. There is a limit to how accurately one can select items from a marking menu using a mark. The experiment in Chapter 5 has determined that selection using marks from menus with more than eight items per level and more than two levels of hierarchy will be error-prone. However, if two levels of eight item menus are used, marks can be used to quickly select from 64 menu items.

Determining configurations of marking menus that produce the best performance. Certain configurations of menu items make marking faster and less error-prone than other configurations. Specifically, our experiments have shown that 4, 6, 8 and 12 item menus and on-axis items enhance performance.

Demonstrating how command item selection and command parameters can be combined. Our case study demonstrates how both the starting point and end point of a mark can be used to express command parameters. This results in efficient interactions.

7.2.2. Issues of human computer interaction.

This work has several contributions to the study of human computer interaction in that it:

Identifies the fact that markings are not self-revealing. In the past, it has been assumed that mark-based interfaces will be easy to use because marks will be "natural" or mnemonic. This may be true in a some situations but not in all cases. There is a danger of falling into the trap that a system will be easy to use because it uses marks. This research makes the important point that while marks can be a very efficient means of interaction, this efficiency cannot be obtained if the user does not first have knowledge about the mark set. In some situations our experience with everyday pen and paper conventions supplies this knowledge. In other situations it

does not, and a self-revealing mechanism must be provided in conjunction with the marks.

Develops interaction techniques for self-revealing markings. Marking menus are a solution to the self-revealing problem for one particular class of mark. The crib-sheet/animator is a solution for more general classes of marks.

Identifies and develops the design principles of self-revelation, guidance and rehearsal. To solve the problem of marks not being self-revealing, this research develops the design principles of self-revelation, guidance, and rehearsal. Marking menus serves as an example of the application of the design principles and the crib-sheet/animator demonstrates that the principles can be applied to other situations. We feel that these design principles are valuable for interface design in general.

Develops a unique way to support novice/expert differences. The notions of guidance and rehearsal are a unique way of supporting novice/expert differences and transitions in mark-based interfaces. We know of no other systems that use a similar scheme.

Other research has dealt with novice/expert differences by providing explicit novice/expert modes. In these types of systems, novice mode has fewer functions than expert mode. The focus of this research is on supporting novice/expert differences and transitions using mark-based interfaces at the level of interaction, not at the level of available functions. These two approaches differ but they are not mutually exclusive.

Demystifies "the folk legend of gesture" in human computer interaction. It is clear from the literature that the types of gestures performed while operating an interface contribute to the overall sense of satisfaction with an interface. While others have observed that careful design of the body language of interactions results in better interface design, the research here is an explicit attempt to make use of this philosophy in a practical interaction technique.

Identifies the real value of marks as an interaction technique. Finally this research demonstrates that if the real advantages of particular interactions are understood, simple technology, used appropriately, can exploit these advantages. It is not simply the case that marks are desirable because marks are easy to remember. Another desirable property is the ability of a mark to efficiently express a command

and its parameters. The marks created by marking menus demonstrate this property. Furthermore, the technology required to support this property is not overly complex. Recognition methods, and ways of embedding and recognizing command parameters, are easily programmable.

7.3. FUTURE RESEARCH

As we developed marking menus we came across many interesting design variations, extensions and applications worth exploring:

• Adapt marking menus to be used on very small screens. A problem with very small screen computers is that there isn't enough room to draw long marks or display hierarchic menus. A variation on our marking menu design is to use a series of short strokes, all starting from the same location to perform a selection from a hierarchy of menus.

• Investigate other types of combinations of marks and menus. Continuous menu items, and dartboard and donut layouts, which were mentioned in Chapter 1, are examples of other types of combinations of marks and menus.

• Investigate feedback and pairing with command parameters. This research has only scratched the surface of things that can be done while performing a selection or after making a selection. Marking menus need the ability to show system status (e.g., display the current font), to preview the effects of selecting a menu item (e.g., highlighting a particular font in a menu causes an example of the font to be displayed), and to embed command parameters after a selection is confirmed (e.g., after selecting "volume" a user is automatically connected to a graphical slider). Integrating these features while maintaining the design principles is an open problem.

• 3D marking menus. Marking menus are based on selection by direction in two dimensions with two dimensional pointing devices. A natural generalization is to three dimensions.

• While our research has established some upper bounds on the limits of hierarchic marking menus, a natural extension would be a case study of user behavior with hierarchic marking menus in a real application. We know from our first case study on non-hierarchic menus that with enough practice users will use marks. Hierarchic

menus have many more menu items than non-hierarchic menus. For example, a menu hierarchy which is two levels deep, with eight items in each menu, contains 64 items. It would be interesting to see if this potential could be tapped in an application.

• Further development and evaluation of the crib-sheet/animator is another topic for future research. Clearly, user testing of the design is required. Also developing a parameterized version of the animator is an interesting research challenge.

• Investigating the application of self-revelation, guidance and rehearsal to other domains, besides marking is of interest. An example of the use of guidance and rehearsal in another domain is keyboard driven menus. The menus serve to reveal functionality to a novice, and the novice is guided through the menu by hitting keys to select menu items. This guidance provides a rehearsal of an expert type of behavior in which menu items are selected without looking or waiting for the menus to be displayed.

• There are many open questions concerning using marks and motor behavior. Does using a distinct gesture when drawing a mark have an advantage? What is a distinct gesture? Are there ways that we can design the gestures of drawing marks such that learning or performance is improved?

7.4. FINAL REMARKS

The interfaces to many ordinary, non-computerized objects have properties which make human operation of them second nature. For example, gear-shifts and turn-signal levers in automobiles have labels which we initially look at to learn the function mappings but with experience these mappings become automatic. Furthermore, with practice, the gestures of operating these devices become secondary to the task of driving. The fact that the gestures are unique contribute to our ability to perform them with very little attention. This provides the advantage of allowing our attention to be focused on other more important tasks, for example, watching traffic or reading street signs.

In this thesis, we have tried to exploit these types of properties in the realm of the computer interface. As computers become more entrenched as our everyday objects, tools and instruments, it is not unreasonable to expect them to exhibit the

properties that make many non-computerized objects easy and effective to use. This dissertation contributes to the understanding and creation of human computer interactions that have these properties.

References

Allen, R. B. (1983) Cognitive factors in the use of menus and trees: an experiment. *IEEE Journal on Selected Areas in Communications,* SAC 1(2), 333-336.

Apple Computer (1992) *Hypercard User's Guide.* Apple Computer, Cupertino, California.

Baecker, R., & Small, I. (1990) Animation at the interface. In Laurel, B. (Ed.)*The Art of Human-Computer Interface Design,* 251-267, Reading Massachusetts: Addison Wesley.

Baecker, R., Small, I., & Mander, R. (1991) Bringing icons to life. *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems,* 1-6, New York: ACM.

Baecker, R. M. (1969) Picture-driven animation. *Proceedings of the 1969 Spring Joint Computer Conference,* 273-278.

Barnard, B. J., & Grudin, J. (1988) Command Names. In Helander, M. (Ed.) *Handbook of Human Computer Interaction*, 237-255, B. V. North Holland: Elsevier Science.

Boritz, J., Booth, K. S., & Cowan, W. B. (1991) Fitts's law studies of directional mouse movement. *Proceedings of Graphics Interface '91,* 216-223.

Bush, V. (1945) As We May Think. *Atlantic Monthly.* July, 101-108.

Buxton, W. (1986). Chunking and phrasing and the design of human-computer dialogues. In Kugler, H. J. (Ed.) *Information Processing '86, Proceedings of the IFIP 10th World Computer Congress*, 475-480, Amsterdam: North Holland Publishers.

Buxton, W. (1990) The "Natural" Language of Interaction: A Perspective on Nonverbal Dialogues. In Laurel, B. (Ed.)*The Art of Human-Computer Interface Design,* 405-416, Reading Massachusetts: Addison Wesley.

Callahan, J., Hopkins, D., Weiser, M., & Shneiderman, B. (1988) An empirical comparison of pie vs. linear menus. *Proceedings of the CHI '88 Conference on Human Factors in Computing Systems,* 95-100, New York: ACM.

Card S. K. (1982) User perceptual mechanisms in the search of computer command menus. *Proceedings of the CHI '82 Conference on Human Factors in Computing Systems,* 190-196, New York: ACM.

Card, S. K., Moran, T. P., & Newell, A. (1983) *The Psychology of Human-Computer Interaction.* Hillsdale NJ: Lawerence Erlbaum.

Card, S. K., Robertson, G. G., & Mackinlay, J. D., (1991) The information visualizer, an information workspace. *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems,* 181-188, New York: ACM.

Carroll, J. M, (1985) *What's in a name?* New York: Freeman.

Carroll, J. M., & Carrithers, C. (1984) Training wheels in a user interface. *Communications of the ACM*, 27, 800-806.

Coleman, M. L. (1969) Text Editing on a Graphics Display Device Using Hand-drawn Proofreader's Symbols. *Proceedings of the Second University of Illinois Conference on Computer Graphics.* 282-291, Chicago: University of Illinois Press.

Cullingford, R. E., Krueger, M. W., Selfridge, M., & Bienkowski, M. A. (1982) Automated explanations as a component of a computer-aided design system. *IEEE Transactions on System, Man and Cybernetics,* March/April, 168-181

Ellis, T. O., & Sibley, W. L. (1967) On the Development of Equitable Graphic I/O. *IEEE Transactions on the Human Factors in Electronics.* 8(1), 15-17.

Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., Tang, J., & Welch, B. (1992) Liveboard: A large interactive display supporting group meetings. presentations and remote collaboration. *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems,* 599-607, New York: ACM.

Fischman, M. G. (1984) Programming time as a function of number of movement parts and changes in movement direction. *Journal of Motor Behavior,* 16(4), 405-423.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 381-391.

Furnas, G., Gomez, L., Landauer, T., & Dumais, S. (1982) Statistical Semantics: How can a computer use what people name things to guess what things people mean when they name things? *Proceedings of the CHI '82 Conference on Human Factors in Computing Systems,* 251-253, New York: ACM.

Gaver, W., W. (1991) Technology affordances, *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems,* 79-84, New York: ACM.

Gibson, J. J. (1979) *The ecological approach to visual perception.* Houghton Mifflin, New York.

Gibson, J. J. (1982) *Reasons for realism: Selected essays of James J. Gibson.* Reed, E. & Jones, R. (Ed.), Hillsdale NJ: Lawerence Erlbaum.

Go (1991) *PenPoint System Manual*, Go Corporation, Foster City, CA.

Goldberg D., & Goodisman A. (1991) Stylus User Interfaces for Manipulating Text. *Proceedings of the ACM Symposium on User Interface Software and Technology,* 127-135, New York: ACM.

Gould, J. D., & Salaun, J. (1987) Behavioral Experiments in Handmarks. *Proceedings of the CHI + GI '91 Conference on Human Factors in Computing Systems and Graphics Interface,* 175-181, New York: ACM.

Guiard, Y., Diaz, G., & Beaubaton, D. (1983) Left-hand advantage in right-handers for spatial constant error: preliminary evidence in a unimanual ballistic aimed movement. *Neuropsychologia,* Vol. 21, No. 1, 111-115.

Hardock, G. (1991). Design issues for line driven text editing/annotation systems. *Proceedings of the Graphics Interface '91 Conference,* 77-84, Toronto: Canadian Information Processing Society.

Hoeber, T. (1988) Face to face with Open Look, Byte, V(13) (Dec. '88), 286-288.

Hopkins, D. (1987) Direction selection is easy as pie menus! *login: The USENIX Association Newsletter*, 12(5), 31-32.

Hopkins, D. (1991) The design and implementation of pie menus. *Dr. Dobb's Journal*, 16(12), 16-26.

Hornbuckle, G. D. (1967) The Computer Graphics User/Machine Interface. *IEEE Transactions on the Human Factors in Electronics.* 8(1), 17-20.

Jorgensen, A. H., Barnard, P., Hammond, N., and Clark, I., (1983) Naming commands: An analysis of designers' naming behavior. *Psychology of computer use,* Green T. R. G., Payne S. J., and van derr Veer, G. C. (Eds.), 69-88, London: Academic Press.

Keele, S. W. (1968) Movement control in skilled motor performance, *Psychological Bulletin*, 70, 387-403.

Kiger, J. L. (1984) The depth/breadth tradeoff in the design of menu-driven user interfaces. *International Journal of Man Machine Studies*, 20, 210-213.

Kirk, R. E. (1982) *Experimental design: Procedures for the Behavioral Sciences.* Belmont California: Wadsworth.

Krueger M. W., Giofriddo T., & Hinrichsen K. (1985) VIDEOPLACE—An Artificial Reality. *Proceedings of the CHI '85 Conference on Human Factors in Computing Systems,* 35-40, New York: ACM.

Kurtenbach, G. & Baudel, T. (1992) HyperMark: Issuing commands by drawing marks in Hypercard. *Proceedings of CHI '92 Conference poster and short talks,* 64, New York: ACM.

Kurtenbach, G. & Buxton W. (1991) Issues in combining marking and direct manipulation techniques. *Proceedings of UIST '91 Conference*, 137-144, New York: ACM.

Kurtenbach, G. & Buxton, W. (1991) GEdit: A testbed for editing by contiguous gesture. *SIGCHI Bulletin,* 22-26, New York: ACM.

Kurtenbach, G. & Buxton, W. (1993) The limits of expert performance using hierarchical marking menus. to appear in *Proceedings of the CHI '93 Conference on Human Factors in Computing Systems,* New York: ACM.

Kurtenbach, G. & Hulteen, E. (1990) Gesture in Human-Computer Communication. In Laurel, B. (Ed.)*The Art of Human-Computer Interface Design,*, 309-317, Reading Massachusetts: Addison Wesley.

Kurtenbach, G., Sellen, A., & Buxton, W. (1993) An empirical evaluation of some articulatory and cognitive aspects of "marking menus". *Human Computer Interaction* , 8(2), 1-23

Landauer, T. K. & Nachbar, D. W. (1985) Selection from alphabetic and numeric trees using a touch screen: breadth, depth and width. *Proceedings of the CHI '85 Conference on Human Factors in Computing Systems,* 73-78, New York: ACM.

Lee, E. & MacGregor, J. (1985) Minimizing user search time in menu driven systems. *Human Factors*, 27(2), 157-162.

Licklider, J. C. R. (1960) Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics.* March 1960, 4-11.

Lieberman, H. (1987) An example-based environment for beginning programmers. *AI and Education: Volume One*, Lawler, R. and Yazdani, M., (Ed.), 135-152, Norwood NJ: Ablex Publishing.

Mackenzie, I. S. & Buxton, W. (1992) Extending Fitts' law to two-dimensional tasks. *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems,* 219-226, New York: ACM.

Mackenzie, I. S., Sellen, A. J., and Buxton, W. (1991) A comparison of input devices in elemental pointing and dragging tasks. *Proceedings of the CHI '91 Conference on Human Factors in Computing Systems,* 161-166, New York: ACM.

Makuni, R. (1986) Representing the Process of Composing Chinese Temples. *Design Computing*. Vol. 1, 216-235.

Malfara, A. & Jones, B. (1981) Hemispheric asymmetries in motor control of guided reaching with and without optic displacement. *Neuropsychologia,* Vol. 19, No. 3, 483-486.

McDonald, J. E., Stone, J. D., & Liebelt, L. S. (1983) Searching for items in menus: The effects of organization and type of target. *Proceedings of Human Factors Society 27th Annual Meeting.* 834-837, Santa Monica, CA: Human Factor Society.

Momenta, (1991) *Momenta User's Reference Manual.* Momenta, 295 North Bernardo Avenue, Mountain View, California.

Morrel-Samuels, P. (1990) Clarifying the distinction between lexical and gestural commands. *International Journal of Man-Machine Studies* , 32, 581-590.

Nilsen, E. L. (1991) *Perceptual-motor control in human-computer interaction.* Technical Report No. 37, University of Michigan, Cognitive Science and Machine Intelligence Laboratory.

Norman, D. A. & Draper, S. W. (1986) *User centered system design: New perspectives on human-computer interaction.* Hillsdale, NJ: Erlbaum Associates.

Norman, D. A. (1981) Categorization of action slips. *Psychological Review*, 88, 1-15.

Normile, D. & Johnson, J. T. (1990). Computers without keys. *Popular Science*, August 1990, 66-69.

Paap, K. R. & Roske-Hofstrand, R. J. (1986) The optimal number of menu options per panel. *Human Factors*, 28(4), 1-12.

Paap, K. R. & Roske-Hofstrand, R. J. (1988) Design of Menus. *Handbook of Human Computer Interaction*, Helander, M. (Ed.), 205-235, B. V. North Holland: Elsevier Science.

Pederson, E. R., McCall, K., Moran, T. P., & Halasz, F. G. (1993) Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. to appear in *Proceedings of the CHI '93 Conference on Human Factors in Computing Systems,* New York: ACM.

Perkins R., Blatt L. A., Workman D., & Ehrlich S. F. (1989) Interactive tutorial design in the product development cycle. *Proceeding of the Human Factors Society 33rd Annual Meeting,* 268-272.

Perlman, G. (1984) Making the right choices with menus. *Proceedings of Interact '84,* 317-320, B. V. North Holland: Elsevier Science.

Rasmussen, J. (1983) Skills, Rules and Knowledge: Signals, Signs and Symbols and other Distinctions in Human Performance Models. *IEEE Transactions on Systems, Man and Cybernetics,* SMC-13, 257-266.

Rebello, K. (1990) New PCs can kiss keyboards good-bye. *USA Today*, Feb. 22., 6B.

Rhyne, J. R. & Wolf, C. G. (1986) *Gestural Interfaces for Information Processing Applications.* IBM Technical Report 12179 (#54544).

Rhyne, J. R. (1987) Dialogue Management for Gestural Interfaces. *ACM Computer Graphics.* 21(2), 137-142.

Robertson, G. G., Henderson, Jr. A. D., & Card S. K., (1991) Buttons as First Class Objects on an X Desktop. *Proceedings of UIST '91 Conference,* 35-44, New York: ACM.

Rubine, D. (1991) Specifying Gestures by Example. *Computer Graphics,* 25(4), 329-337.

Rubine, D. H. (1990) *The Automatic Recognition of Gestures.* Ph.D. Thesis, Dept. of Computer Science, Carnegie Mellon University.

Sellen, A. J., Kurtenbach, G., & Buxton, W. (1992) The prevention of mode errors through sensory feedback. *Human-Computer Interaction.* 7(2), 141-164.

Sellen, A. J. & Nicol, A. (1990) Building user-centered on-line help. In Laurel, B. (Ed.)*The Art of Human-Computer Interface Design,* 143-153, Reading Massachusetts: Addison Wesley.

Sellen, A. J. (1992) Speech patterns in video-mediated conversation, *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems,* 49-59, New York: ACM.

Shneiderman, B. (1987) *Designing the User Interface: Strategies for Effective Human Computer Interaction.* Reading Massachusetts: Addison-Wesley.

Sibert, J., Buffa, M. G., Crane, H. D., Doster, W., Rhyne, J. R., & Ward, J. R. (1987) Issues Limiting the Acceptance of User Interfaces Using Gestures Input and Handwriting Character Recognition. *Proceedings of the CHI + GI '91 Conference on Human Factors in Computing Systems and Graphics Interface,* 155-158, New York: ACM.

Snoddy, G. S. (1926) Learning and stability. *Journal of Applied Psychology* 10, 1-36.

Snowberry, K., Parkinson, S. R., & Sisson, N. (1983) Computer display menus. *Ergonomics,* 26(7), 699-712.

Sukaviriya, P. & Foley, J. D. (1990) Coupling a UI framework with automatic generation of context-sensitive animated help. *Proceedings of the ACM Symposium on User Interface Software and Technology '88,* 152-166, New York: ACM.

Sukaviriya, P. (1988) Dynamic construction of animated help from application context. *Proceedings of the ACM Symposium on User Interface Software and Technology '88,* 190-202, New York: ACM.

Sutherland, I. E. (1963) Sketchpad: A man-machine graphical communication system. *AFIPS Conference Proceedings 23,* 329-346.

Walker, N., Smelcer, J. B., & Nilsen, E. (1991) Optimizing speed and accuracy of menu selection: a comparison of walking and pull-down menus. *International Journal of Man-Machine Studies*, 35, 871-890.

Ward, J. R. & Blesser, B. (1985) Interactive Recognition of Handprinted Characters for Computer Input. *IEEE Computer Graphics & Algorithms* . Sept. 1985, 24-37.

Weiser, M. (1991) The computer for the 21st century. *Scientific American,* 265(3), 94-104.

Welbourn, L. K. & Whitrow, R. J. (1988) A gesture based text editor. *People and Computers IV*, *Proceedings of the Fourth Conference of the British Computer Society Human-Computer Specialist Group.* 363-371, Cambridge UK: Cambridge University Press.

Westheimer, G. & McKee, S. P. (1977) Spatial configurations for visual hyperacuity. *Vision Research*, 17, 941-947.

Wiseman, N. E., Lemke, H. U., & Hiles, J. O. (1969) PIXIE: A New Approach to Graphical Man-machine Communication. *Proceedings of 1969 CAD Conference Southhampton*, 463, IEEE Conference Publication 51.

Wixon, D., Whiteside, J., Good, M., & Jones, S. (1983) Building a user-defined interface. *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems,* 185-191, New York: ACM.

Wolf, C. G. & Morrel-Samuels, P. (1987) The use of hand-drawn gestures for text editing. *International Journal of Man-Machine Studies* , 27, 91-102.

Wolf, C. G. (1986) Can People Use Gesture Commands? *ACM SIGCHI Bulletin*, 18, 73-74, Also *IBM Research report* RC 11867.

Wolf, C. G., Rhyne, J. R., & Ellozy, H. A., (1989). The paper-like interface. *Designing on Using Human Computer Interface and Knowledge-Based Systems*. 494-501, B. V. North Holland: Elsevier Science.

X11 (1988) *X window system user's guide for version 11.* X window series, v. 3, Sebastopol CA: O'Reilly & Associates._

Appendix A: Statistical Methods

This appendix explains the statistical methods used in this dissertation. Analysis of variance (ANOVA) is used for hypothesis testing. Specifically, the *F*-statistic is used to determine if an *independent variable* has any effect on a *dependent variable.* In an experiment, the dependent variable is a variable being measured. The independent variable is a variable being controlled.

Testing for differences in means: $F(k - 1, k(n -1)) = f, p < \alpha$.

Data is grouped according to different values of the independent variable. Each group is commonly referred to as a *treatment.* Random samples of size n are selected from each of k treatments. It is assumed that the k treatments each have a population that is independent and normally distributed with means $\mu_1, \mu_2, \ldots, \mu_k$ and a common variance σ^2. The null hypothesis can be represented as:

$$\mu_1 = \mu_2 = \ldots = \mu_k$$

The ANOVA procedure separates the total variability of the samples into two component: s_1^2 and $s^{2\cdot}$ The variance s_1^2 is the variability between treatments attributed to changes in the independent variable and chance or random variation. The variance s^2 is the variability within treatments due to chance or random variation.

It can be shown that, assuming the null hypothesis is true, the ratio:

$$f = s_1^2/s^{2\cdot}$$

is a value of the random variable F having the F distribution with k - 1 and $k(n - 1)$ degrees of freedom. Since s_1^2 overestimates the true variance when the null hypothesis is false, a large value for f suggest a large portion of the variance in the

dependent variable is caused by the independent variable. A test can be done by comparing the observed value *f* with the theoretical value of $F(k - 1, k(n - 1))$ and reporting the probability, *p*, of such a large value for *f* occurring simply by chance. If *p* is very small (e.g., $p < .05$), this suggests that the null hypothesis should be rejected.

Multiple comparison of means: Tukey HSD, $\alpha = p$

After determining a significant *f* ratio, it is may be necessary to determine which pairs of means are significantly different. Various procedures, which are referred to as post-hoc comparisons, allow this. If means μ_1 and μ_2 are being compared, the null hypothesis is:

$$\mu_1 - \mu_2 = 0.$$

A Tukey HSD post-hoc test reports the significantly differing means with a probability of α of incorrectly rejecting the null hypothesis (i.e., no difference exists between the means). Generally a .05 level of significance is used. This means one can be 95% sure that two means actually differ.

Contrasting means: $F(1) = f$, $p < \alpha$.

Post-hoc tests are not available for within subjects factors in repeated measures experimental design. An alternative method for determining which pairs of means are significantly different is by contrasting means. ANOVA separates the variance into two components: *SSw* and s^2. *SSw* is the variance attributed to the difference between the means. The variance s^2 is the variability due to chance or random variation.

It can be shown that, assuming the null hypothesis is true, the ratio:

$$f = SSw/s^2.$$

is a value of the random variable *F* having the *F* distribution with 1 and $n - k$ degrees of freedom. Since *SSw* overestimates the true variance when the null hypothesis is false, large values of *f* indicate a large portion of the variance is due to a difference between the means. A test can be done by comparing the observed value *f* with the theoretical value of $F(1, n - k)$ and reporting the probability, *p*, of such a large value

for *f* occurring simply by chance. If *p* is very small (e.g., $p < .05$), this suggests that the null hypothesis should be rejected.

Testing for linear relationships: F(1, n - 2)

The *F*-statistic is used to provide a single significance probability of a linear relationship between dependent and independent variables. In this case, the null hypothesis is that the slope of the regression line is zero. If the null hypothesis is true, then

$$f = SSR/s^{2.}$$

Where *SSR* is the amount of variation explained by the straight regression line. The variance s^2 is the variability around the regression line due to errors. It can be shown *f* is the value of the random variable *F* having the *F* distribution with 1 and *n* - 2 degrees of freedom. A test can be done by comparing the observed value *f* with the theoretical value of $F(1, n - 2))$ and reporting the probability, *p*, of such a large value for *f* occurring simply by chance. If *p* is very small (e.g., $p < .05$), this suggests that the null hypothesis should be rejected.

Testing a linear relationship for goodness of fit: r^2

The sample correlation coefficient r^2 is used to test the quality of the fit of a linear regression line. The amount of variation in the dependent variable which is explained by the independent variable is $r^2 \times 100\%$. A r^2 value greater than .5 is considered to indicate a linear relationship.

For further information on these statistical methods see Kirk (1982).