COMPUTATIONAL MODELS OF THE PERCEPTUAL, COGNITIVE, AND MOTOR PROCESSES INVOLVED IN THE VISUAL SEARCH OF PULL-DOWN MENUS AND COMPUTER SCREENS

by

Anthony John Hornof

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
1999

Doctoral Committee:

Associate Professor David E. Kieras, Chair
Associate Professor William P. Birmingham
Professor John E. Laird
Professor David E. Meyer
Professor Judith S. Olson

© Anthony John Hornof 1999
All Rights Reserved

For all those people

who have a hard time finding

what they are looking for.

ACKNOWLEDGMENTS

This work was supported by the Advanced Research Projects Agency under Order B328 to the University of Michigan, David E. Kieras, principal investigator, and by the Office of Naval Research through Grant N00014-92-J-1173 to the University of Michigan, David E. Kieras and David E. Meyer, principal investigators.

I would like to thank my advisor David Kieras for overseeing my transformation from a New York City business systems consultant to a research university professor. At every step of the way, you asked the right questions and lowered your brows just enough to motivate good performance but never enough so that your patience and kindness failed to shine through. I am also very grateful for the financial support that you provided.

Thank you, David Meyer, for letting me crash all of your graduate seminars on human cognition. I hope to take your Socratic style of interrogation with me to all of the graduate seminars I ever get to teach.

Thank you, Judy and Gary Olson, for setting a model for how to run a welcoming and inclusive laboratory that invites investigation and provides training in all aspects of my favorite field of research, human-computer interaction.

Thank you, John Laird and Bill Birmingham, for your ongoing support of my dissertation research and general intellectual growth, as well as for your thoughtful advice regarding my recent career decisions.

Thank you, Erik Nilsen, for your assistance and critique of my work.

Thank you to my family for your support and encouragement.

Thank you to the Creator, who goes by many names to many people, including but not limited to Allah, Brahma, God, Jah, Jehovah, Jesus, Krishna, and Yahweh.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF APPENDICES

Appendix

CHAPTER 1
INTRODUCTION

A major challenge in making software easy for people to use is to design screen layouts that people can search efficiently. Although there has been a great deal of research on visual search, the field of human-computer interaction (HCI) still does not have an empirically validated model of the perceptual, cognitive, and motor processes that people use when they look for a known item on a computer screen. There are many guidelines to direct the design of computer screen layouts, but few if any have been explained in terms of how these processes give rise to the guidelines. This dissertation presents the first empirically validated models of the perceptual, cognitive, and motor processes involved in the visual search of computer menus, models that can be generalized to explain the cognitive processes involved in more general computer layout visual search tasks on a computer.[1] These models should contribute to the design and analysis of more usable computer systems.

1.1 Computer Interfaces Require Visual Search

People using computers are routinely faced with the challenge of finding something on a computer screen. The task could be as simple as finding the trash can on the Apple Macintosh desktop (it is always at the bottom right hand corner), or as difficult as finding the link to “Liberal Arts” on the Oregon State University web page shown in Figure 1.1.

As more information is delivered via computers (such as the World Wide Web)

---

[1]The same models have already been presented by Hornof and Kieras (1997; 1999).

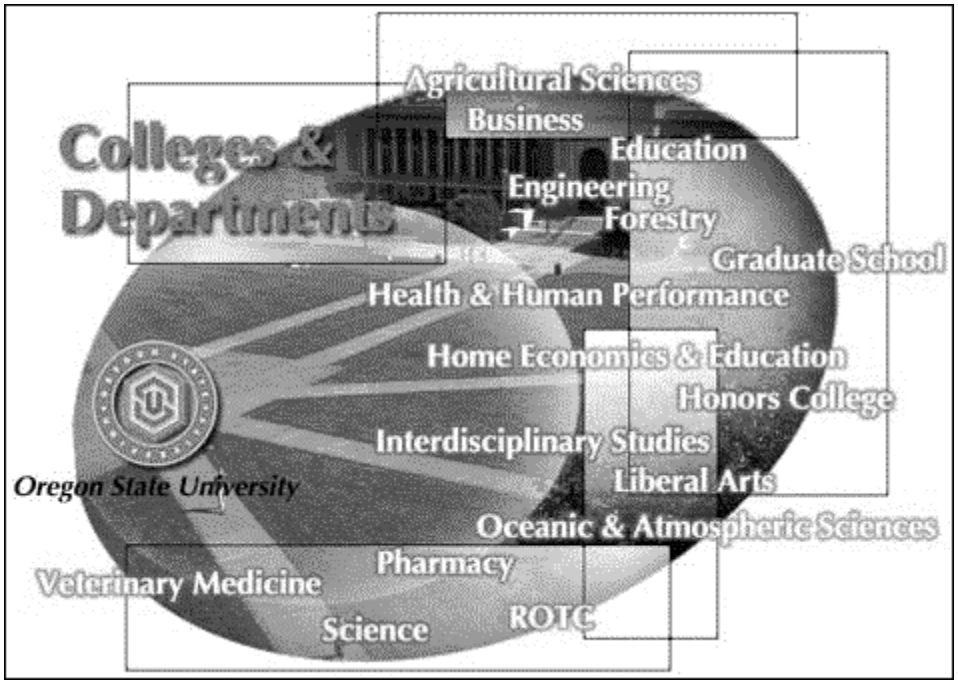

Figure 1.1. The link to "Liberal Arts" is very difficult to find on this web page because of the random placement of all the items. (http://osu.orst.edu/colleges/colleges.htm, 6/8/99)

rather than print media, and in the absence of typographic standards for computer layouts and in the absence of armies of trained graphic designers, people will continue to be faced with difficult visual search tasks.

1.2 Many Screen Layouts Are Unnecessarily Difficult to Search

The World Wide Web offers a cornucopia of screen layouts that are difficult to search, such as that shown in Figure 1.1, but even day-to-day desktop software presents challenging visual search tasks. Software developers have led the public to believe that the more buttons and gadgets that are visible to the user on the computer screen, the better. But the proliferation of menu bars (such as that shown in Figure 1.2) has only added to the difficulty in finding the few things that the user really needs.

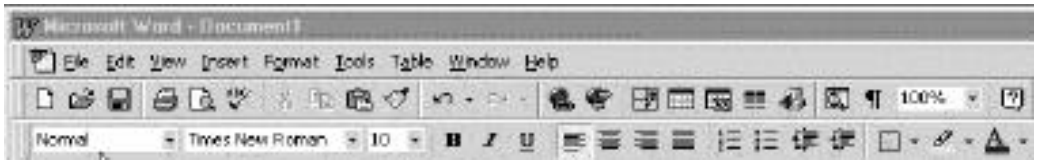

Figure 1.2. The menu bar in Microsoft Word 97. Some users opt to display another three or four rows of buttons, led to believe by software manufacturers that more is better. But more visual distractors will only make it more difficult to find the truly useful visual targets.

1.3 Possible Solutions Exist

It is possible to design computer screens that can be efficiently searched. Guidelines exist (Apple Computer, 1993; Galitz, 1993; Mayhew, 1992; Smith & Mosier, 1986), and most difficult-to-search screen layouts could have been vastly improved if the designers had better followed these guidelines. But sometimes guidelines can be misleading. For example, some designers (Tufte, 1983) profess that visual clutter should be eliminated. This would seem to fix a lot of problems, such as the icon bar proliferation shown in Figure 1.2, but sometimes a crowded layout is actually best.[2] Staggers (1993) demonstrated this in a field study of computer workstations used by nurses; the densest screen layouts evoked the best overall performance, and were also the nurses' preferred layouts. Guidelines need to be further evaluated and improved based on empirical studies, but even those that currently exist can help software designers to develop more usable software.

Guidelines can only take the designer part way to a usable interface; the usefulness of many screen layouts depends largely on the task for which the screen layout is intended. Hence, the true usefulness of a layout will always be an empirical question that can only be answered in the exact context of its intended use. A person designing a screen layout can

---

[2]Carswell (1992) in fact demonstrated that Tufte's (1983) data-ink principle (which states that any redundant display of information should be avoided) is not a good predictor of user performance for a graphical perception task, and that a more principled approach such as Cleveland's basic tasks model (Cleveland & McGill, 1985) is a better predictor.

work with guidelines as best as possible, but he or she cannot conduct a usability study for every single layout he or she proposes. Designers need automated usability prediction tools built directly into the same tools that they use to build their interfaces. The designers would set a usability goal in terms of time, number of steps required to accomplish a task, cognitive or visual complexity, or other objective criteria, and propose and evaluate layout after layout until arriving at one that the tool predicts would meet the design requirements.

Just as guidelines need to be based on empirical observations, a tool that predicts human performance in visual search tasks also needs to be based on a theory of human visual search that is itself based on empirical data. Many theories have been proposed regarding visual search. This dissertation attempts to synthesize a number of the dominant theories into a model that can be used to make a priori predictions regarding how long a person would take to find something on a computer screen layout, and the cognitive processing required. The specific task modeled in this dissertation is that of searching for a known target item in a pull-down menu. Cognitive models of how people search pull-down menus will lead to models of more general search tasks, both of which will lead to better predictive tools and more usable interfaces.

1.4 The Goals of This Research

The high level goals of this dissertation are: (a) Build models that explain the perceptual, cognitive, and motor processes that people use when they search for a known item in a pull-down menu. (b) Validate these models with empirical data. (c) Use the models to propose the perceptual, cognitive, and motor processes that people likely use in more general search tasks on computer screen layouts. (d) Make practical and theoretical contributions, enumerated next.

Practical contributions include: (a) With a better understanding of how people find what they are looking for on a computer screen, provided by this dissertation, screen layout designers will be able to design interfaces that better complement the processes and

strategies that people really use. (b) The dissertation provides a theoretical basis for design guidelines that will help the field of human-computer interaction to better understand which guidelines have true merit and should be promoted. (c) Exploratory modeling of visual search, such as that done this dissertation, is necessary before purely predictive modeling of visual search can be done; this dissertation thus moves the field of human-computer interaction closer to when it can incorporate predictive tools into screen layout design tools.

Theoretical contributions include: (a) This dissertation contributes to basic research in the area of visual search by applying the new theory-building power of cognitive modeling to pre-existing data and tasks, and in doing so resolves theoretical debates that exist in the literature. (b) This dissertation integrates research from many different researchers and many different subfields (including vision, visual search, cognitive psychology, and computer science) and moves the scientific community closer to a realization of Newell's (1990) unified theory of cognition.

1.5 The Structure of This Dissertation

This chapter introduces the high level problem addressed by the dissertation. Chapter 2 discusses previous work done in the major areas of research that feed into this dissertation, including search, visual search, and menu search. Chapter 3 introduces the basic components of the cognitive models of menu search developed for this dissertation. Chapter 4 presents the results of these models. Chapter 5 discusses various implications of the models, such as how they contribute to improving the usability of computer systems.

CHAPTER 2
REVIEW OF RELEVANT RESEARCH

There has been an enormous amount of research done in the fields of visual search, menu search and selection, and cognitive modeling, much of which contributes to the cognitive models of visual search of pull-down menus and computer screens presented in this dissertation. This chapter reviews the literature, emphasizing research that contributes to the modeling work presented in this dissertation, and that demonstrates the need for this new work. The main bodies of literature that will be reviewed include general visual search, visual search of computer menus, and cognitive modeling of human-computer interaction.

2.1 The Origin of the Scientific Study of Search

Many important technological advances have been motivated by military enterprise (Smith, 1985). One of these was the study of *search*. The military began the scientific study of search during World War II because, in this war, search requirements and challenges vastly exceeded those of previous wars. In World War II, complex human-machine technology systems were used to move, hide, search, and kill over much longer distances than in previous conflicts. Developments in electronic search aids to meet these challenges, such as radar and sonar, were required.[3]

The first modern scientific work studying search is a Navy document entitled

---

[3]Dramatic accounts of high-tech searches for submarines, for example, are presented in Sontag and Drew (1998) and Clancy (1984).

Search and Screening (Koopman, 1946). The preface of this report states:

> In World War II, it became progressively more apparent that large classes of problems were united by common bonds and could be handled by common methods, that there was indeed unity in diversity. And as in other fields of scientific endeavor, where the clarifying influence of general ideas and methods can form a body of isolated facts into a powerful theory..., methods borrowed from the mathematician and mathematical physicist showed their power and usefulness in those classes of problems in which the body of practical information had sufficiently accumulated. In this regard, one field was pre-eminently ripe for mathematical treatment: the field involving problems of *search*.
>
> In every question of search there are in principle two parts. One involves the targets, and studies their physical characteristics, position, and motion.... The other part involves the searcher, his capabilities, position, and motion. (p. ix)

This excerpt points out that (a) this was, as far as these scientists could determine, the first scientific investigation of the study of search, and (b) the new field brought together problems and methods from many different fields of study. The latter has remained the case over the years, as will be seen in the discussion of visual search that follows. The excerpt also succinctly states the basic problem of search—a person with certain capabilities looks for a target with certain features. This characterization of the problem pervades all models of search.

Koopman (1946) contains numerous mathematical models for target detection by visual contact, radar, and sonar. The report details tactics for conducting aerial escorts, sonar screens, and other wartime activity. Since it presents techniques for finding and avoiding being found by the enemy, it is easy to understand why the report was initially classified as “confidential.”

But the report is more than just an interesting historical artifact; rather, it seeds the field of visual search. The earliest mathematical models of computer-based menu selection, those of Card (1982; 1983) were influenced by the work of Krendel and Wodinsky (1960), who were influenced by the work of E. S. Lamar, the author of the visual search models presented in Koopman (1946).

2.2 Psychological Research on Visual Search

The two fields that seem to be making the most progress in building comprehensive theories of visual search are the fields of cognitive psychology and eye movement research. A third field that provides a wealth of data, but less theory to integrate it, is the field of human factors. This review will discuss the literature that is most relevant for this dissertation drawing from all three fields—experimental observations from the field of human factors, and theory developed from the fields of cognitive psychology and eye movement research.

The theories developed in cognitive psychology and eye movement research will be presented separately because, when it comes to visual search, the two fields are somewhat disconnected. Cognitive psychology emphasizes the cognitive processes that are required, and tends to assume that eye movements will occur when needed. Eye movement research looks at the processes that would be needed to move the eyes, but says little about how these processes would be integrated into other perceptual, cognitive, and motor processes. One researcher states that "eye movements and visual cognition have in the past seemed like two separate continents separated by an enormous and ill-charted ocean." (Findlay, 1992)

But there is gold in all three continents. This review will first discuss empirical observations, drawing in part from the field of human factors. Then, the review will discuss theories of visual search from cognitive psychology and from eye movement research.

2.2.1 Experimental Phenomena of Visual Search

In visual search experiments, participants are presented with a series of individual tasks, or *trials*, of the following form:

1. The participant is shown the object that he or she will be expect to find, the object that will be the *target* during the search. This preparatory presentation is called the *precue*.

2. The precue disappears and the target appears hidden amidst many

nontarget objects, or *distractors*, usually in an unstructured visual field that has no particular organization. A clock starts to measure how long the participant takes to find the target.

3. The participant searches for the target and indicates when he or she has found it, which stops the clock. The participant also provides some information to confirm that they did actually find the target, such as by indicating the location of the target object.

An example task can be seen in Figure 2.1. The target is the open circle. The distractors are the closed circles.

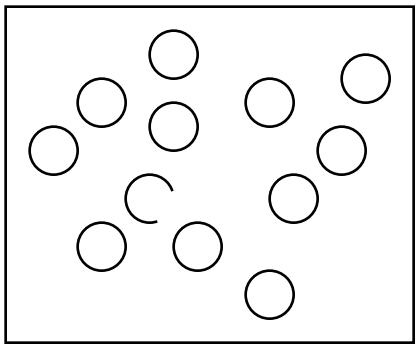

Figure 2.1. An example of a target (an open circle) amidst distractors (closed circles), based on Treisman (1986).

Many visual search experiments have been nicely summarized by Boff and Lincoln (Boff & Lincoln, 1988). Basic phenomena that have been consistently observed include:

1. A person can find a target more quickly when the target can be distinguished from distractors based on a primary feature, such as size, color, shape, or orientation. This is sometimes described as "target pop-out." (Engel, 1977; Treisman, 1986; Williams, 1966a)

2. A person can find a target more quickly when there are fewer distractors present (Drury & Clement, 1978; Neisser, 1963).

3. A person can find a target more quickly when there are fewer different *kinds* of distractors present (i.e., the same number of distractors, but of more different colors, shapes, sizes, or orientations.) (Gordon, 1968).

4. The more discernible the target is from the background (i.e., the greater the contrast), the more quickly a person can find the target (Boynton & Boss, 1971; Krendel & Wodinsky, 1960).

5. The more discernible the target is from the distractors (i.e., the greater the difference in color or size), the more quickly a person can find the target (Bloomfield, 1972; Carter & Carter, 1981).

6. A person can find a target more quickly in a smaller search field (Krendel

& Wodinsky, 1960).

7. A person can find a target more quickly after practice with the same target and distractors (Gordon, 1968; Neisser, 1963).

8. A person can find a target more quickly than they can determine that a target is absent (Jonides & Gleitman, 1972; Neisser, 1963).

9. A person can find a target more quickly when there is a predictable structure to a visual scene (Biederman, 1972; Biederman, Glass & Stacey, 1973).

This is a brief summary of consistently observed phenomena from experiments in which participants were presented with visual search tasks. Many of these studies also propose some sort of small-scale theory to explain the particular phenomena observed in the study. But rather than discussing these isolated theories, this review will instead discuss three comprehensive theories of visual search from the field of cognitive psychology.

2.2.2 Models of Visual Search

This section will present three cognitive psychology models that account for many of the experimental observations made in visual search experiments discussed above. They are Neisser's (1963) simple search model, Treisman's (1986) feature integration theory, and Cave and Wolfe's (1990) guided search model.

Simple Search Model

Neisser's (1963) simple search model assumes that people look at each item in turn and terminate the search when they find the target. The model thus predicts that response time will increase linearly as a function of the total number of objects that must be examined. The model (as presented by Dosher, 1998) states that the mean predicted response time when a target is absent will be

$$RT_a = \alpha_a + \beta(m_s)$$

and that the mean predicted response time when a target is present will be

$$RT_p = \alpha_p + \frac{\beta(m_s + 1)}{2}.$$

where m_S represents the total number of items in the display; β represents the amount of search time required to examine each item in a serial, non-overlapping search; and α represents a constant amount of time required for other processes not related to the number of items in the display (which might change depending on whether the target is absent or present).

This model can explain the preceding experimental observations 2 and 8.

Feature Integration Model

Treisman's (1986) feature integration theory pertains to what kind of pre-attentive visual information is placed in visual working memory and is thus made available to a search strategy. The theory is not a complete model of visual search. However, when combined with a theory of parallel and serial search strategies, the theory becomes a complete model of visual search.

The feature integration theory asserts that early in the processing of visual information, a separate *feature map* is created for each primary feature (such as size, color, shape, and orientation), a map that encodes the location of all occurrences of that feature. These feature maps are searched in parallel, pre-attentively, for the presence of a single object that possesses a simple feature corresponding to one of the maps. If the target item possesses such a feature, it can be found very quickly and without serial attentive search. To the participant, the target object appears to "pop out" without search. If the target item does not possess a distinguishing primary feature, then the model must resort to a serial search process much like Neisser's (1963) simple search model. Hence, the model asserts an initial parallel search followed, if necessary, by a secondary serial search.

This model can explain all of the preceding experimental observations except (#9) that a person can find a target more quickly when there is a predictable structure to a visual

scene.

Guided Search Model.

Cave and Wolfe's (1990) guided search model asserts a slightly different combination of parallel and serial processes. In the guided search model, an activation score is calculated for each display object in an early, parallel processing stage. The activation score combines target-driven and display-driven contributions. The target-driven contribution is a measure of how similar the object is to the anticipated target along various stimulus dimensions. The display-driven contribution is a measure of how unusual the object is in the display along various stimulus dimensions. These stimulus dimensions correspond roughly to the primary features in Treisman's feature integration theory.

This activation score is computed for every item in the display in an early parallel processing stage, which is followed by a serial processing stage in which items are evaluated in order of their activations. The object with the highest activation score is evaluated first, followed by the object with the next highest activation score, and so forth. The sequence of evaluation stops either when the target is found or when all items above a threshold or criterion activation level have been examined. This threshold can be varied depending on task requirements. An additional detail is that a tiny bit of noise, or variability, is added to each activation before ordering them; as a result, even if the target item has a slightly higher activation than some similar nontargets, in the final ordering those nontargets might get ranked higher and evaluated earlier in the serial stage.

Like the feature integration model, this model can also explain all of the preceding experimental observations above except (#9) that a person can find a target more quickly when there is a predictable structure to a visual scene.

It is interesting to note that Cave and Wolfe used a computer program to compute all of the activation scores for all of the visual objects in their guided search model, and to generate the model's predictions. Doing so, they demonstrated how *computational*

cognitive modeling may be the most practical approach to building simulations of human visual search, simulations that will need to incorporate many complex interacting systems.

Summary

This section presented three visual search models: Neisser's simple search model, Treisman's feature integration theory, and Cave and Wolfe's guided search model. All of these models discuss the perceptual, cognitive, and motor processes involved in visual search. Neisser proposed a simple serial search strategy. Treisman and Cave and Wolfe account for preattentive processing and combine both serial and parallel consideration of visual objects. But something is missing from all three models: Though all of the models the models assume that eye movements will take place, none of the models explicitly predict or account for them; instead, they discuss the "focusing of attention," which may or may not correspond to eye movements. As mentioned earlier, there seems to be a separation between the literature discussing the cognitive processing involved in visual search and the literature discussing eye movements. Theories pertaining to eye movements and eye movement programming will be presented next.

2.2.3 The Programming of Eye Movements for Visual Search

Humans perceive their visual environment via light that enters the eyeball through the *cornea*, the transparent front surface, passes through a lens, and falls on the *retina*, the thin, light-sensitive surface that lines the inside back of the eyeball. The retina contains millions of receptor cells known as *rods* and *cones*. There is a small area on the retina opposite the cornea, the *fovea*, in which the rods and cones are most dense and thus resolution of visual information is the greatest. The fovea corresponds to a field of view that is roughly circular and roughly two degrees of visual angle in diameter. Around the fovea, resolvability tapers gradually.

When conducting a visual search, a person moves his or her eyes to get different

areas of the visual scene into the fovea (to *foveate* different areas), so that those areas can be examined with greater precision. In a nonmoving scene, these eye movements will be *saccades*, quick, ballistic movements, typically lasting about 5 to 30 msec and separated by saccadic latencies of at least 200 msec (Rosenbaum, 1991). The lack of eye movement between two successive saccades is by default a *fixation*, during which visual information is gathered. Saccades have two main features: direction and extent. Saccades are typically synchronized so that the same image will fall into the fovea of both eyeballs.

There is an extensive literature of theories regarding the preparation, or *programming*, of eye movements (Fisher, Monty & Senders, 1981; Monty & Senders, 1976; Rayner, 1992; Senders, Fisher & Monty, 1978). One basic question that these researchers ask is how and when the features (direction and extent) of a saccade are programmed. Two major opposing ideas regarding high level control of saccade programming have been proposed, *global control* and *local control*. Under global control, saccades are programmed in advance using strategies that are based on the structure of the visual field and the task. Under local control, saccades are programmed in response to visual information that becomes available during task execution. Enough evidence supports both ideas, however, to suggest that saccade programming is both strategy-driven *and* feature-driven.

It is easy to imagine that both strategy and perceptual features contribute to the programming of saccades. A person looking for a hardware store while riding a city bus, for example, will make strategy-driven saccades to look at each storefront. But perceptual features (such as color, size, or shape) will guide the eyes to the sign identifying each store.

Feature-driven theories have been proposed for a variety of tasks. Antes (1974) and Loftus (1976) studied how people accomplish a picture-viewing task, and proposed that eye movements are programmed based on perceptual features in the visual field. Williams (1966a) studied how people accomplish a visual search task, and also concluded

that peripherally available perceptual information can be used to direct the gaze.

Feature-driven models of saccade programming can be further separated into (a) *direct moment-to-moment control* models, which use features gathered during one fixation to program the features of the very next saccade, and (b) *indirect local control* models, which use features gathered during the last several fixations to gradually modify a scan path of saccades. Prinz, Nattkemper, and Ullmann (1992) made this distinction and collected data that support the idea that people use direct moment-to-moment control.

Other researchers have emphasized the strategy-driven component in the programming of eye movements for visual search. Neisser (1976) asserted that people use a stored pre-programmed *plan* or a *schema* to program eye movements, but that the schema can be influenced by both perceptual information and other task details gathered during schema execution.

Russo (1978) also emphasized the contribution of strategies. He studied the relationship between eye movements and cognitive strategies to account for the time course of cognitive processes that gives rise to the interval between saccades. Russo points out that eye movements themselves take only about 30 msec, but are separated by about 200 msec of no movement; he emphasizes that a good model should account for *all* cognitive processing occurring during the 230 msec between successive saccade initiations.

> Eye fixations serve only to acquire the information needed to execute a given cognitive strategy. Therefore, "interpreting" eye fixations should imply identifying the underlying cognitive strategy. Instead, the use of eye movements to study cognitive processes has been characterized by sterile analyses based exclusively on summary statistics, such as fixation frequency, spatial distribution, and mean duration.... Rather, eye movements should be aggregated into meaningful cognitive units or examined for interpretable sequential patterns.... Because eye movements are always directed by the active cognitive process, an explanation of the eye movements must rely on an understanding of the controlling cognitive strategy. (Russo, 1978, p. 109)

Russo emphasizes strategy-driven analysis, but does not claim that such analysis can exclusively account for eye movements. He proposes a stage model that incorporates (a) strategies, (b) peripherally visible information, and (c) information gathered from previous

fixations, but not necessarily the immediately prior fixation. The model uses all three sources of information to program each saccade.[4]

Other researchers have validated such an integrative explanation. Cohen (1981), for example, provided evidence that people use schemata that are influenced by perceptual features.

A good model of visual search should account for all nine of the basic visual search phenomena enumerated in the previous section. A good model should also account for eye movements, strategy-driven and feature-driven aspects of visual search, and be explicit regarding whether the feature-driven component of the search assumes direct moment-to-moment control (in which each fixation can influences the very next saccade) or indirect local control (in which the last several fixations influence the next saccade). Though a great deal of work has been done to account for reaction time and eye movement data, there is not yet a model of visual search that explicitly accounts for all of these processes and all of the data.

2.2.4 Tools for Predicting Visual Search of Computer Interfaces

If an empirically validated model of the perceptual, cognitive, and motor processes involved in visual search did exist, it could be used to predict how long a person would take to find something on a computer screen. This would be very useful to interface designers. Several researchers have already developed automated techniques for predicting aspects of visual search in human-computer interaction tasks (Lohse, 1993; Sears, 1993; Tullis, 1988).

Tullis (1988) built a tool called the Display Analysis Program (DAP) that takes as input an alphanumeric computer screen layout and analyzes the layout with respect to

---

[4]This is a particularly interesting model because the menu search models presented later in this dissertation also combine all three sources of information to predict eye movements.

grouping, density, and layout complexity. DAP predicts the time required for a user to find any object on the screen based on those input parameters. DAP cannot take a specific search task as an input, but instead predicts one mean search time per display. Though DAP has predictive power, it says nothing and make no claims about the underlying processes involved.

Lohse (1993) developed a system called Understanding Cognitive Information Engineering (UCIE) that predicts the time required to answer a specific question based on information presented in a line graph, bar graph, or table. He attached timing parameters to eye fixations and other component processes. UCIE predicts total task execution time by summing the time required for all component tasks. Unlike DAP, UCIE makes some claims and assumptions about perceptual, cognitive, and motor processes involved. Though UCIE might generalize to other visual search tasks, it was only built and validated for graphs and tables. It does not address more general visual search tasks that arise when using a computer.

Sears (1993) developed a metric called Layout Appropriateness (LA) that can be applied to many computer search tasks. LA evaluates a screen layout with respect to how well it supports efficient execution of a task specified by the analyst. The tool is based on the human factors method called *link analysis*, which is used to determine the optimal placement of equipment in a room. In link analysis, a designer defines links between pieces of equipment that a person will need to use in consecutive order, and labels each link with how frequently that movement will need to be made. The designer then determines the optimum placement of equipment to minimize the overall distance that a person will need to travel while using the equipment. Sears' tool works similarly, and recommends where to place visual objects on a screen to reduce the distance the cursor and the eyes will need to travel while accomplishing a specific task. One shortcoming with LA is that it reduces the distances the eyes will need to travel but not the number of fixations required, which might thus result in little overall time savings because eye movements are fast

compared to fixations. Another shortcoming is that it does not simulate the perceptual, cognitive, and motor processes required for a task, but stays on a higher level of analysis. It does not predict how long a person would take to accomplish a task, but just compares each layout against a theoretically LA-optimal layout.

The tools built by previous researchers suggest that even better predictive tools could be built, and that they would be very useful. Such tools could be used to predict how long a person would take to find something on a computer screen, such as an item on a pull-down menu. The next section delves into this specific subproblem—visual search of pull-down menus.

2.3 Menu Search

Ever since computers advanced to where the user could interact directly with the machine via a keyboard and video display terminal, menu systems have been used to visually inform the user of the available command options at any point in time, and to enable the user to select one of the options. Computer menus are one of the most important visual components of contemporary graphical user interfaces, which incorporate all sorts of menus.

Menus pervade contemporary computer systems. Most textbooks on human-computer interaction or interface design and analysis have a chapter dedicated to the topic (such as Mayhew, 1992; Shneiderman, 1992). Most web pages contain a menu in one form or another; in a sense, every web page that contains links is a menu, and every link is a menu option. Pull-down menus are used extensively by most contemporary computer operating systems. For example, menus are the starting point for launching applications in Microsoft Windows (see Figure 2.2, left), and are used by most computer programs that run on the Apple Macintosh (see Figure 2.2, right).

Computer menus pervade life in the United States. Computer menus are used in bus information kiosks in downtown Portland, Oregon; computer menus have replaced the

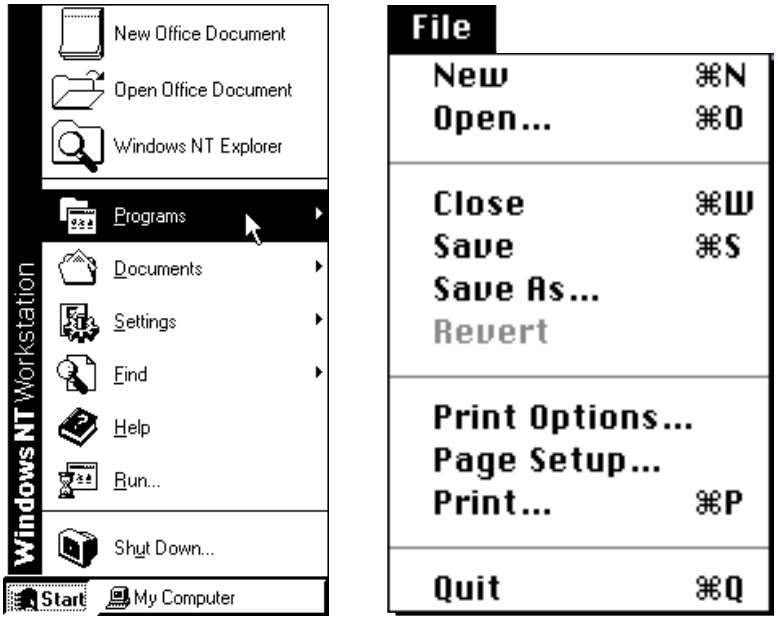

Figure 2.2. On the left, the Microsoft Windows NT "Start" menu, opened millions times across the world every workday. On the right, the standard "File" menu that appears in most Macintosh applications.

old wall-mounted building directories in New York City office buildings; computer menus appear on new televisions across the country when the viewer tries to adjust the picture quality or other settings using the remote control. At many restaurants, cashiers and waiters enter food orders by finding the items on computer menus.

Auditory computer menus also pervade life in the United States, menus such as those encountered when calling an airline ("Press 1 to make a reservation, press 2 for flight information..."). Auditory menus are discussed by Resnick (1993), Raman (1997), and others. But most computer menus are visual menus, and thus require visual search.

2.3.1 Menu Design Guidelines

Researchers have proposed many guidelines to help people design menu systems that are efficient and easy to use (for example, Galitz, 1993; Mayhew, 1992; Shneiderman, 1992; Smith & Mosier, 1986). The guidelines are often based on empirical studies that determine which design choices result in superior performance. Sometimes the guidelines

are based on common practice or supposition. This section will present a comprehensive list of guidelines that has proposed by Mayhew (1992). The list seems particularly useful to designers because of its level of detail and the directness of its recommendations. Smith and Mosier (1986) provide an equally useful but more detailed set of guidelines. Galitz (1993) provides a list that is too detailed, and almost reads as if it were a design specification. Shneiderman (1992) provides a list that is too vague.

To explain how these guidelines might help a person use a menu more quickly and easily, the guidelines can be examined in terms of the menu usage subtasks they are likely to affect. Subtasks in using a menu include:

1. Decide if the menu can be bypassed entirely by using a shortcut key. If not, continue...
2. Open the menu and/or move your gaze to the menu.
3. Figure out where to look next in the menu, and move your gaze there.
4. Find the best match (either an identical match or a close approximation of the target).
5. Figure out how to select the found item. For example, figure out exactly where to point or which key to press.
6. Select the desired item, and confirm that you successfully selected it.
7. If necessary, navigate through a series of menus.
8. Learn the menu items as you use the menu.

Menu design guidelines proposed by Mayhew (1992) are shown in Figure 2.3. Also shown are the subtasks that each guideline is likely to help a person accomplish more quickly or easily.

Examining each guideline in terms of the perceptual, cognitive, and motor processing it is likely to influence should help the field of HCI (a) to better understand the usefulness and validity of each guideline, and (b) to propose *new* guidelines based on an understanding of the perceptual, cognitive, and motor processing that takes place during menu selection.

	Search-related subtasks			Selection-related subtasks				
Menu Design Guidelines	Bypass menu	Open menu	Figure out where to look next	Find best match	Figure out how to select	Select item	Navigate through hierarchy	Learn menus
Establish conventions for menu design and apply them consistently on all menus within a system.	X	X	X	X	X	X	X	X
Allow direct access through type-ahead, menu screen names, and user-created macros for expert users.	X							
Use permanent menus when possible. Reserve pop-up or user-invoked menus for high-frequency users and situations where screen real estate is scarce.		X						
On full-screen text menus, present menu choice lists vertically.			X	X		X		
Make menu choice labels brief, consistent in grammatical style and placement, and matched with corresponding menu titles.			X	X				
Create logical, distinctive, and mutually exclusive semantic categories with clear meanings.			X	X				X
Order menu choice labels based on the user and task variables.			X	X				X
Either gray out or delete inactive menu items, depending on user experience and the input device.			X	X		X		X
Use menu choice descriptors if choice labels may not be clear and unambiguous.				X				X
Provide menu selection defaults when possible.					X			
Use cursor selection for short menus on keyboard-driven menu systems with casual use. Use mnemonically lettered selection codes for longer menus and/or high-frequency users. Let the user point if there is a pointer.					X	X		
Consider pie menus when the choices lend themselves to a pie format.					X	X		X
Provide menu selection feedback.						X		
Make menu hierarchy shallow and broad, if possible.							X	
Allow context labels, menu maps, and place markers as navigation aids in complex menu systems.							X	
Facilitate backwards navigation.							X	
Distinguish between “Choose one” and “Choose many” menus.				X			X	
Match menu structure to task structure.	This guideline is too vague.							
Allow users to tailor menu structure to task structure.	There is no clear benefit from this guideline.							

Figure 2.3 Menu design guidelines proposed by Mayhew (1992), and menu usage subtasks. The subtasks that are part of the processes of search and selection are indicated at the top. An x indicates that accomplishing the subtask will probably be easier when the guideline is followed.

2.3.2 Experimental Phenomena in Computer Menu Search

This is a discussion of phenomena that have been observed in experiments that involve finding and selecting an item on a computer menus. The review is limited to

experiments that emphasize the visual search component of the task by reducing the task to (a) finding the target and (b) selecting the target. This review will exclude studies that introduce subtasks such as navigating through more than one menu or deciding which menu item provides the best semantic match with the target. Rather, the review will only include research that pertains to the search of a single menu for a known target item.

Most of the menu studies reviewed have an experimental procedure similar to that of the general visual search tasks discussed in the previous section. The experimental procedure is as follows:

1. The participant is presented with a precue of the target menu item on the computer screen.
2. The target precue disappears.
3. The menu appears on the screen, and the clock starts.
4. The participant finds and selects the target item, and the clock stops.

The usual experimental variations include exactly how the participant is precued, what are used as menu items, the ordering of the menu items, the number of items in the menu, the size and spacing of menu items, whether or not menu items are grouped, and how the participant indicates that they have found the target item (usually with a mouse click or a key press). The dependent variable of interest is usually selection time, measured from when the menu appears to when the participant has selected the correct target item. Another important dependent variable is number of errors.

The location and duration of eye fixations during task execution are emerging in the literature as another dependent variable. But because of the large amount of (a) data generated by eye tracking devices, (b) data processing required in order to report these variables, and (c) noise in fixation locations, it is very challenging to record and summarize these data in a manner that reveals fundamental aspects of visual search. Nonetheless, several researchers have braved the flood of data and distilled some interesting observations, including Crosby and Peterson (1991), Aaltonen, Hyrskykari and Räihä

(1998), and Byrne, Anderson, Douglass, and Matessa (1999). The eye movement data reported by Aaltonen et al. and by Byrne et al. are subject to Russo's (1978) criticisms and are reported as "summary statistics, such as fixation frequency, spatial distribution, and mean duration" rather than "aggregated into meaningful cognitive units or examined for interpretable sequential patterns." The data reported by Crosby and Peterson, however, are what Russo seeks—interpretable sequential patterns. All three studies are informative nonetheless, and will be discussed below.

A review of the literature reveals eight basic phenomena that are consistently observed from study to study. There is a close parallel between these phenomena and those that have been observed in more general visual search tasks, which were summarized in Section 2.2.1, "Experimental Observations". These phenomena include:

1. People can find a target more quickly when it appears higher (physically closer to the top) in the menu (Card, 1983; Nilsen, 1991; Perlman, 1984; Somberg, 1987; Somberg, Boggs & Picardi, 1982).

2. People can find a word more quickly in a vertical list than in a horizontal list (Backs, Walrath & Hancock, 1987; Wolf, 1986; Woodward, 1972).

3. People take more time to find a target item in a longer menu than in a shorter menu. This even holds true when the target items are in the same position of two randomly ordered menus (Nilsen, 1991; Perlman, 1984; Somberg et al., 1982).

4. People can find a target more quickly when menu items are alphabetically or numerically ordered than when they are randomly ordered (Card, 1982; Perlman, 1984; Somberg, 1987).

5. With practice, people can learn to find a target in an unordered menu almost as quickly as in a sorted menu (Card, 1982; Somberg, 1987).

6. People can search some kinds of menu items more quickly than others, such as simple icons more quickly than complex icons (Arend, Muthig & Wandmacher, 1987; Bednall, 1992; Byrne, 1993; Landauer & Nachbar, 1985; Perlman, 1984; Vartabedian, 1971; Williams, 1988).

7. When searching unordered menus, people adopt somewhat systematic top-to-bottom scan patterns (Aaltonen et al., 1998; Byrne et al., 1999; Crosby & Peterson, 1991).

8. People do not fixate every item along the way to finding the target (Aaltonen et al., 1998; Byrne et al., 1999; Crosby & Peterson, 1991).

Studies illuminating each of these eight basic phenomena will now be presented.

1. Higher items are faster.

A systematic increase in selection time as a function of the Serial Position of the target item, referred to as a *Serial Position effect*, appears across all positions in randomly ordered menus, but only for the topmost menu items in ordered menus and positionally constant menus.

In randomly ordered menus, Somberg et al. (1982) found a Serial Position effect of 140 msec per position for words and Nilsen (1991) found an effect of about 100 msec per position for single numerical digits.

In alphabetically or numerically ordered menus, Somberg (1987) and Perlman (1984) found a Serial Position effect, but only for items near the top of the menu, and Nilsen (1991) found a logarithmic Serial Position effect.

Card (1983) presented his participants with an initially randomly ordered menu but kept the ordering constant for a total of forty-three trials, resulting in something between randomly ordered and positionally constant menus. Card found a Serial Position effect of only 26 msec per position.

It should be pointed out that in none of these studies did the experimental design motivate the participant to start the search at the *bottom* of the menu, as might be the case with menus that open from the bottom, such as the Microsoft Windows “Start” menu shown in Figure 2.2. For such menus, this observation might not generalize.

2. Vertical is faster than horizontal.

Backs et al. (1987) demonstrated that people can select a target item more quickly in a vertical menu than in a horizontal menu. They presented participants with menus of either one vertical column or multiple horizontal rows, and observed that participants took on average 150 msec less to select a target item in the vertical menu.

Wolf (1986, cited by Tullis, 1988a) observed that using vertical lists alphabetized within each column, rather than using horizontal lists, resulted in 37% less time being required to locate a target.

Woodward (1972) asked participants to compare pairs of 3-digit numbers in different arrangements presented on paper and to respond as to whether or not the two numbers in each pair were identical. He observed that response time was significantly faster when one number was directly above the other (as in vertical menus) than when the numbers were arranged side-by-side on the same line (as in horizontal menus).

3. Longer menus take longer.

This phenomena is not surprising because in a longer menu there will be more menu items that need to be examined. Perlman (1984) observed average selection times ranging from 1.26 seconds for 5-item menus, to 2.23 seconds for 20-item menus. Nilsen (1991) observed average selection times ranging from 0.57 seconds for 3-item menus to 0.99 seconds for 9-item menus.

Researchers have also observed the more surprising phenomenon that, when the target item appears in the same position in two different randomly ordered menus of different lengths, more time is required to select a target from the same position of a longer menu. But this only seems to be the case when the participant can predict the length of the menu in advance, as when experimental trials are blocked by menu length. Perlman (1984) observed that selection time for the same menu position increased by about 0.1 seconds each time the menu length was increased by 5 items. Nilsen (1991) observed that selection time for the same menu position increased by about 0.15 seconds each time the menu length was increased by 3 items. Both Perlman and Nilsen blocked their experiments by menu length. Both Somberg (1982) and Byrne et al. (1999) did *not* block by menu length (hence the participant could not predict the menu length from trial to trial), and they did not find a consistent, significant increase in selection time for the same position in

randomly ordered menus of different lengths.

4. Sorted is faster than randomly ordered.

Card (1982) asked participants to select a target item from a menu of command names presented in different orderings. He observed that, in early trials, participants tended to find the target more quickly when the menu was ordered either alphabetically (0.81 sec) or by function (1.28 sec) than when the menu was ordered randomly (3.23 sec).

Perlman (1984) asked participants to find a word in a list. He presented both sorted and randomly ordered lists, and observed that participants could find a target significantly faster in sorted lists (1.45 sec) than in randomly ordered lists (2.01 sec).

Somberg (1987) presented participants with menus of twenty words that were either alphabetically or randomly ordered, and observed that people performed significantly faster with alphabetically ordered menus.

5. Positional constancy is fastest.

If the positions of menu items are held constant across trials, with practice people can find a target in a randomly ordered menu as quickly as in a sorted menu.

Card (1982) asked participants to select a target item from a menu of command names presented in different orderings. Each ordering was held constant across all trials. He observed that in early trials participants tended to find the target more quickly when the menu was ordered alphabetically than when the menu was ordered randomly. But he also observed that in later trials the relative advantage of alphabetized menus nearly disappeared, and participants could select items almost as quickly in the randomly ordered menu.

Somberg (1987) presented participants with menus of twenty words that were randomly ordered. For some participants, he kept the initially random positions constant throughout the experiment. For other participants, he re-ordered the menu randomly for every trial. He observed that on the first trial people took about the same amount of time to

select the target item (2780 msec), but that after an average of just one presentation, people could select the target more quickly from a positionally constant menu (2090 msec) than from a randomly re-ordered menu (2690 msec). After twenty trials for each position, performance for randomly re-ordered menus stayed relatively constant but performance for positionally constant menus improved (to 1400 msec).

6. Some menu items are faster than others.

When scanning a menu, the nature of the menu items sometimes affects how quickly a person can find the target. For example, people can search through numbers more quickly than words, and some kinds of words or icons more quickly than others. But of course not all features affect search time. Whether items are in upper versus lower case, for example, does not significantly affect search time.

Comparing the search of numbers versus words, Perlman (1984) presented participants with menus of either the Arabic numbers from 1 to 20, or of well known words drawn from Battig and Montague (1969). Perlman observed that, on average, participants selected the numbers (1.56 sec) more quickly than words (1.89 sec). Landauer and Nachbar (1985) presented participants with menus of either Arabic numbers (1 to 4 digits long) or words randomly chosen from an online dictionary (4 to 14 characters long). Participants selected the numbers more quickly than the words.

Comparing different kinds of icons with text-only menus, Arend, Muthig and Wandmacher (1987) observed that people could find a target in a menu of distinctive icons, each with a distinct primary feature, more quickly (1173 msec) than they could find a target in a menu of words (2011 msec) or a menu of representational icons, icons whose meanings are represented in smaller details (2004 msec). Similarly, Byrne (1993) observed that participants could find a target icon more quickly when all icons are simple rather than complex; this is analogous to the “target pop-out” phenomena mentioned in Section 2.2.1.

However, it is not always possible predict which kinds of menu items can be searched faster. It is not clear, for example, how upper versus lowercase text of menu items affects search time. There is empirical evidence (reviewed briefly by Bednall, 1992) that lowercase text can be read more quickly than uppercase text, but Vartabedian (1971) observed that uppercase lists could be searched more rapidly than lower case lists. Other researchers observed no clear advantage to upper or lowercase. Bednall (1992) observed that participants took the same amount of time to search uppercase and mixed case menus. Williams (1988) observed that there was no clear time advantage between upper or lowercase menu items.

7. People adopt somewhat systematic scan patterns.

Researchers have used eye tracking devices to capture eye movements during menu search tasks, and found that saccade scan paths tend to be somewhat systematic (that is, regular and orderly). The systematicity manifested itself in various ways.

Crosby and Peterson (1991) presented participants with a screen containing three columns of randomly ordered three-digit numbers, and asked participants to find a number in the list. Crosby and Peterson categorized the visual scan path used for each trial, and found that the college student participants exhibited a systematic scan path on 66% of the trials. The scan path was usually top-to-bottom, but was sometimes bottom-to-top for the second column.

Aaltonen et al. (1998) presented participants with pull-down menus containing names or concepts grouped by category, and asked participants to find specific targets in the menus. Participants tended to scan menus with a series of upward and downward sweeps—sequences of eye movements in the same direction—that decreased in both duration and length throughout the trial. Downward sweeps were more common that upward sweeps.

Byrne et al. (1999) presented participants with pull-down menus containing

numbers or letters, randomly re-ordered for each trial, and observed that (a) the initial fixation was usually on one of the top few menu items, (b) more fixations were required to find a target lower in the menu, and (c) people tended to not fixate items below the target position.

8. People do not fixate every item.

The same researchers found that people scanning a randomly ordered menu do not fixate every menu item on the way to finding the target, but instead jump over several items at a time.

Crosby and Peterson (1991) found that, after practice, participants were able to find the target item with fewer saccades than would be required if they fixated every item. Crosby and Peterson also report a pictorial representation of one specific scan path that was recorded, and it can be seen that the participant systematically fixated every third or fourth item.

Aaltonen et al. (1998) found an average vertical saccade distance of 2.21 menu items, which suggests that participants did not fixate every item. Though this average saccade distance might have been inflated by saccades in which participants jumped over items to get to the next group, a pictorial representation of two specific scan paths also suggests that participants did not fixate every item.

For menus containing nine or twelve items, Byrne et al. (1999) found that participants made an average of 2.5 fixations before selecting the target when the target appeared in the first position but that, on average, participants made only an additional 0.2 fixations per Serial Position for targets further down in the menu. Hence, for items below the fourth position, people exhibited too few fixations to have fixated all of the items along the way to the target.

Other observations

Researchers have studied other factors that relate to how people search for a known item in a single menu, such as the effects of arranging menu items in groups, (Bednall, 1992; Hollands & Merikle, 1987), removing temporarily disabled menu items completely versus just lightening the text (Francik & Kane, 1987), and varying the distance between adjacent menu items (Nilsen, 1991; Williams, 1988). But the effects of these and other factors are not as clear or are not confirmed by duplicate studies as are the above five phenomena. The list above represents the current understanding of what will be reliably observed when a person selects a known item from a single computer menu.

2.3.3 Proposed Hypotheses for Menu Search

Besides making observations about how people search menus, researchers have proposed hypotheses relating to menu search. This review will only include hypotheses that attempt to explain fundamental underlying principles of human performance that come into play when people search menus. A statement such as "People search vertical menus more quickly than horizontal menus" will not be included as a hypothesis since this is more of a summary of observed data. As in the previous discussion regarding observed data, this review will only include hypotheses that relate to visual search when the target item is known in advance. This review excludes hypotheses that discuss how the search process is affected by processes such as deciding which menu item *best* matches the precued target item. Hypotheses that have been proposed, including several that conflict, are as follows:

1. People use *recognition* memory for menus, and *recall* memory for command line interfaces.

2. People process one menu item at a time and hold their gaze on each item as they consider it.

3a. People search systematically from top to bottom.

versus

3b. People search randomly.

versus

3c. Search is both random and systematic.

4a. People terminate the search when they find the target.

versus

4b. People exhaustively examine every item on the menu before deciding which is the target.

5a. People do not begin the process of selecting a target item until they have found the target.

versus

5b. People overlap the search and selection processes.

Each of these hypotheses will now be discussed in greater detail, including who proposed each one, observational data that supports it, and a brief critique of each hypothesis.

1. People use recognition for menus and recall for command lines.

To support a belief that menu interfaces are superior to command line interfaces, some researchers point out that humans are better at recognition than recall, and theorize that people use recognition for menu interfaces and recall for command line interfaces (for example, Galitz, 1993, p. 209; Mayhew, 1992; Paap & Roske-Hofstrand, 1988). The hypothesis that people use recognition for menus is relevant to this project because it bears on the ecological validity of the assumption that a target menu item can be known in advance. In all of the menu models built in this research project, it is assumed that the person knows the target in advance. It could be argued that people using menus do not recall the target menu item in advance but instead recognize it only when they see it, and thus the models are built on a flawed reproduction of a task. A response to such a criticism would be to point out that the initial distinction—that people use recognition for menus and recall for command lines—is itself flawed.

In classic recognition and recall tasks, a participant is typically given a list of words to learn followed by a test. The test for the recognition task differs from the test for the recall task. In the recognition test, the participant is given a list that contains words from

the original list, as well as distractor (non-target) words, and must identify which words are from the original list. In the recall test, the participant is asked to recall the list with no retrieval cues other than an indication of which list to recall. In terms of proportion of correct responses, people are much better at recognition tasks than recall tasks (Klatzky, 1980).

But it is not clear that these experimental tasks of recognition and recall map to the computer tasks of using menu interfaces and command line interfaces as readily as some researchers would like to believe. Issuing a command to a computer via a menu or a command line is a very different task than merely reporting words that were previously observed, especially if the person using the computer is accomplishing a piece of work and knows the steps required to accomplish the task. Mayes, Draper, McGregor and Oatley (1988) observed that when people were in the middle of accomplishing a task, they could recall menu items needed to accomplish a task without looking at the menu, even menu items they could not recall when asked to list the contents of menus. These results demonstrate that experienced users recall commands based on retrieval cues that come from the execution of the procedural knowledge, and know what they are looking for when they open a menu. There is little or no evidence that experienced users use recognition for menus and recall for command lines.

3. People serially process one menu item at a time.

Researchers have proposed that people conduct a serial, nonoverlapping scan of menu items. These hypotheses are proposed to account for the Serial Position effect, discussed earlier. Norman (1991) proposed that people find the next item, encode it, decide if it matches, and proceed to the next item if the previous did not match. Somberg et al. (1982) and Anderson, Matessa, and Lebiere (1997) made similar proposals. Vandierendonck, Van Hoe, and De Soete (1988) even went so far as to assert that people hold their gaze on each menu item, one item at a time as they consider each item; but this

was evidently not based on analysis of eye movement data. Card (1982; 1983) also assumed that people process one item at a time; this assumption was carried over from the early mathematical models of visual search on which he based his models.

4a. People search systematically from top to bottom.

Researchers have proposed that, to examine a vertical menu, people begin their search at the top of the menu and move their gaze down across successive menu items. Observing a nearly linear Serial Position effect, several researchers concluded that people search menus systematically from top to bottom (Nilsen, 1991; Somberg, 1987; Somberg et al., 1982; Somberg & Picardi, 1983). As well, MacGregor and Lee (1987) interpreted menu selection data presented by Card (1982; 1983) to explain how a sequential search could account for Card's data.

4b. People search randomly.

Other researchers have proposed that when people search a vertical menu, they move their eyes randomly up and down the list. Card (1982; 1983) observed that the cumulative probability of selecting a target item within a given amount of time increases as time increases, and at a rate that can be explained by a random search model and not by one specific systematic model. Other researchers have restated Card's assertion that search is entirely random (such as Giroux & Bellau, 1986; Parton, Huffman, Pridgem, Norman & Shneiderman, 1985) but without offering new rationale or data. MacGregor and Lee (1987) argued that Card's conclusion does not necessarily follow from his observations. There are also other problems with Card's conclusion, which will be enumerated here.

Problem 1: As discussed at the beginning of this chapter, Card's models are derived from the visual search of menus based on models of visual search for enemy airplanes and warships, which assume visual search in an unstructured visual field. Since a menu is a highly structured visual field, these mathematical models do not necessarily

apply to menu search. Even though the trends in the data for both structured and unstructured visual fields are the same, the same models will not necessarily account for both sets of data.

Problem 2: Card's conclusion of random search relies on the assumption that people will consider only one menu item at a time. The original visual search models for search in an unstructured field assume that people consider only one fixation at a time. Card's models carry forward this assumption, but also assume that only one menu item will be processed per fixation, and thus assume serial processing of menu items. But in Card's task, when menu items can fall closely together (about 0.5˚ of visual angle apart), it is possible that more than one menu item could be examined at the same time. So the serial processing assumption may not apply, but his conclusion of random search relies on this assumption.

Problem 3: Card concluded that people searched randomly based in part on observing that, during search, participants tended to make roughly the same number of upward and downward eye movements. But in Card's task the to-be-selected item appeared above the menu at the same time as the menu itself, and at the same time that timing started. The participant did not know the target item in advance and may have spent time and eye movements getting the target item into memory even after beginning the search. In experiments run by other researchers—experiments that gave rise to Serial Position curves with slopes that suggest systematic search—the participant knew the target item before opening the menu and before timing started, which perhaps more closely resembles a real-world menu task in which the participant knows the target in advance. Card's task was perhaps less of a search task and more of a *matching* task in which participants compared menu items to the to-be-selected item. In fact, Aaltonen et al. (1998) ran a menu experiment in which the precue stayed visible after the menu appeared, collected more precise eye movement data, and observed that sometimes the gaze returned to the still-visible precue during search.

4c. Search is both random and systematic.

Perhaps the most plausible explanation is that visual search of menus is both random *and* systematic. This explanation can perhaps be derived from a hypothesis proposed by Williams (1966b), that people use a systematic process with randomness superimposed upon it. Another hypothesis, proposed by Arani, Karwan and Drury (1984) is that people attempt to search in a systematic manner, but forget all of the places where they have already looked, which introduces randomness. Another menu selection model that incorporates both random and systematic search is presented in this dissertation, which was presented in Hornof and Kieras (1997).

5a. People terminate the search when they find the target.

Researchers have proposed that when people know the exact target that they are looking for in advance, they terminate their visual search as soon as they have found the target. Based primarily on the Serial Position effect, Somberg (1982), Somberg and Picardi (1983), and Nilsen (1991) proposed that people terminate their search as soon as they find the target. Eye movement data collected by various researchers seems to confirm this assumption (Aaltonen et al., 1998; Byrne et al., 1999; Crosby & Peterson, 1991).

5b. People exhaustively examine every item on the menu.

Other researchers have proposed that, even when people know the exact target item, they exhaustively examine all menu items before selecting the target.

Lee and MacGregor (1985) offer a general approach for predicting menu selection time across a series of menus that allows for either a self-terminating or exhaustive search strategy within each menu. Their model leaves search strategy as a free parameter, allowing for either a self-terminating or exhaustive strategy. MacGregor, Lee, and Lam (1986) extended their previous model to assert that a self-terminating search occurs when a single menu item is highly likely to be recognized as the target, but that an exhaustive search occurs when no single item is highly likely to be recognized as the target.

MacGregor and Lee (1987) also proposed that even when a person knows the exact target in advance, they might exhaustively examine all items before deciding on the target. But MacGregor and Lee were offering this hypothesis mostly to counter Card's assertion that a self-terminating random search could best explain a flat Serial Position curve. MacGregor and Lee argued that this data could also be accounted for by an exhaustive systematic top-to-bottom search.

6a. Search and selection are independent.

Researchers such as Norman (1991) and Vandierendonck et al. (1988) have proposed that people do not begin the process of selecting the target item (as with a mouse) until after they have found it, but do not offer any empirical evidence for this supposition.

Card (1983) also speculates that the two are independent. In analyzing the menu selection times he observed, he assumes that some of the total selection time is dedicated exclusively to search, and some exclusively to selection. When plotting the selection times, he states that "the intercepts for the regression lines" of roughly 1100 msec can be considered to be "the contribution to the task of the nonvisual search components such as mouse movement and reaction time."

6b. Search and selection are combined.

Other researchers have proposed that people combine the processes of search and selection.

Sears and Shneiderman (1994) proposed a specific, time-consuming combination of the search and selection processes. They propose that as people "move the cursor down the menu to the item of interest, the cursor acts as a visual anchor guiding their search." But they do not discuss whether or not they observed the participants in their study actually moving the mouse in such a manner. They did report selection times as fast as 1.75 sec for the 13th item on the list, which suggests that their participants were not limiting their eye

movements (which are fast by nature) by their mouse movements (which are slow by nature).

Anderson et al. (1997) proposed a menu selection model that contained a similarly time-consuming and implausible combination of search and selection. To account for Nilsen's (1991) menu data, the model assumes that "Subjects tend to move the mouse down as they scan for the target" and that no mouse movement is required once the target is located. This theoretical assumption is not supported by the raw selection times, which are always under 1.7 seconds even for randomly ordered menus of nine items. Selection times are just too fast for participants to have waited until they completed each mouse movement before making their next eye movement.

It is doubtful that people constrain their eye movements to be as slow as their hand movements in a high speed visual search task, and there is no evidence to support this hypothesis. Since people can move their eyes and their hands independently and in parallel, it is reasonable to expect that they would find a way to use this ability to optimize performance.

Though these fast selection times for randomly ordered menus indicate that people do not move the cursor and their gaze to menu items at the same time, this does not mean that search and selection are completely independent. A reasonable conclusion is that at a higher level of analysis, the search and selection processes can be thought of as independent, but at a lower level of analysis, the two are combined.

2.3.4 Many Conflicting Hypotheses

Researchers have proposed many hypotheses about the details of how people select a known item from a menu on a computer screen. These hypotheses are generally based on interpretation of data, so sometimes different researchers have proposed conflicting hypotheses even from the same data. Usually, just one or two of the above hypotheses are proposed at a time, and the others are held as (sometimes unstated) assumptions, even if

the assumptions have not been generally accepted. As a result, many of the conflicts are still unresolved, and the field of HCI does not have a good theoretical understanding of the cognitive processes that people use when they select an item from a menu.

This review of previous literature regarding menu selection indicates that, though much research has investigated menu selection, many questions remain open. As recently as 1993, in fact, a review of the literature on visual search in human-computer interfaces (Scott, 1993) stated that "Little is yet known about user search processes in menu retrieval."

2.4 Cognitive Modeling

Cognitive modeling is the study and the process of building simulations of human performance which are referred to as *cognitive models*. Calling the simulations *cognitive* models can lead to some confusion because *cognition* generally refers to central strategic and decision processing, and is thought to be separate from the peripheral processes of *perception* and *action*. Action is also referred to as *motor* processing. In this dissertation, cognitive models are assumed to account for all three: perceptual, cognitive, and motor processing.

Cognitive models permit aspects of user interfaces to be evaluated for usability by making predictions based on task analysis and established principles of human performance (Card, Moran & Newell, 1983; John & Kieras, 1996). Cognitive models can predict task execution time based on specification of interface and task, and thus reduce the need for user testing early in the development cycle. They can reveal underlying strategies that people use to accomplish a task, and thus help designers build interfaces and interaction techniques that better complement the strategies. Cognitive modeling has demonstrated its usefulness in the design and analysis of interfaces through simple KLM models as well with more complex GOMS models.

An emerging form of cognitive modeling that shows great promise for the design

and analysis of interfaces is *computational* cognitive modeling, in which computer programs are written to simulate the various perceptual, cognitive, and motor processes involved in accomplishing a task. One form of computational cognitive modeling is GOMS modeling using a computer-based interpreter such as GLEAN (Kieras, Wood, Abotel & Hornof, 1995; Wood, 1993). Another is building models with a *cognitive architecture* such as EPIC (Kieras & Meyer, 1997; Meyer & Kieras, 1997a; Meyer & Kieras, 1997b), Soar (Laird, Rosenbloom & Newell, 1986; Newell, 1990), EPIC-Soar (Chong, 1998a; Chong, 1998b), ACT-R (Anderson, 1993; Anderson et al., 1997), or ACT-R/PM (Anderson & Lebiere, 1998). A cognitive architecture is a computational framework for building cognitive models that simulates and constrains fundamental aspects of human performance.

The cognitive models presented in Chapter 3 and 4 of this dissertation were built using the EPIC cognitive architecture. EPIC is particularly well-suited for modeling visual search because (a) it has a particularly well-developed set of constraints regarding visual and ocular-motor processing and (b) it uses a *multi-match, multi-fire* production system, which means that the cognitive processor can match and fire any number of production rules in a single cycle.

EPIC's constraints regarding visual processing include that visual stimuli are less perceptible when they appear further from the foveal region of the retina. EPIC's ocular-motor processing includes simulating the preparation and execution of physical eye movements. These are important aspects of human performance that will constrain the search space of models that can be used to explain visual search data.

Other architectures vary with respect to how well the architectures themselves account for these aspects of visual and ocular-motor processing. Though Soar does not incorporate visual perceptual and ocular-motor mechanisms at the architectural level, the architecture has nonetheless been used to model some aspects of visual performance entirely through the creation and application of Soar operators (Wiesmeyer, 1992).

EPIC-Soar of course incorporates visual perception and ocular motor processing at the architectural level because it incorporates these components from EPIC. But since the models in this dissertation do not model learning, the learning that is afforded by the Soar component of EPIC-Soar is not needed.

The visual interface of ACT-R and ACT-R/PM partially accounts for these aspects of visual and ocular-motor processing. ACT-R and ACT-R/PM's visual interface assumes that visual features are available no matter where they appear in the visual field, but that conjunctions of an object's features can only be formed when the object is captured in the "spotlight of attention." The visual interface also assumes that time is required to shift attention from one location to another. The spotlight of attention, however, is not linked to any physical zones on the retina, and it is unclear whether or not the simulated attentional shifts are intended to represent physical eye movements.

A multi-match, multi-fire production system such as that used by EPIC is preferred because it means that the cognitive processor will not constrain the overlapping of perceptual and motor processing that is likely to occur during a high speed visual search. Soar is also a multi-match, multi-fire production system. ACT-R and ACT-R/PM are also multi-match, but single-fire. As argued by Meyer and Kieras (1997a; 1997b), people do not seem to have a central cognitive processing bottleneck of this sort, and so the architecture should not impose this constraint. A multi-fire production system, for example, allows one subset of production rules to move the eye throughout the visual scene as quickly as possible while another subset of rules, in parallel, keeps track of whether or not the target has yet been seen.

2.5 Conclusion

Visual search is a very important human activity for accomplishing many tasks, from survival in war to day-to-day computer usage. Much work has been done to collect data and build models to try to predict and explain how people accomplish these tasks.

There are large bodies of work studying general visual search in cognitive psychology, vision research, and human factors. Visual search of computer menus has sparked debate in the human-computer interaction literature for decades. And yet, there are no empirically validated models that explain the perceptual, cognitive, and motor processes involved in visual search. Accounting for the perceptual and motor processing involved is critical because these peripheral processes will constrain the assumptions that can be made about the central cognitive processing involved. Building cognitive models using an existing computational cognitive architecture such as EPIC holds great promise for accounting for these basic processes and for the data.

CHAPTER 3
INTRODUCTION TO THE MODELS

This dissertation presents the first empirically validated models of the perceptual, cognitive, and motor processes involved in the visual search of computer menus. This chapter discusses the basic components of these models: the cognitive architecture used for the model-building, the human experiment that the models are based on, the empirical data collected during the experiment, and the specific inputs to the architecture that are used to model the data.

3.1 The EPIC Cognitive Architecture

The EPIC (Executive Process-Interactive Control) cognitive architecture (Kieras & Meyer, 1997; Meyer & Kieras, 1997a; Meyer & Kieras, 1997b) provides a general framework for simulating a human interacting with his or her environment to accomplish a task, and is well-suited to model visual search. EPIC resembles the Model Human Processor (Card et al., 1983), but differs in that EPIC is a precise computational model, has a programmable production-rule cognitive processor, and incorporates more specific constraints synthesized from human performance literature.

EPIC consists of a production-rule cognitive processor and perceptual-motor peripherals. To model human performance aspects of accomplishing a task, a cognitive strategy and perceptual-motor processing parameters must be specified. A cognitive strategy is represented by a set of production rules, as in CCT (Bovair, Kieras & Polson, 1990), ACT-R (Anderson, 1993; Anderson et al., 1997), ACT-R/PM (Anderson & Lebiere, 1998), and Soar (Laird et al., 1986; Newell, 1990) represent procedural

knowledge. The simulation is driven by a description of the task environment that specifies aspects of the environment that would be directly observable to a human, such as what objects appear at what times, and how the environment changes in response to EPIC's motor movements.

EPIC models are *generative*. When all the necessary inputs for a model are provided to the architecture and the model is run, EPIC interacts with the task environment and generates (a) a specific sequence of perceptual, cognitive, and motor activities to accomplish the task and (b) a prediction of the time required to accomplish the task.

EPIC takes as its input:

- The cognitive strategy for accomplishing a task, stored as production rules in the production memory.
- The availability of each object property for each visual zone, to represent human perceptual capabilities.
- Details of the task environment, such as when and where objects appear.

EPIC generates as output:

- A detailed trace of the flow of information and control among the various processors and memories.
- The time required to execute the task.

As shown in Figure 3.1, information flows from the task environment into sense organs, through perceptual processors, to a cognitive processor (consisting of a production rule interpreter and a working memory [WM]), and finally to motor processors that control effector organs that interact with the task environment. All processors run independently and in parallel. Information processing and motor movement times are held constant across models, and are based on human performance literature.

A single stimulus in the task environment can produce multiple inputs into a perceptual processor, which can be deposited in WM at different times. First to arrive in WM is the detection of a perceptual event, followed later by properties that describe the event. The perceptual processors are "pipelined." If an object's properties begin moving

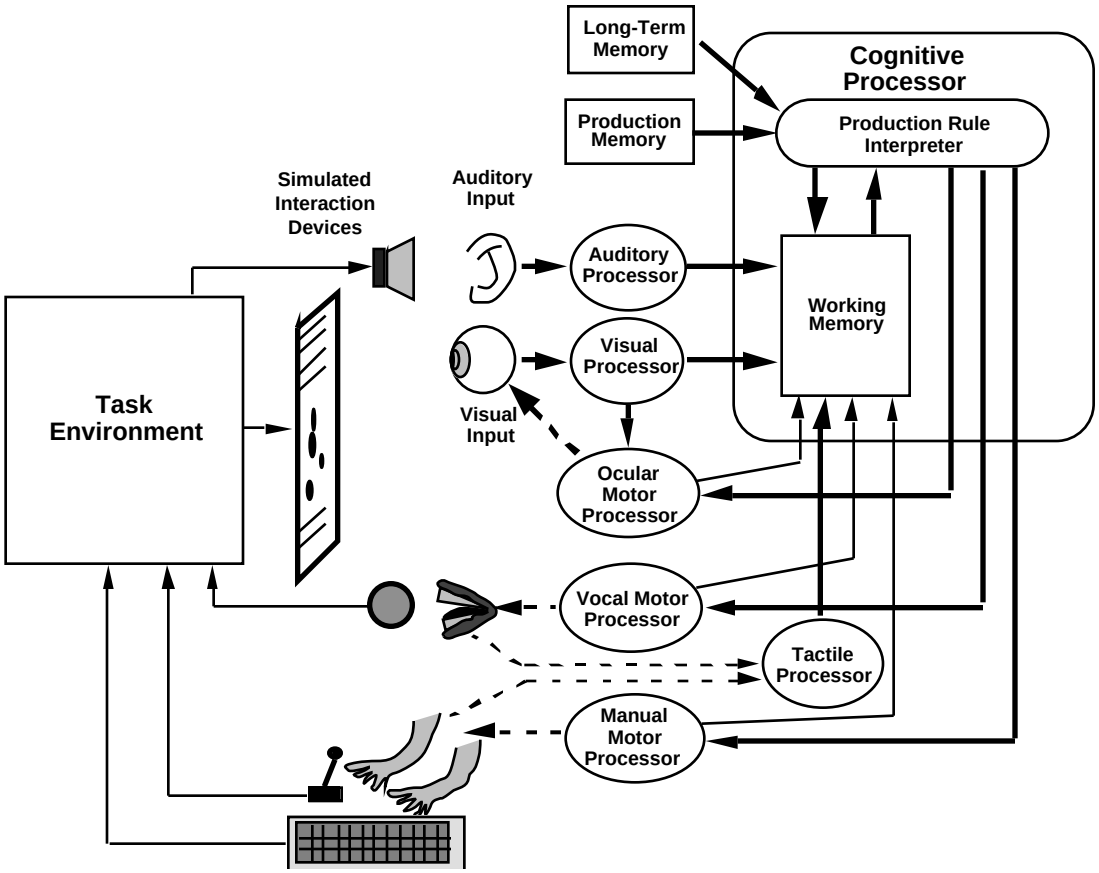

Figure 3.1. Overview of the EPIC architecture, showing flow of information and control. The processors run independently and in parallel. (From Kieras & Meyer, 1997.)

to WM, the arrival of those properties will not be delayed by any other processing. WM contains these items deposited by perceptual processors, as well as control information such as the current task goal. At the end of each simulated 50 msec cycle, EPIC fires all of the production rules whose conditions match the current contents of WM. EPIC allows for parallel execution of production rules in the cognitive processor, and some parallelism in each motor processor.

EPIC is particularly well-suited for modeling aspects of visual search because EPIC, in effect, has eyes that see. EPIC has an ocular motor processor that moves a simulated gaze to locations in the environment as specified by the executing strategy. The gaze can be fixed at one point in the task environment at a time. Time is required to prepare

and execute an eye movement to a new location. Retinal zones are defined as concentric circles around the center of the gaze, with sizes typically set as follows: The bouquet (a radius of 0.25˚ of visual angle), the fovea (1˚), the parafovea, (7.5˚), and the periphery (60˚). These zones are depicted in Figure 3.2.

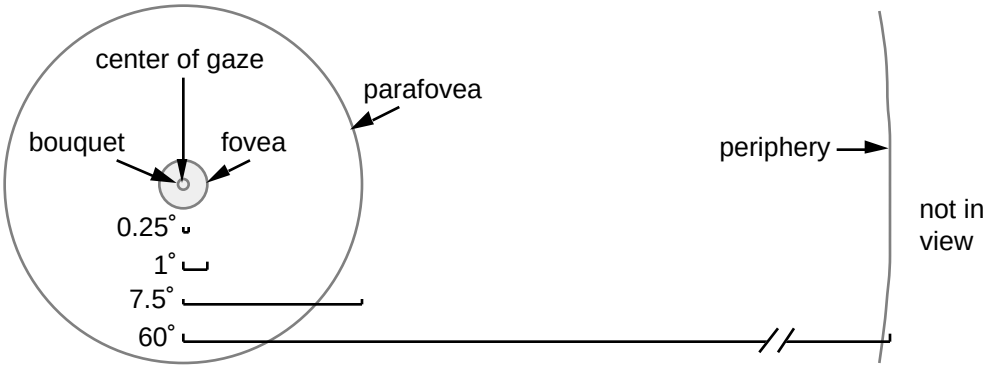

Figure 3.2. The retinal zones defined in EPIC, and typical sizes used in modeling. Sizes are the radii in degrees of visual angle. The visual stimulus " " appears outside of the bouquet and fovea, but inside of the parafovea and periphery.

Recall that one of the inputs to an EPIC model is the availability of object properties. These parameters define (a) which properties are available in which zones, and (b) the delay imposed on each property in each zone. The availability of each property of an object is a function of in which zone the object is located during the current fixation. Thus, there is a different availability and delay setting for each property for each zone (bouquet, fovea, parafovea, periphery, and not-in-view). The availability and delay for the *location* and *text* properties, for example, are typically set as follows: The location property will be available as long as the object is in view, and will take 50 msec to travel from the retina to the visual processor. The text property will only be available when the object falls within the fovea, and will take 100 msec to travel from the retina to the visual processor.

Thus, if during the execution of a model a box containing the text "GO" suddenly appears in the periphery, its location will be available in WM shortly thereafter, but not its

text. An eye movement to that location can then be executed. Once the "GO" box appears in the fovea, the text will also make its way into WM after a delay.

Setting these zones with fixed boundaries is clearly a simplification in the architecture, but it is a profound architectural feature nonetheless. The feature allows the availability of object properties to decrease gradually as the objects appear further from the center of the gaze. This is a well-documented aspect of human performance that is not present in other cognitive architectures such as ACT-R, ACT-R/PM, and Soar, but one that needs to be incorporated into models of visual search because it will constrain the strategies and ocular motor processing that can be used to explain and predict data. EPIC-Soar (Chong, 1998a; Chong, 1998b) of course uses the same visual zones because EPIC-Soar directly combines EPIC's perceptual and motor processors with Soar's cognitive processor.

Location information can also be made available to the cognitive processor by defining *named locations* for a particular task environment. Named locations represent knowledge of fixed locations in visual space, task knowledge that has been gained with task experience.

To act upon the environment, a production-rule strategy sends motor commands to the various motor processors. These motor commands specify a movement in terms of its *style*, as well as other characteristics such as direction and extent. Predefined manual movement styles allow EPIC to point with a mouse (the POINT style), press a mouse button (PRESS), point with a mouse while holding down the mouse button (POINT-PRESSING), and release a mouse button (RELEASE).

There is an assumption in EPIC that with practice a person can combine two or more consecutive motor movements into one *compound* movement, all the submovements of which can be prepared in advance. *Compound* movement styles combine multiple movements into a single command. For example, the PUNCH compound movement style executes a PRESS and RELEASE with a single command. A PUNCH of a mouse button

is more commonly referred to as "clicking" the mouse button. It is called PUNCH, however, because it is an EPIC movement style for interacting with any button.

A motor movement must be *prepared* and then *executed*. Movement preparation time will be reduced if the previously executed movement had any identical features. A POINT, for example, has four features: movement style, hand (left or right), direction, and distance. The standard 200 msec to prepare a POINT will be reduced to zero if all four features are identical to those of the previous manual motor command. Execution time represents the time required for mechanical muscular movements in the physical world, and is thus determined in part by features such as the distance that an effector must travel. Motor movement styles and their associated timing functions and parameters are based on the human motor control literature (see Rosenbaum, 1991). Execution time for a mouse point, for example, is determined by the Welford version of Fitts' law (Card et al., 1983), with a minimum execution time of 100 msec enforced:

$$T = \max\left(100,\ K \cdot \log_2\left(\frac{\text{Distance}}{\text{Width}} + 0.5\right)\right) \text{ msec}$$

For a POINT movement, the coefficient K is set to 100, as given in Card et al. (1983). For a POINT-PRESSING movement, the coefficient K is set to 140; this value is derived from data reported by Walker, Meyer & Smelcer (1993). These two values allow EPIC to account for the button-depression effect discussed by Nilsen (1991).

In short, EPIC is applied to a task as follows: The production-rule strategy directs the eyes to objects in the environment. The eyes have a resolving power which determines the processing time required for different object features, such as location and text. When information needed to determine the next motor movement arrives in WM, the strategy instructs the ocular motor and manual motor processors to move the eyes and hands.

This provides a cursory overview of the EPIC cognitive architecture. A more thorough description of EPIC is presented by Kieras and Meyer (1997) and Meyer and Kieras (1997a; 1997b; 1999)

All of the models for this dissertation were built using the EPIC cognitive architecture. There are many reasons for building models using an existing architecture such as EPIC, including (a) the models contribute to the validation and thus evolution of a unified theory of cognition (Newell, 1990), (b) the architecture represents a theory, so the models build on existing theory, and other researchers can more readily understand and evaluate basic assumptions of the models, and (c) the model builder can get something interesting up and running much more quickly.

3.2 The Modeled Task

The menu selection task modeled in this dissertation was designed by Nilsen, who presented the task to human participants in an experiment (Experiment 2 in Nilsen, 1991). This study is in some ways typical of the menu studies discussed in Chapter 2, but it also has some unique features that make the experiment and its data particularly useful for studying the perceptual, cognitive, and motor processes of visual search and response selection. First, this experiment isolates a subset of the processes required in a "real world" menu task. The task is not confounded with more complex processes of reading, comprehension, judgment, decision making, and problem solving. Second, Nilsen varied menu length and reported selection time as a function of the Serial Position of the target menu item. Few researchers have reported such data but, as will be seen in Chapter 4, this combination was critical for revealing search strategy.

A description of the experimental procedure and observed data follows.

3.2.1 Procedure

In his experiment, Nilsen presented participants with pull-down menus of three, six, and nine menu items. Menu items were the numerical digits from 1 to *n*, where *n* was the length of the menu. Menu items were either randomly re-ordered for each trial or presented in numerical order. Trials were blocked by menu length and ordering. Menus

always appeared at the exact same location on a computer screen. The distance between menu items was roughly 0.2 inches. The distance from eye to screen was neither controlled nor measured, but was probably between fifteen and thirty inches for each participant.

As shown in Figure 3.3, each trial consisted of the following steps: Using a mouse, the participant moved the cursor to the GO box, which caused the precue of the target item to appear above the GO box. The participant clicked the mouse button. The GO box and precue disappeared, the menu appeared, the cursor was automatically positioned one pixel above the first menu item, and the clock started. The participant used the mouse to move the cursor to the target item in the menu. The participant clicked the mouse button. The clock stopped.

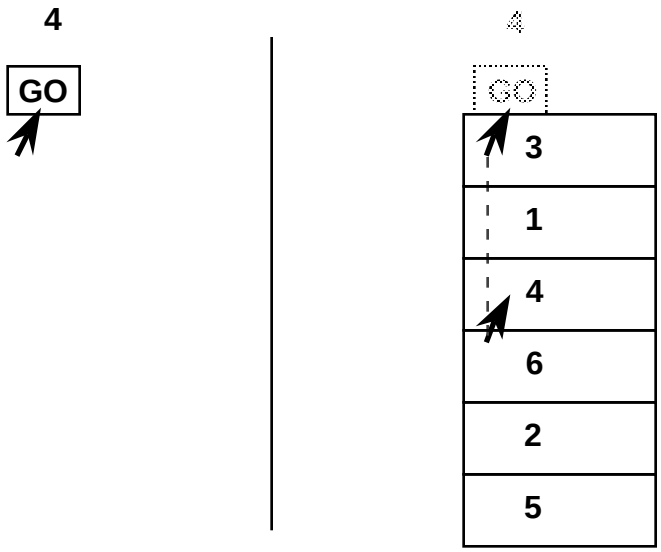

Figure 3.3. Nilsen's menu selection task with a randomly ordered menu and six items in the menu.

Two different menu styles were used: walking and click-open. With walking menus, participants moved the cursor to the GO box, pressed *and held down* the mouse button, moved the cursor to the target while keeping the mouse button depressed, and then released the mouse button. With click-open menus, participants moved the cursor to the

GO box, clicked the mouse button, moved the cursor to the target, and then clicked the mouse button. Within a block, all menus were of the same style.

Eight experienced mouse users participated in the experiment, and were financially motivated to perform each trial as quickly as possible. Nilsen presented each participant with eighteen trials for every possible combination of target position, menu length, menu ordering, and menu style (walking versus click-open). The final fifteen asymptotic trials are reported in the data.

3.2.2 Results

Figure 3.4 shows Nilsen's observed data for randomly and numerically ordered menus, averaged across participants, blocks, and menu style (walking versus click-open), as well as the time required to move the mouse to each position as predicted by the Welford form of Fitts' law (see Card et al., 1983, Ch. 2) with a coefficient of 120. Figure 3.5 shows the same data, but collapsed by menu length and expanded by menu style (walking versus click-open).

The important features in the randomly ordered menu data include:

- *Menu length effect.* When the target item is in the same Serial Position across menus of different lengths, shorter menus are faster.
- *Serial Position effect.* Selection time increases with a fairly linear slope of about 100 msec per item. As can be seen in the graph, the mouse movement time predicted by Fitts' law cannot entirely account for this slope either in shape or magnitude.
- *Position 1 effect.* The selection time for Serial Position 1 is a little higher than the selection time for Serial Position 2.

The important features in the numerically ordered menu data include:

- *Faster than random.* Participants select an item from a numerically ordered menu substantially faster than from a randomly ordered menu. The same model will probably not be able to account for both random and numerically ordered menu data. Evidently, less extensive visual search is needed for the numerically ordered menus.
- *Very fast.* Participants select the target item from numerically ordered menus very quickly, requiring only 350 to 950 msec to click on the GO

box, move the cursor to the target, and click on the target.

- *No menu length effect*. In the numerically ordered menu data, every Serial Position takes the same amount of time regardless of the menu length.

- *Diminishing Serial Position effect*. There is a negatively accelerated increase in the numerically ordered menu data; the increase is greater than that of the Fitts' law prediction also shown on the graph.

The important feature for both randomly and numerically ordered menus that is illustrated in Figure 3.5 is the *button depression effect*: People move the mouse more slowly when the mouse button is depressed, as is the case with walking menus.

The above features in the data will help to guide the construction of cognitive models in Chapter 4. All of the proposed models will be evaluated in terms of how well the models' predictions match Nilsen's observed data.

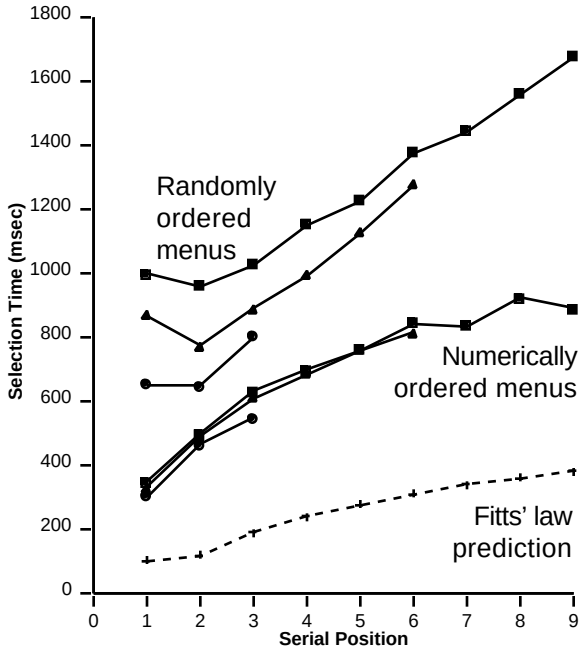

Figure 3.4. Nilsen's observed data for randomly and numerically ordered menus. Mean selection times as a function of Serial Position of target item, for menus with three (●), six (▲), or nine (■) items. Also: Time required to move the mouse to each Serial Position as predicted by Fitts' law with a coefficient of 120. (From Experiment 2 in Nilsen, 1991.)

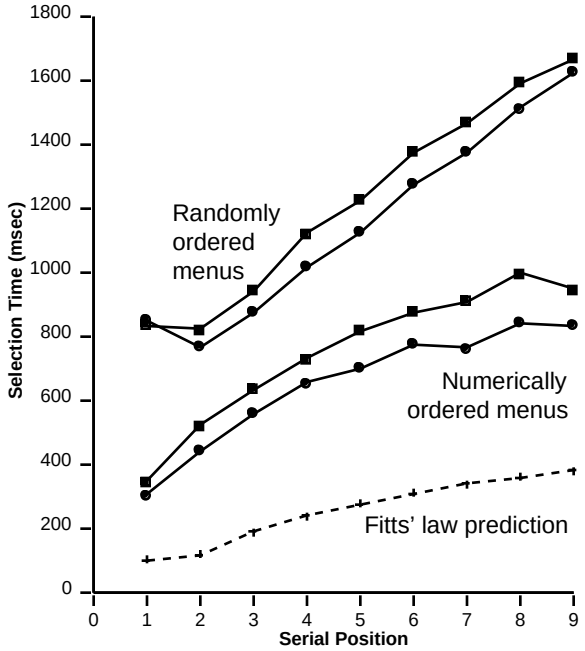

Figure 3.5. The same data as in Figure 3.4, but collapsed by menu length and expanded by menu style (walking ■ versus click-open ●).

3.3 Inputs to the Architecture

All of the components that were required for the models to run using EPIC are discussed in this chapter. These components include (a) the simulation of the task environment, (b) the perceptual encodings, (c) minor enhancements to the architecture, (d) named locations, and (e) task strategies.

3.3.1 The Task Environment

Though ultimately it is the perceptual, cognitive, and motor processing that is of interest in a cognitive model, there is a tight coupling between the perceptual-motor systems and the task environment in a visual search task. Hence, Nilsen's task environment must be simulated precisely to provide EPIC an accurate "physical" world with which to interact.

The task environment is basically responsible for two things—displaying stimuli and responding to actions. The stimuli that must be displayed in Nilsen's experiment include: (a) the cursor, (b) the GO box, (c) the precue, and (d) each of the individual menu items. The responses to actions include:

1. When the mouse is moved, move the cursor.
2. When the cursor is moved into the "GO" box, display the precue.
3. When the mouse clicks inside the "GO" box, remove the "GO" box, remove the precue, start the clock, and display the menu.
4. When the target item is clicked, stop the clock, record the time, remove the menu, and display the "GO" box.

The task environment was programmed in LISP to replicate Nilsen's experiment. It displays all of the appropriate stimuli at the correct time and location, and responds to EPIC's motor activities in the same manner as the experimental software would have responded to the participants' mouse points and clicks.

A visual depiction of the physical environment of an executing model appears in EPIC's Visual Space window (see Figure 3.6). This view is helpful for verifying correct interaction between EPIC and the task environment.

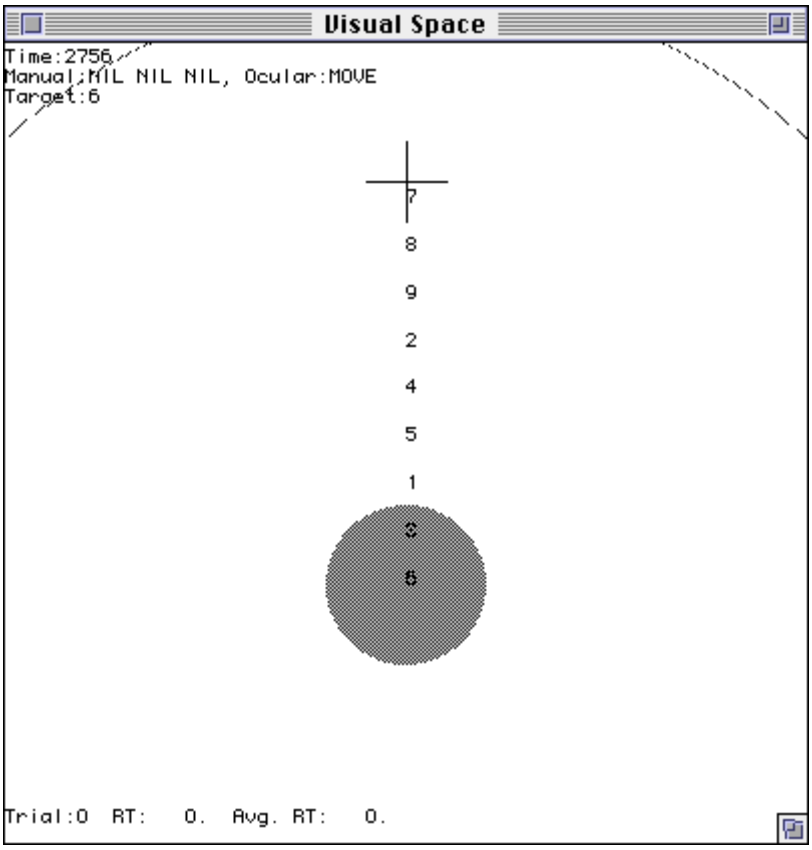

Figure 3.6. EPIC's Visual Space window shows the objects in the task environment and where EPIC is looking during model execution. Shown here is the task environment for the Nilsen menu selection experiment with an eye-to-screen distance of 24 inches. The mouse cursor, represented by the crosshairs, is sitting where it clicked on the "GO" box. The gray circle is EPIC's fovea, indicating that EPIC's gaze is currently on the bottom menu item. The borders of the parafovea can be seen in the top left and right corners.

The task environment simulates the creation of *physical* objects in the *physical* visual space. A separate physical object is defined for the cursor, the GO box, the precue, and each of the individual menu items. Each object is given a unique name, such as *physobj29*. The task environment assigns *properties* to each physical object. Each property is assigned a *value*. The object-property pairs used in the Nilsen models will now be enumerated.

Every physical object is assigned a *location*, which is a Cartesian (*x*, *y*) coordinate that corresponds to that object's location on an imaginary stationary grid in physical space. Every physical object is also assigned a *zone* property with a value that identifies the

physical location of the object's image on the retina; the value can change when the eye or the object moves.

The cursor object also has a *shape* property that identifies it as the cursor. It also has a *points-to* property with a value that is the name of the physical object currently under the cursor, or *nil* if the cursor points at nothing; this represents the physical relationship between the cursor and the object under the cursor.

The remaining objects are all text items, and are assigned a *text* property, set to "GO" for the GO box, and set to a numerical digit for the precue and menu items. Text objects are also assigned an *in-menu* property, set to *yes* for items in the menu, and *no* for all other items; this represents physical clues would distinguish the "GO" box and precue from items in the actual menu.

Every menu item object also has additional properties that represent its physical position in the menu in relationship to the other menu items. The *is-above* property names the menu item that is below the object, and *is-below* property names the menu item that is above the object. The *is-below* of the top item and the *is-above* of the bottom item has a value of *nothing*.

All objects that will be selected with the mouse also have a *size* property that is used to predict the time required for a mouse movement to the object.

After being deposited into *visual physical space*, physical objects and properties are perceived by EPIC's perceptual processors and transformed into *psychological* objects. The exact perceptual encodings of physical object properties are defined by the *visual encoding function* in the visual perceptual processor. The exact parameters used in the *visual encoding function* for the Nilsen models will be discussed next.

3.3.2 The Perceptual Encodings

For every *physical* object that enters EPIC's simulated eyeball, a corresponding *psychological* object is created. First, a psychological *sensory* property is created for

every physical property as a function of the availability of that property for the zone in which the object appears. The availabilities and delays used in the models are listed in Table 3.1. These parameters were established and validated in other EPIC models that involved reading text from computer screens (Kieras & Meyer, 1997; Kieras, Wood & Meyer, 1997).

After each psychological sensory property is created, it is then recoded into a psychological *perceptual* property as a function of the recoding defined in the *visual recoding function*. The visual recoding function gets called every time an object appears, disappears, or moves from one retinal zone to another. It gets called once for every sensory property of that object, and receives as input (a) the time tag (after the zone property delay), (b) the psychological object name, (c) the sensory property name, and (d) the property value. The visual recoding function uses these inputs to create perceptual properties that are then deposited into visual WM.

In the visual recoding function for the Nilsen models, most of the sensory properties are passed through and encoded into perceptual properties "as is", with no additional delay. This is the case for *shape*, *in-menu*, *is-above*, and *is-below*. Other

Object Property	Availability of property when object is... In bouquet	In fovea	In parafovea	In periphery	Not in view
Used by all items					
LOCATION	50	50	50	50	—
ZONE	50	50	50	50	50
Used by cursor only					
SHAPE	100	100	—	—	—
POINTS-TO	50	50	50	—	—
Used by all text items					
TEXT	100	100	—	—	—
Used by menu items only					
IN-MENU	50	50	50	50	—
IS-ABOVE	51	51	51	51	—
IS-BELOW	51	51	51	51	—

Table 3.1. The availability of physical object properties in the menu models as a function of the retinal zone in which the object appears. A number indicates the delay (in msec) imposed by retinal filtering. A dash indicates the property is not available in that zone.

sensory properties, however, trigger more elaborate recodings.

The sensory property *text* is recoded into the perceptual property *label*, with an additional *text-recoding-time* delay of 100 msec; the renaming of the property and the delay represent the mental effort required to transform each physical piece of text into something meaningful to the cognitive processor.

The sensory property *points-to*, used only for the mouse cursor, enters the visual recoding function with a value that is either *nil* or the physical object currently pointed to by the cursor. A value of *nil* gets passed through directly, but a physical object is recoded as the corresponding *psychological* object.

When the recoding function is called with the sensory property *zone*, the perceptual property-value pair *fovea yes* or *fovea no* is created for the object. This represents a theoretical assertion that, when visually searching, a person can distinguish whether or not each object is in the fovea. This property is used by some strategies to saccade to items outside of the fovea, and represents a perceptual feature that contributes to direct moment-to-moment control (Prinz et al., 1992), a saccade programming scheme in which perceptual features gathered during one fixation are used to program the features of the very next saccade.

The sensory properties *zone* and *location* are also used to encode a *global perceptual property* named *next-sweep-item*. A global perceptual property resides in visual WM but is not attached to a specific psychological object. The *next-sweep-item* property is assigned the name of the next psychological object that must be fixated in a *maximally efficient foveal sweep* of the menu. This recoding represents a theoretical assertion that people have the ability to move their gaze down the menu and foveate all menu items with as few saccades as possible, and that the sweep is carried out by selecting the next appropriate menu item as quickly as possible after the previous saccade, another example of direct moment-to-moment control. Figure 3.7 depicts such a sweep.

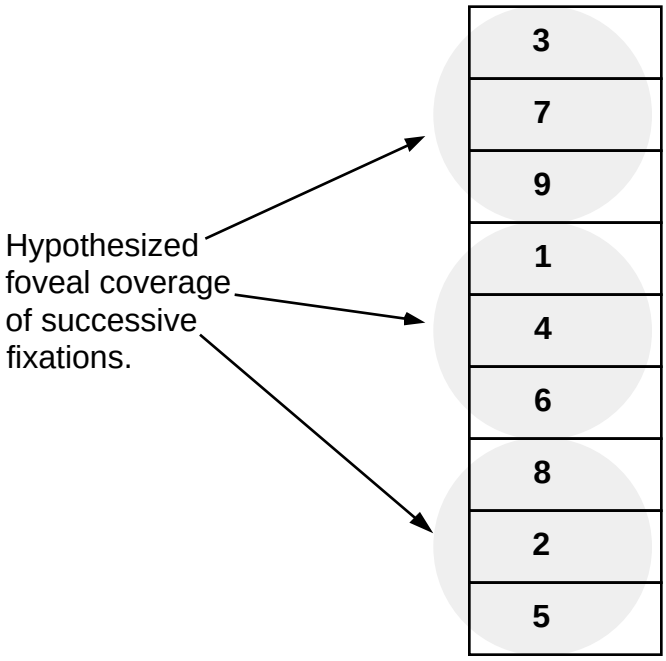

Figure 3.7. A maximally efficient foveal sweep. The models make the theoretical assertion that a person can make a chain of eye movements that will sweep a column of visual objects with as few fixations as possible, and yet insure that every object will be captured by the fovea during a fixation.

A maximally efficient foveal sweep requires a specific interaction between perceptual availabilities and search strategy. As soon as a saccade is completed, the visual recoding function must compute a new next-sweep-item, a computation that is assumed to require 100 msec. As soon as the new location is available, the strategy must command an eye movement to the new location. The processing results in an intersaccade latency of 200 msec, which fits well with what is typically observed for a rapid succession of saccades (Russo, 1978).

3.3.3 Enhancements to the Architecture

Two small enhancements are made the EPIC architecture to model the Nilsen data. They are *visual recoding after saccade* and a *click-and-point compound movement*. The first is introduced to model Nilsen's randomly ordered menu data, and the second to model Nilsen's numerically ordered menu data.

The first enhancement, *visual recoding after saccade*, is required to implement direct moment-to-moment control, in which the direction and extent of the next saccade are programmed using information gathered after the completion of the previous saccade. Such control is required for the maximally efficient foveal sweep described above. To facilitate the sweep, a new value for the global property *next-sweep-item* is computed after every saccade. To trigger this computation, the visual perceptual processor must be notified whenever a saccade has occurred. Though research has demonstrated that the human visual system is notified when an eye movement occurs (Rosenbaum, 1991), this notification had not yet been incorporated into the EPIC architecture.

EPIC was thus modified to call the new *visual recoding after saccade* function following the execution of every ocular-motor *move* command. For these models, the function then triggers the computation of a new *next-sweep-item*, which allows the models to make a maximally efficient foveal sweep.

The second enhancement to the EPIC architecture is a *click-and-point* compound movement. As discussed in Chapter 3, there is an assumption in EPIC that with practice a person can combine two or more consecutive motor movements into one *compound* movement, all the submovements of which can be prepared in advance. Several compound movements are already defined in the architecture. The new *click-and-point* compound movement allows the preparation for a *point* to be combined with that of an immediately preceding *click*. Such overlapped preparation would be possible when a person decides to click and point, and knows the destination of the point before clicking.

The specific modifications to the architecture to implement the *click-and-point* compound movement are as follows: (a) The existing *point* movement style is modified to allow a *point* to begin during the release of a mouse button rather than waiting for its completion. (b) The existing *point* and *point-pressing* movement styles are modified to require only 150 msec of preparation rather than the usual 200 msec if the movement was preceded by a *press* or *punch.*

The *click-and-point* modifications are exploratory; a more complete representation would be to introduce a completely separate movement style to the EPIC motor processor. These modifications will only be used in the final models for the numerically ordered menus.

3.3.4 Named Locations

EPIC uses *named locations* to represent the notion that people can learn a location and initiate a movement to that location even if no physical object is currently visible at that location. The Nilsen models employ three named locations: *first-fixation-location*, *target-location-correct*, and *target-location-with-error*.

First-fixation-location is used to program the initial eye movement in the visual search of the randomly ordered menus. It is assigned one of three locations: the position of the first or second menu item, or the position of a randomly-chosen menu item.

Two named locations are used by the numerically ordered menu models. They represent the notion that, when the exact same menu items appear in the exact same locations trial after trial, people can learn the location of every item, and they can thus anticipate the target location before the menu actually appears.

The first of these, *target-location-correct,* is used to explore the possibility that people can *exactly* predict where the target will appear. At the beginning of every ordered menu trial, this named location is set to the exact location where the target will appear.

The second, *target-location-with-error,* is used to explore the possibility that people can *approximately* predict where the target will appear, that they sometimes get it wrong, and that the error will be greater for items lower in the menu. To introduce this variable error, the vertical coordinate of *target-location-with-error* is perturbed at the beginning of every trial. The perturbations are normally distributed with a mean that is the correct vertical position, and with a standard deviation

$$\sigma = e \cdot d$$

where *e* is a constant error coefficient, and *d* is the distance from the GO box to the target. Thus, the lower the target on the menu, the greater the error in *target-location-with-error*.

Abrams, Meyer, and Kornblum (1989) observed just such a linear relationship between target distance and the standard deviation of endpoints in initial eye movements directed at a single target that is peripherally visible before the start of the trial. A value of $e = 0.04$ provides a very good fit with the data presented in Abrams et al. (1989). Though Abrams et al. were explaining error in eye movements to visible targets and not error in predicted locations, a similar relationship is used here.

3.4 Conclusion

This chapter introduced the basic components of the menu models, including an overview of the EPIC cognitive architecture, Nilsen's menu experiment, and the parameters in the models. The next chapter presents the strategies that were developed to explain Nilsen's data using EPIC.

CHAPTER 4

MODELING RESULTS

This chapter completes the presentation of the menu selection models introduced in the previous chapter. Specifically, this chapter narrates the development of cognitive strategies in an effort to explain Nilsen's menu selection data. Each strategy, and thus model, will be evaluated with respect to how well it accounts for Nilsen's data. Models of Nilsen's randomly ordered menu data will be presented first, and then models of Nilsen's numerically ordered menu data .

4.1 Randomly Ordered Menu Models

As discussed in the previous chapter, the important features in Nilsen's randomly ordered menu data include:

- *Menu length effect.* When the target item is in the same Serial Position across menus of different lengths, shorter menus are faster.
- *Serial Position effect.* Selection time increases with a fairly linear slope of about 100 msec per item. As can be seen in Figure 3.5, the mouse movement time predicted by Fitts' law cannot entirely account for this slope either in shape or magnitude.
- *Position 1 effect.* The selection time for Serial Position 1 is a little higher than the selection time for Serial Position 2.

Six randomly ordered menu models will be presented. The six models result from varying two strategic dimensions and one parameter in the task environment. The strategic dimensions are (a) serial versus parallel processing of menu items, and (b) random versus systematic search. The parameter in the task environment is eye-to-screen distance, which is set (to either 8 or 20 inches) to result in one or three items being visible in the fovea at the same time.

The discussion of each model includes a flowchart that summarizes the production rules written in EPIC to represent that model. Production rules were written to maximize performance within the constraints imposed by EPIC, and to be as simple as possible.

4.1.1 Serial Processing Models

The serial processing models represent a belief that people process one menu item at a time—that people move their gaze to an item, visually process it, decide if it is the target, click on the item if it is, or go on to the next item if it is not. As discussed in Chapter 2, many researchers have proposed such a process. The proposed model does not specify the search strategy used to find the next item and, as discussed in Chapter 2, researchers have proposed two major opposing ideas: random versus systematic. So two separate sets of production rules were built in EPIC to represent two possible serial processing models: one with random search and the other with systematic top-to-bottom search. These sets of rules are listed in Appendices A and B.

Both serial processing models were only run with an eye-to-screen distance of 8 inches so that only one item would fit into the fovea at a time, insuring a serial encoding process. At greater distances, more than one item would fit into the fovea simultaneously, and parallel encoding would ensue.

Serial Processing Random Search Model

The results from running the Serial Processing Random Search model are shown in Figure 4.1. Each predicted selection time is averaged from 300 trials run for that menu length and Serial Position combination.

The results in Figure 4.1 suggest that the Serial Processing Random Search model is wrong. The only feature in the observed data that this model accounts for is that shorter menus are faster than longer menus. Otherwise, the model does not fit the observed data. Selection times are much too high overall. Slopes are very small because every item takes

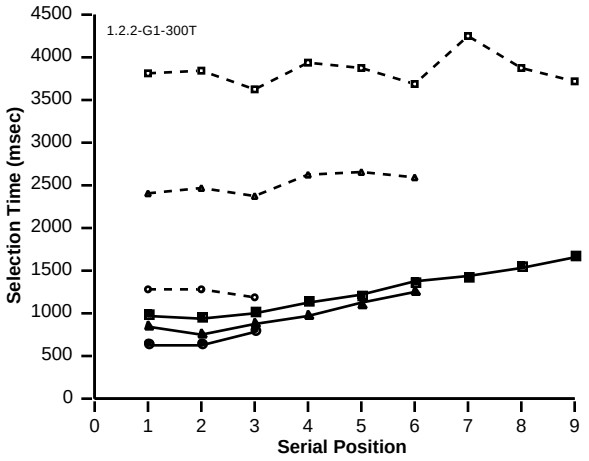

Figure 4.1. Selection times observed (solid lines) and predicted (dashed lines) by the Serial Processing Random Search model run with one item fitting into the fovea.

on average the same amount of time to find and select; any slope that appears is due to the mouse movement. A higher selection time for Serial Position 1 is not predicted. This model does not account for the observed data.

Serial Processing Systematic Search Model

The results from running the Serial Processing Systematic Search model, shown in Figure 4.2, suggest that this model is also wrong. The only feature in the observed data that this model accounts for is a positive slope greater than that of the predicted Fitts movement time. The model accounts for no other features in the observed data. Shorter menus are not faster. The slope of the predicted data is too steep. The selection time for Serial Position 1 is not higher than for Serial Position 2. This model does not account for the observed data.

The prediction has a slope resulting from more than just the mouse movement, but the predicted slope is too steep, about 380 msec per item as opposed to about 100 msec per item in the observed data. The discrepancy between the predicted and observed data results from all of the processing that must take place before moving the gaze to the next menu item. The slope of approximately 380 msec results because this is the time required for

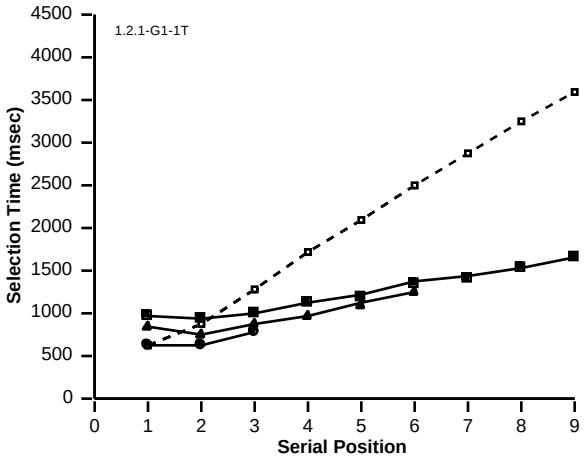

Figure 4.2. Selection times observed (solid lines) and predicted (dashed lines) by the Serial Processing Systematic Search model run with one item fitting into the fovea. The predicted times for the same Serial Position in different menu lengths are the same and are thus superimposed.

EPIC to move the eye, perceptually process a menu item, move the features to working memory (WM), and decide on an item. Serially processing each item cannot produce a slope of 100 msec per item unless the architecture is radically altered by setting the processing times for each of these components to be implausibly fast. Only by processing multiple items at once can a model produce such a small slope.

The results provided by the serial processing models provide strong evidence that, when scanning a menu, people process more than one menu item at a time. The serial processing models proposed by Norman (1991) and Vandierendonck et al. (1988) are highly implausible given the constraints set by the architecture. Menu selection models should incorporate the assumption that people process more than one menu item at a time. The remaining models presented in this paper utilize parallel processing of menu items.

4.1.2 Parallel Processing Models

The parallel processing models represent a belief that people move their gaze across the menu as quickly as their perceptual-cognitive-motor processes allow, process the features of all objects that appear in the fovea in parallel using a “pipeline” facility to

continue recognition even after the gaze has shifted away, and at the same time continually check WM to see if the target item has been seen. As soon as the target item has been located, the person moves his or her gaze to it and clicks on it. In one of the parallel processing models, people search randomly for the target; in the other, they start at the top and scan down the menu.

Both parallel processing models were run with different eye-to-screen distances that resulted in one and three items fitting into the fovea simultaneously. When more than one item is visible in the fovea, all of those objects' properties are sent to WM in parallel and, when searching, the next eye movement will always be to an item not currently in the fovea.

Parallel Processing Random Search Model

The Parallel Processing Random Search model was inspired by Card (1983, reviewed above in Chapter 2), who proposed that people search randomly. The production rules for the Parallel Processing Random Search strategy are listed in Appendix C. Figure 4.3 shows a flowchart summarizing the rules. For the sake of brevity, the flowchart summarizes the two different sets of motor movements required for the two different menu styles (walking versus click-open) as just click, move, and click. To prevent a random eye "movement" to essentially the same location, the model chooses the next item to look at from the items currently outside the fovea.

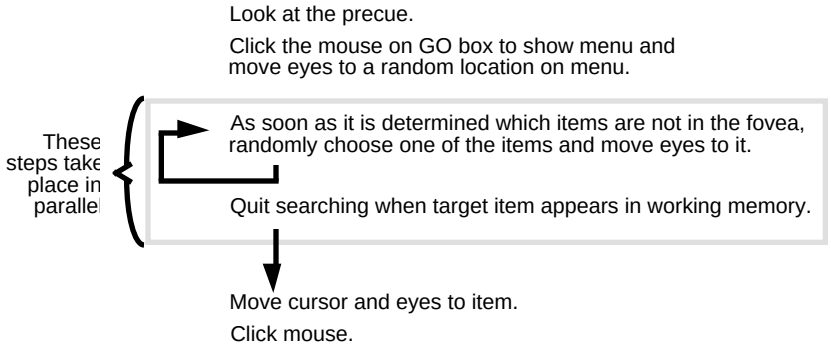

Figure 4.3. Parallel Processing Random Search model.

The results from running the Parallel Processing Random Search model are shown in Figure 4.4. Each predicted selection time is averaged from 300 trials run for that menu length and Serial Position combination.

The predictions from the Parallel Processing Random Search model have some features that correspond to the observed data, but also have some problems.

As can be seen in Figure 4.4 (top graph), when one item at a time is visible in the fovea, the model accounts for shorter menus being faster, but no other features of the observed data. The overall predicted times are, however, significantly lower than in the Serial Processing Random Search model discussed above.

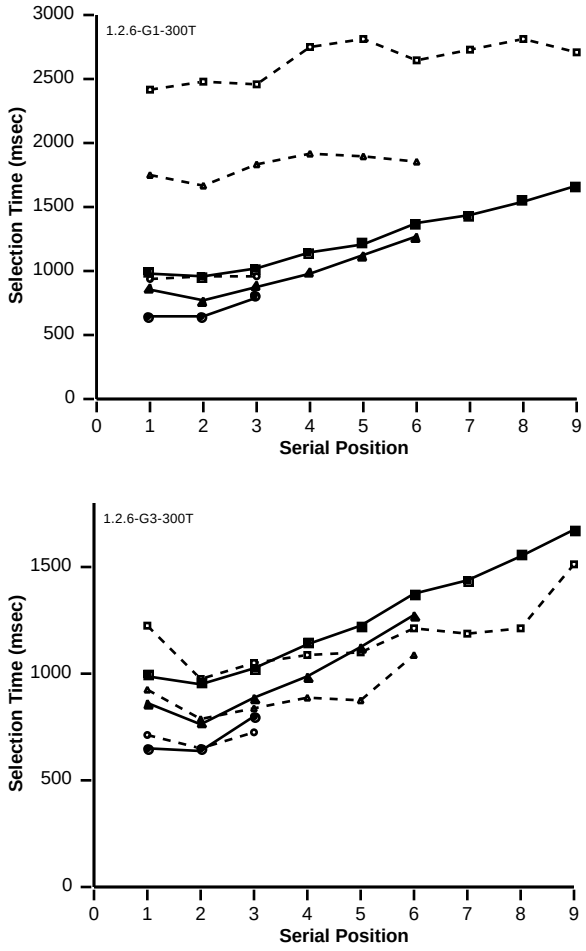

Figure 4.4. Selection times observed (solid lines) and predicted (dashed lines) by the Parallel Processing Random Search model run with one item (top graph) and three items (bottom graph) fitting into the fovea.

As can be seen in Figure 4.4 (bottom graph), when three items are visible in the fovea simultaneously, the model accounts for shorter menus being faster, and about the right amount faster, as is shown by the distance between the predicted lines approximating the distance between the observed lines. The predicted values fall entirely within the range of the observed values. Most importantly, this model accounts for Serial Position 1 being higher than Serial Position 2. However, the overall slope is still too small.

In Figure 4.4 (bottom graph), both the first and last Serial Positions are higher because the model combines random search with three menu items fitting into the fovea. Items at both ends of the menu have a lower probability of being in the fovea after any random fixation. Any of the middle menu items can be foveated by moving the eye to that item, or to either of the two adjacent items. But the first and last items only have one adjacent item. This might explain Serial Position 1 being higher than Serial Position 2 in the observed data.

The predictions from the Parallel Processing Random Search model suggest that the model is partly correct, and partly incorrect.

Parallel Processing Systematic Search Model

Figure 4.5 is a flowchart that represents the production rules built in EPIC to investigate the possibility that participants used a Parallel Processing Systematic Search strategy. Though other systematic searches are possible, top-to-bottom is the most obvious one to explore. The production rules are also listed in Appendix D.

In this model, the first eye movement is made to any of the items that are within one foveal radius from the topmost item (to insure the first gaze captures the topmost item). Each subsequent movement is made to an item one foveal diameter below the center of the current fixation. These details represent the belief that, when using a systematic search strategy, people attempt to maximize the foveal coverage with a minimum number of eye movements.

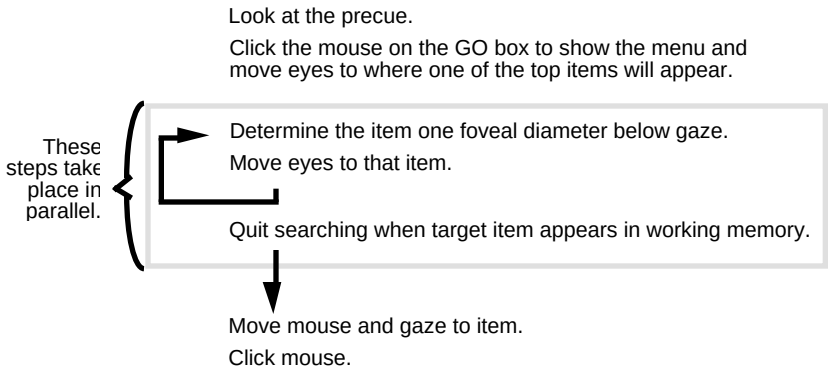

Figure 4.5. Parallel Processing Systematic Search model.

The results from running the Parallel Processing Systematic Search model are shown in Figure 4.6. Each predicted selection time is averaged from one trial run for each possible combination of menu length, Serial Position, and first eye movement.

The predictions from the Parallel Processing Systematic Search model have some features that correspond to the observed, but also have some problems.

As can be seen in Figure 4.6 (top graph), when one item at a time is visible in the fovea, the model only accounts for a positive slope. The model does not predict that shorter menus will be faster, the slope is too steep, and Serial Position 1 is not higher.

As can be seen in Figure 4.6 (bottom graph), when three items are visible in the fovea simultaneously, the model can account for important features of the data. The slope is correct and the predicted values fall entirely within the range of the observed values. But again, the model does not account for shorter menus being faster, and Serial Position 1 is not higher.

These results show that the Parallel Processing Systematic Search model can partially explain how the participants accomplished the task, but not account for all aspects of the observed data.

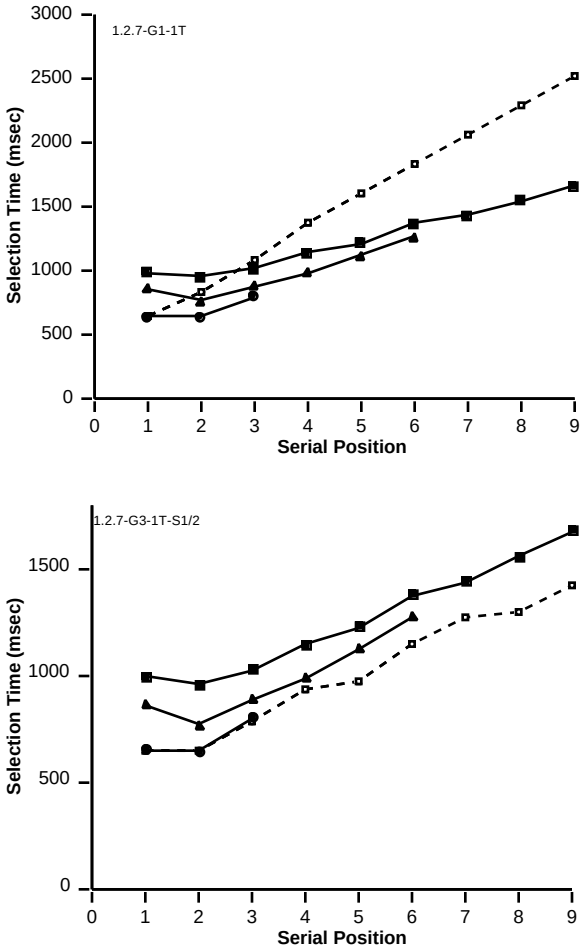

Figure 4.6. Selection times observed (solid lines) and predicted (dashed lines) by the Parallel Processing Systematic Search model run with one item (top graph) and three items (bottom graph) fitting into the fovea. In each graph, the predicted times for the same Serial Position in different length menus are the same and are thus superimposed.

Hybrid Models

None of the models presented thus far can account for all of the features in the observed data. The serial processing models account for essentially none of the features of the observed data. But all features of the observed data are accounted for by at least one of the various parallel processing models. The hybrid models represent a belief that, when Nilsen ran his experiment, (a) participants used both random *and* systematic search, and (b) screen-to-eye distance varied across trials.

The hybrid models were motivated by observing that all of the features in the observed data are accounted for by at least one of the parallel processing models with one or three items fitting into the fovea. The random search model accounts for faster selection times in shorter menus. When three items fit into the fovea, the random search model also accounts for Serial Position 1 being higher. The systematic search model accounts for the correct slope when three items fit into the fovea.

Dual Strategy Hybrid Model

The Dual Strategy Hybrid model represents the hypothesis that participants processed menu items in parallel in all of the observed trials, but that participants searched randomly in some of the trials and systematically in the remainder of the trials. Such a model could accurately account for the observed data if (a) some participants searched randomly and others systematically, or (b) participants varied their search strategy from trial to trial. Since the observed data were averaged across participants and blocks, either scenario would produce the same results.

Predictions from this hybrid model can be obtained in two ways. The first is to build a set of EPIC production rules that contain the rules from both the Parallel Processing Random Search strategy and the Parallel Processing Systematic Search strategy; the strategy would randomly choose which search strategy to use at the start of each trial. The second is to compute the weighted average of the predicted values produced by running the two models independently. Since both approaches would produce the same predictions, the second approach was chosen for expedience. Figure 4.7 shows the results of this model, as determined by averaging the results shown in Figure 4.4 and Figure 4.6. An initial weighting of 50% for each strategy was chosen to explore the predictive potential of the model.

The predictions from the Dual Strategy Hybrid model can account for most of the features in the observed data, but do not fit the observed values perfectly.

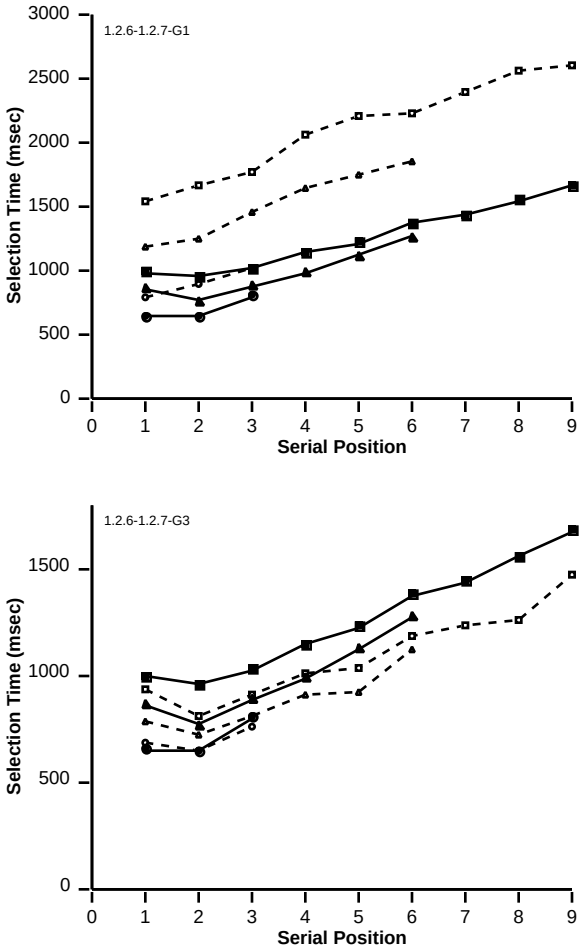

Figure 4.7. Selection times observed (solid lines) and predicted (dashed lines) by the Dual Strategy Hybrid model, with one item (top graph) and three items (bottom graph) fitting into the fovea.

As can be seen in Figure 4.7 (top graph), when one item fits into the fovea, the model accounts for faster selection times in shorter menus and produces a near-perfect slope. But the model does not account for the higher selection time in Serial Position 1, and overall the predicted values are higher than the observed values.

As can be seen in Figure 4.7 (bottom graph), when three items fit into the fovea, the model accounts for faster selection times in shorter menus, produces a comparable slope, accounts for the higher selection time in Serial Position 1, and predicts values that are in range of the observed data. The only shortcoming of this model is that the predicted values do not exactly match the observed values.

Dual Strategy Varying Distance Hybrid Model

The Dual Strategy Varying Distance Hybrid model represents a hypothesis that (a) participants searched randomly in some of the trials and systematically in the remainder of the trials and (b) the eye-to-screen distance varied across trials. Since this distance was not controlled or measured during the experiment, it is very likely that some participants sat closer to the computer screen than others, and that participants moved nearer to and further from the screen during the course of the experiment.

Predictions from this hybrid model can be obtained in two ways. The first is to build a task environment that varies the screen distance from trial to trial, and to randomly choose the search strategy at the start of each trial. The second is to compute the weighted average of the predicted values produced by each of the following four models:

1. Random search, one item in the fovea (Figure 4.4, top).
2. Random search, three items in the fovea (Figure 4.4, bottom).
3. Systematic search, one item in the fovea (Figure 4.6, top).
4. Systematic search, three items in the fovea (Figure 4.6, bottom).

Since both approaches would produce the same predictions, the second approach was chosen for expedience.

The procedure for obtaining the best-fitting weighted average of the four models will now be described. The four models result from varying two factors: (a) strategy and (b) eye-to-screen distance (which determines the number of items in the fovea). Each of the two factors was varied independently. A computer program stepped through all possible contributions of each of the two factors, that is, w_r% random and (100-w_r)% systematic, and w_1 one-item-in-the-fovea and (100-w_1)% three-items-in-the-fovea, where w_r and w_1 were both integers from 0 to 100. The contribution of each of the four models was determined by multiplying the strategy weight by the items-in-the-fovea weight, such as $w_r \times w_1$ to determine the contribution of random search with one item in the fovea.

The weights that produced the lowest average absolute error across all of the data

points were 43% random (and 57% systematic) and 18% one item in the fovea (and 82% three items in the fovea). Multiplying these strategy weights and items-in-the-fovea weights resulted in the following contribution for each model:

7.7% Random search, one item in the fovea (Figure 4.4, top).

35.3% Random search, three items in the fovea (Figure 4.4, bottom).

10.3% Systematic search, one item in the fovea (Figure 4.6, top).

46.7% Systematic search, three items in the fovea (Figure 4.6, bottom).

Summing these contributions for each data point individually results in the Dual Strategy Varying Distance Hybrid model shown in Figures 4.8 and 4.9.

The Dual Strategy Varying Distance Hybrid model accounts for all of the features in the observed data. As can be seen in Figure 4.8, the model predicts the observed values very well, with an average absolute error of 2.8%. As can be seen in Figure 4.9, the model also accounts for the data when the data and the predictions are collapsed by menu length and expanded by menu style (walking versus click-open), with an average absolute error of 2.2%. Matching the observed values, the Dual Strategy Varying Distance Hybrid model offers a highly plausible explanation of the task environment and strategies used by participants in Nilsen's experiment.

The empirical data that is available aggregates the performance for all participants, and thus the models do the same. If individual participant data were available, they could be used to determine if some participants tended toward random search and others towards systematic search. This could be done by determining, for each participant, whether his or her data were better explained by the Parallel Processing Random Search Model or by the Parallel Processing Systematic Search.

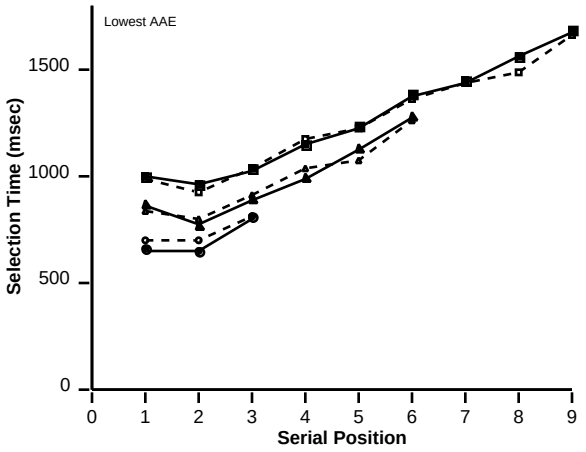

Figure 4.8. Selection times observed (solid lines) and predicted (dashed lines) with by the Dual Strategy Varying Distance Hybrid model, with 43% of the trials using random search and 57% of the trials using systematic search, and with one item fitting into the fovea on 18% of the trials and three items fitting into the fovea on 82% of the trials.

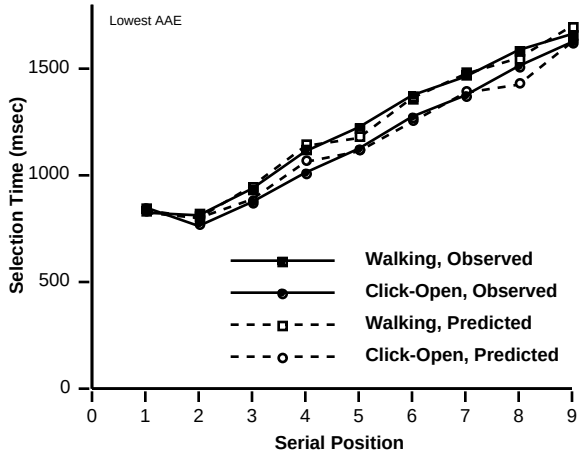

Figure 4.9. The same data and predictions as shown in Figure 4.8, but collapsed by menu length and expanded by menu style (walking versus click-open).

4.2 Numerically Ordered Menu Models

Models of Nilsen's numerically ordered menu data will now be presented. As discussed in Chapter 3, the important features in the numerically ordered menu data include:

- *Faster than random.* Participants select an item from a numerically ordered menu substantially faster than from a randomly ordered menu. The same model will probably not be able to account for both random and numerically ordered menu data.
- *Very fast.* Participants select the target item from numerically ordered menus very quickly.
- *No menu length effect.* Every Serial Position takes the same amount of time regardless of the menu length.
- *Diminishing Serial Position effect.* There is a negatively accelerated increase in the numerically ordered menu data; the increase is greater than that of the Fitts' law prediction also shown on the graph.

Two classes of models will be presented to explain the data—the Immediate Look, Point, and Click models and the Immediate Look, Point, Check and Correct models. All of the models assume that people will use anticipated location knowledge to prepare and execute eye and hand movements to the target without waiting for the menu to actually appear in WM. The anticipated location knowledge is made available by means of the named locations discussed in Chapter 3.

4.2.1 Immediate Look, Point, and Click Models

The Immediate Look, Point, and Click models represent a hypothesis that people anticipate a target location before opening a menu, execute an eye movement and a mouse movement to that location immediately upon opening the menu, and then click on that location without confirming that the cursor is actually on the target. This strategy assumes that anticipated target locations are correct, and thus takes maximum advantage of the anticipated locations to complete the selection as efficiently as possible. The EPIC

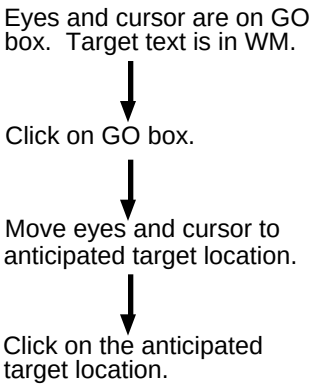

Figure 4.10. The Immediate Look, Point, and Click strategy.

production rules to represent this strategy are summarized in Figure 4.10, and are listed in Appendix E.

Standard Fitts' Law Coefficients Model

The results from running the Immediate Look, Point, and Click strategy are shown in Figure 4.11. Each predicted selection time is averaged from one trial run for every menu length and Serial Position combination. For these trials, the Fitts' law coefficients in EPIC were set to the standard 100 for a POINT and 140 for a POINT-PRESSING.

The predictions of this model are shown in Figure 4.11. Since there is no menu

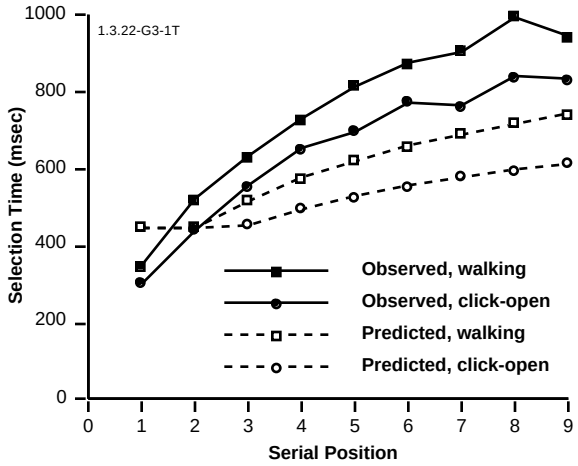

Figure 4.11. Selection times observed by Nilsen and predicted by the Immediate Look, Point, and Click strategy run with standard Fitts' coefficients of 100 and 140.

length effect in the numerically ordered data, as shown earlier in Figure 3.4, Nilsen's data and the model's predictions are shown collapsed by menu length, but expanded by menu style (walking versus click-open). The predictions demonstrate that the model is incorrect.

The predicted values are negatively accelerated, as are the observed data, and the difference between the two menu styles is predicted to be the same as the observed data. But the predictions for most Serial Positions are much too fast, the trend in the predicted values does not increase steeply enough, and the prediction for Serial Position 1 is much too high.

For Serial Positions 2 through 9, the model could be underpredicting for a number of reasons, including (a) participants could not anticipate the exact location of the target, which would imply that (b) this is not the strategy participants really used, or (c) participants took longer to point than is predicted by Fitts' law with the standard coefficients. The next model investigates the third of these possibilities.

Nonstandard Fitts' Law Coefficients Model

The Immediate Look, Point, and Click strategy run with nonstandard Fitts' coefficients represents the belief that participants could anticipate the exact location of a target item before the menu appears and always execute a correct eye and hand movement to the target, but that mouse points took longer than is predicted by standard Fitts' coefficients. The results from running the Immediate Look, Point, and Click strategy with exactly known location information and with nonstandard Fitts' coefficients of 175 and 220 are shown in Figure 4.12. The values of 175 for POINT and 220 for POINT-PRESSING were chosen iteratively to provide a good fit. The implications of these increased values are discussed later.

With the increased Fitts' coefficients, this model now does a very good job of predicting selection times for Serial Positions 2 through 9. The difference between the predicted and observed values for the two menu styles is the same, and both the predicted

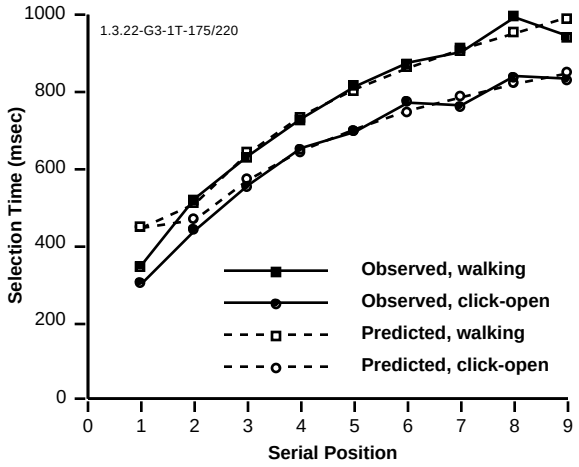

Figure 4.12. Selection times observed by Nilsen and predicted by the Immediate Look, Point, and Click strategy run with Fitts' coefficients increased to 175 and 220.

and the observed values follow the same negatively accelerated trend. The overall plausibility of this model and the implications of the nonstandard Fitts' coefficients will be discussed after providing a plausible explanation for the high speed of Serial Position 1.

Special Case for Position 1 Model

An explanation for how participants selected targets in Serial Position 1 so quickly requires a detailed analysis of the task. Recall that upon clicking on the GO box, the cursor is automatically positioned exactly one pixel above the first menu item. When the participant knows the target item will be in Serial Position 1, all that he or she must do is click on the GO box, make a tiny downward movement with the mouse, confirm that the target has actually appeared, and click again.

Additional production rules were added to the Immediate Look, Point, and Click strategy to create a Special Case for Position 1 branch, rules that will only be executed if the precue is a *1*. A flowchart summarizing the production rules appears in Figure 4.13. In the Special Case for Position 1 branch, there is no separate POINT movement. Rather, to accommodate this aspect of the experimental procedure that only affects Serial Position 1, the click on the GO box is assumed to include a tiny downward twitch. The rules are

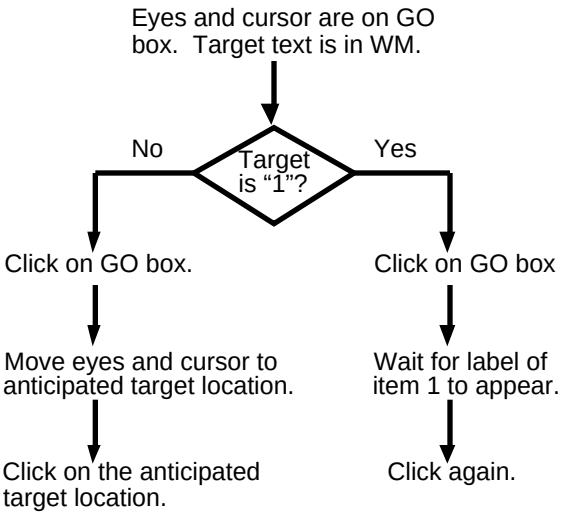

Figure 4.13. The Immediate Look, Point, and Click strategy with Special Case for Position 1 branch.

listed in Appendix F.

EPIC's predictions when running the Immediate Look, Point, and Click strategy with Special Case for Position 1 and nonstandard Fitts' coefficients are shown in Figure 4.14. As can be seen, this model predicts the observed data very well, for an average absolute error of 3.0%.

These nonstandard Fitts' coefficients will only slightly decrease the good fit of the Dual Strategy Varying Distance Hybrid model for randomly ordered menus presented in Section 4.1.10 above. Repeating the procedure outlined in that section, but with these nonstandard Fitts coefficients of 175 and 220, results in a Dual Strategy Varying Distance Hybrid model with nonstandard Fitts' coefficients prediction of 4.3% rather than 2.8% when run with the standard Fitts' coefficients.

Though this model explains the data well and offers a reasonable explanation for how people accomplish the task, there are two aspects of this model that make it questionable. First, it is hard to accept Fitts' coefficients so much higher than the standard values. Second, the model asserts people know exactly where to look and point before the menu even appears.

The first problem, of the increased Fitts' coefficients, actually points to a

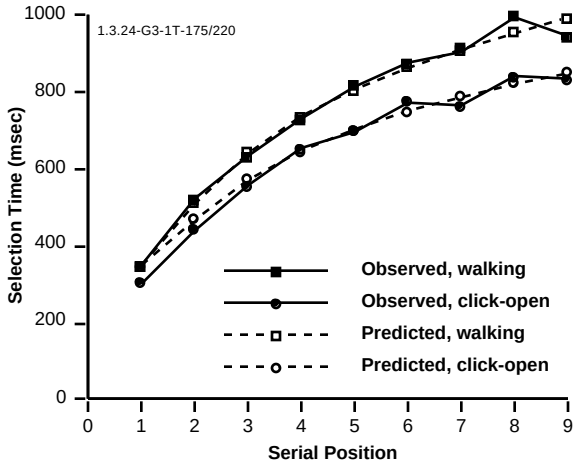

Figure 4.14. Selection times observed by Nilsen and predicted by the Immediate Look, Point, and Click strategy with Special Case for Position 1 run with nonstandard Fitts' coefficients of 175 and 220.

shortcoming in the HCI literature. Though Fitts' law is often cited as a useful tool for prediction and design in HCI (such as in Card et al., 1983; Han, Jorna, Miller & Tan, 1990; MacKenzie & Buxton, 1992), the exact form and coefficients of Fitts' law are not settled. Fitts' equation appears in several forms (for example, compare Card et al., 1983; Han et al., 1990; MacKenzie & Buxton, 1992), which makes some coefficients incomparable. Some studies in fact provide evidence for a Welford form of Fitts' law with a coefficient of about 175 for a mouse point (Han et al., 1990; MacKenzie & Buxton, 1992). But it is not clear whether such large Fitts' coefficients are reasonable. Much more work needs to be done to determine the correct Fitts' coefficients for various tasks, pointing devices, and environments.

The second problem is that all of the Immediate Look, Point, and Click models assume that a person has exact location knowledge for all menu items before the menu even appears. This assertion seems to contradict Perlman's (1984) and Somberg's (1987) findings that, even with numerically and alphabetically ordered menus and a constant time to select an item once it is found, the top menu items can be selected faster than lower menu items. Perlman's and Somberg's findings suggest that some items do take longer to locate even in a known, ordered menu.

So, the Immediate Look, Point, and Click models provide a good fit and a reasonable explanation for how people select an item from an ordered menu. But the models discussed next provide an equally good fit and perhaps an even more plausible explanation.

4.2.2 Immediate Look, Point, Check and Correct Models

The Immediate Look, Point, Check and Correct models represent a hypothesis that people anticipate a target location before opening a menu, execute an eye movement and a mouse movement to that location immediately upon opening a menu, check to see if the cursor actually landed on the target, make a corrective eye movement and mouse movement if necessary, and then click on the target. These models allow us to explore the possibility that people cannot predict the exact location of the target before it appears, but only an approximate location.

The flowchart in Figure 4.15 summarizes the production rules written in EPIC to explore the plausibility of this strategy. Note that the strategy carries forward the Special Case for Position 1 branch discussed in the previous section. The rules are listed in Appendix G.

For simplicity, the model asserts that a third eye and mouse movement will never be necessary. For the small amount of error introduced in these models, the first movement will rarely fall more than one menu item away from the target, in which case the correct location information will be readily available for the second eye and hand movement.

Exactly Known Location Model

Running the Immediate Look, Point, Check and Correct strategy in EPIC with exactly known location information reveals the baseline prediction of the strategy, before adding any error to the initial eye and hand movement location. The results from running this model are shown in Figure 4.16. Each predicted selection time is averaged from one

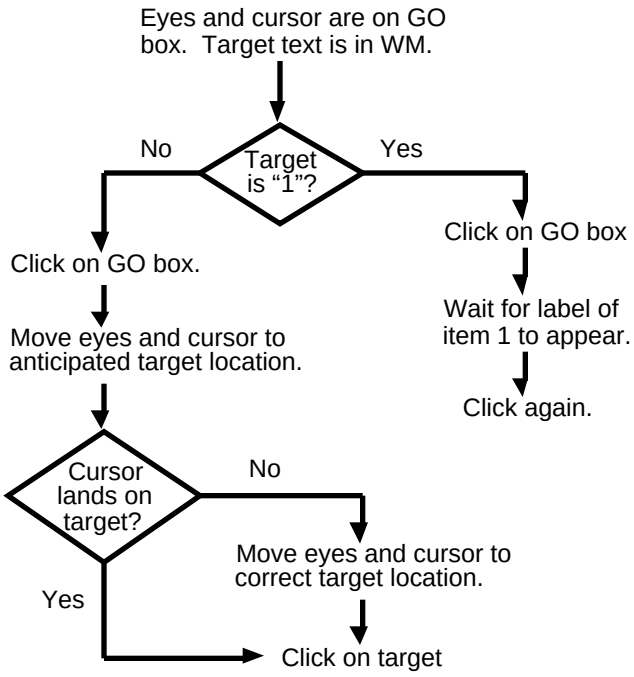

Figure 4.15. The Immediate Look, Point, Check and Correct strategy.

trial run for every menu length and Serial Position combination.

As can be seen in Figure 4.16, the model does not account for the data. But the results are informative nonetheless. The model's predictions for the first three Serial Positions are very close to the observed, and with roughly the same negatively accelerated slope as the data. The model underpredicts for Serial Positions 4 and above, which might be remedied by adding some error to the model that would sometimes make necessary a second, corrective eye and hand movement.

Approximately Known Location Model

The Immediate Look, Point, Check and Correct model run with approximately known location information represents the belief that people can anticipate the location of a target in a menu before the menu actually appears, but that people can anticipate the location of items higher in the menu more accurately than items lower in the menu. Approximately known locations are introduced to the model by means of the *target-location-with-error*

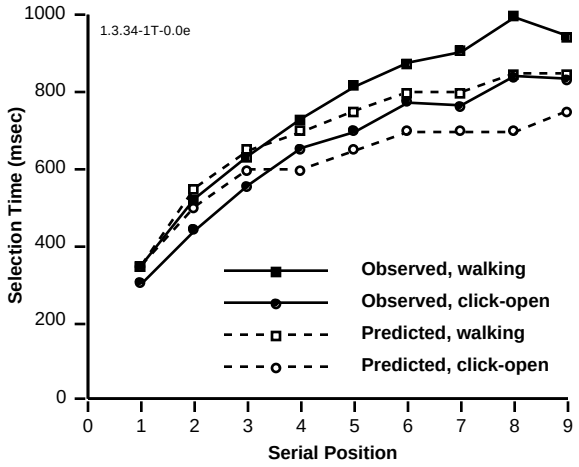

Figure 4.16. Selection times observed by Nilsen and predicted by the Immediate Look, Point, Check and Correct strategy run with exact location knowledge.

named location. As discussed in Section 3.3.4, this named location is computed at the beginning of every trial by first computing the exactly correct location, and then perturbing the vertical component of that location. The error increases as a function of the distance from the GO box to the target, and as a function of a constant error coefficient *e*.

The results from running the Immediate Look, Point, Check and Correct strategy with an initial error coefficient $e = 0.1$ are shown in Figure 4.17. The value of 0.1 was chosen iteratively to provide a similar slope as that of the data. Three hundred trial runs were executed for every unique combination of menu length, Serial Position, and menu style. The predictions in Figure 4.17 average those results.

As can be seen in Figure 4.17, the model comes very close to explaining the observed data. The predicted values have almost exactly the same negatively accelerated slope as the observed data, and are very close to the observed data, but the model's predictions are a little too slow for how quickly people accomplished this task.

Perhaps the overall high speed of the observed data results because people are able prepare and execute complex and subtle combinations of movements as if they were a single movement. For example, perhaps people can prepare and execute a compound click-and-point movement, a movement style not currently implemented in EPIC. To explore

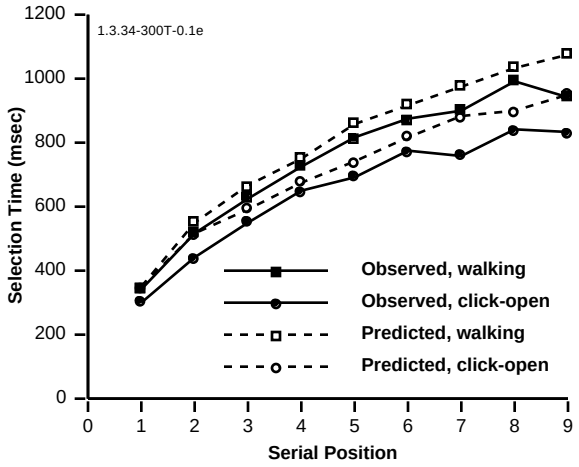

Figure 4.17. Selection times observed by Nilsen and predicted by the Immediate Look, Point, Check and Correct strategy run with approximate location knowledge (e = 0.1).

this possibility, the tentative new *click-and-point* compound movement style, discussed in Section 3.3.3, is used in the next model.

Introducing the *click-and-point* compound movement style to the Immediate Look, Point, Check and Correct model represents a hypothesis that, since the destination of the initial mouse point can be determined in advance, the motor preparation for the point movement can also be partly prepared in tandem with the first mouse click.

Click-and-Point Compound Movement Style Model

The results from running the Immediate Look, Point, Check and Correct strategy with an initial location error coefficient $e = 0.1$ and a click-and-point compound movement style are shown in Figure 4.18. The predictions in Figure 4.18 average the results from three hundred trial runs executed for every unique combination of menu length, Serial Position, and menu style.

As can be seen in Figure 4.18, this model explains the observed data very well, with an average absolute error of 3.92%. This model demonstrates that two problems with the Immediate Look, Point, and Click models—increasing the Fitts' coefficients and asserting perfect location knowledge—can be overcome by a more subtle analysis of the

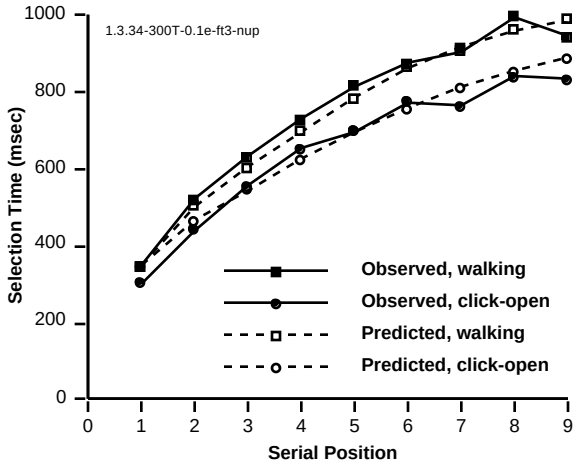

Figure 4.18. Selection times observed by Nilsen and predicted by the Immediate Look, Point, Check and Correct strategy run with approximate location knowledge (e = 0.1) and with a click-and-point compound movement style.

task and a more detailed representation of the perceptual-motor activity required to accomplish the task.

4.3 Discussion

4.3.1 Further Support of the Models

The level of detail and completeness of these models lends them to a wide variety of testing and validation, including with eye movement studies. Two recent independent eye movement studies evaluate the Dual Strategy Varying Distance Hybrid (DSVDH) model for randomly ordered menus presented in Section 4.1.2. Both studies support the major findings of the model.

Aaltonen, Hyrskykari, and Räihä (1998) ran an experiment in which they presented participants with pull-down menus containing names or concepts grouped by category, and asked participants to find specific targets in the menus. They collected eye movement data that support the two major conclusions of DSVDH model—that people process menu items in parallel, and that search is both random and systematic. They went so far as to state:

"The average saccade length... was 2.21 menu items. This supports the parallel search strategy (suggested by Hornof and Kieras [1997]) where more than one menu item are processed at a time." As well, Aaltonen, Hyrskykari, and Räihä observed systematic top-to-bottom scan paths on some trials, and more random scan paths on other trials.

Byrne et al. (1999) presented participants with pull-down menus containing numbers or letters, randomly re-ordered for each trial, in an experiment that closely resembled Nilsen's. The Byrne et al. study was designed in part to evaluate the plausibility of the DSVDH model. The various data reported in the study support several aspects of the DSVDH model, as follows.

The eye movement data from Byrne et al. support the major conclusions of the DSVDH model—that people process menu items in parallel, and that search is both random and systematic. The data indicate that participants executed fewer fixations per item than would be required by serial processing, indicating parallel consideration of items. The data revealed more fixations to lower items when the target was lower in the screen—suggesting systematic search—but with a lot of noise in the actual fixation locations—suggesting random search.

The selection time data from Byrne et al. (1999), shown in Figure 4.19, also support the DSVDH model's finding that the menu length effect in Nilsen's data results from an interaction between random search and menu length. The menu length effect, though persistent in all Serial Positions in Nilsen's data, mostly disappears for Serial Positions 2 through 4 in Byrne et al.'s data. This is what the DSVDH model would predict if the first random eye movement were constrained to the first few menu items. This would be a reasonable constraint when modeling the data from Byrne et al. because the experiment was not blocked by menu length, as was Nilsen's. Hence, participants in Byrne et al.'s experiment could not anticipate the menu length before the menu appeared, and would likely confine their first eye movement to a location where they would be confident a menu item would appear.

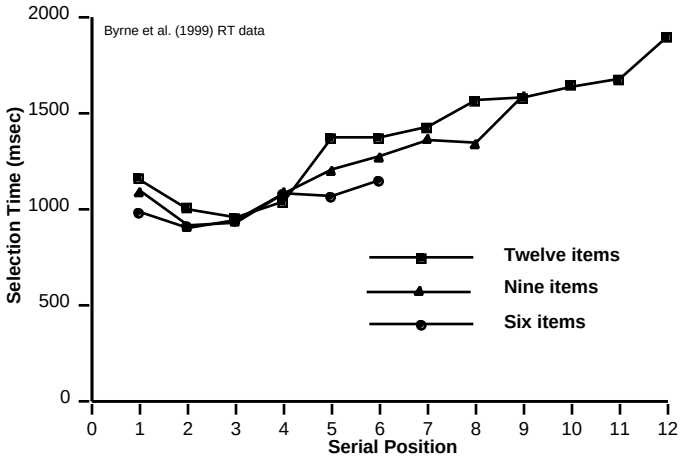

Figure 4.19. Observed data from Byrne et al. (1999). Mean selection times as a function of the Serial Position of target item, for menus with six, nine, or twelve items. The disappearance of the menu length effect in Serial Positions 2 through 4 would be predicted by the DSVDH strategy if the first random eye movement were constrained to land on one of the first few items.

Lastly, the mouse movement data collected by Byrne et al. support the DSVDH model's assumption that, in a high speed menu task, people wait until they have found the target and then execute one mouse point to the target. A single aimed movement to a target, such as a mouse point to a known menu location, is likely to contain several submovements (Meyer, Abrams, Kornblum, Wright & Smith, 1988; Meyer, Smith, Kornblum, Abrams & Wright, 1990; Rosenbaum, 1991). Fitts' law predicts the total time required for all of the submovements that together comprise a single aimed movement (Meyer et al., 1988; Meyer et al., 1990; Rosenbaum, 1991). The Byrne et al. data indicate that, for each trial, the initial submovement covered an average of two-thirds of the distance to the correct target location, and that there was on average 1 to 1.6 total submovements. These are the submovements that would be expected as part of a single aimed movement to the correct target location, such as the mouse point in the DSVDH model, but not what would be expected of submovements made while dragging the mouse down the menu while searching, as several researchers assume is the case.

All in all, the data provided by Aaltonen, Hyrskykari, and Räihä (1998) and Byrne

et al. (1999) provide strong support for the assumptions and conclusions of the models presented in this dissertation.

4.3.2 Comparison to a Menu Model in ACT-R

This section compares the EPIC models presented above to menu selection models built by Anderson et al. (1998; 1997) using another cognitive architecture, ACT-R (Anderson, 1993). The ACT-R models attempt to explain Nilsen's randomly ordered menu data.

ACT-R was originally developed as a model of higher cognition, but it has recently been updated to account for visual attention (Anderson & Lebiere, 1998). Nilsen's (1991) menu selection task is one of visually intensive tasks that Anderson et al. have modeled using ACT-R with its new visual interface.

The predictions of the EPIC and ACT-R models are shown in Figure 4.20. As can be seen, the EPIC model better predicts the data. For randomly ordered menus, the EPIC model has an average absolute error of only 3%, and the ACT-R model has an average absolute error of 12%. The EPIC model accounts for all of the features in the observed data: the slope, the Position 1 effect, and the menu length effect. The ACT-R model accounts only for the slope.

There are other differences between the two models.

The EPIC architecture and thus the EPIC models make specific assertions regarding eye movements. The ACT-R model "moves attention," and the relationship between ACT-R's attentional shifts and physical eye movements is unclear.

The EPIC models specifically account for the mouse movement time that was part of Nilsen's observed selection time. The ACT-R model assumes that mouse movement time was 0 msec based on an assumption that "subjects tend to move the mouse down as they scan for the target" (Anderson et al., 1997, p.456), so no mouse movement is required once the target has been found. But empirical evidence (Byrne et al., 1999)

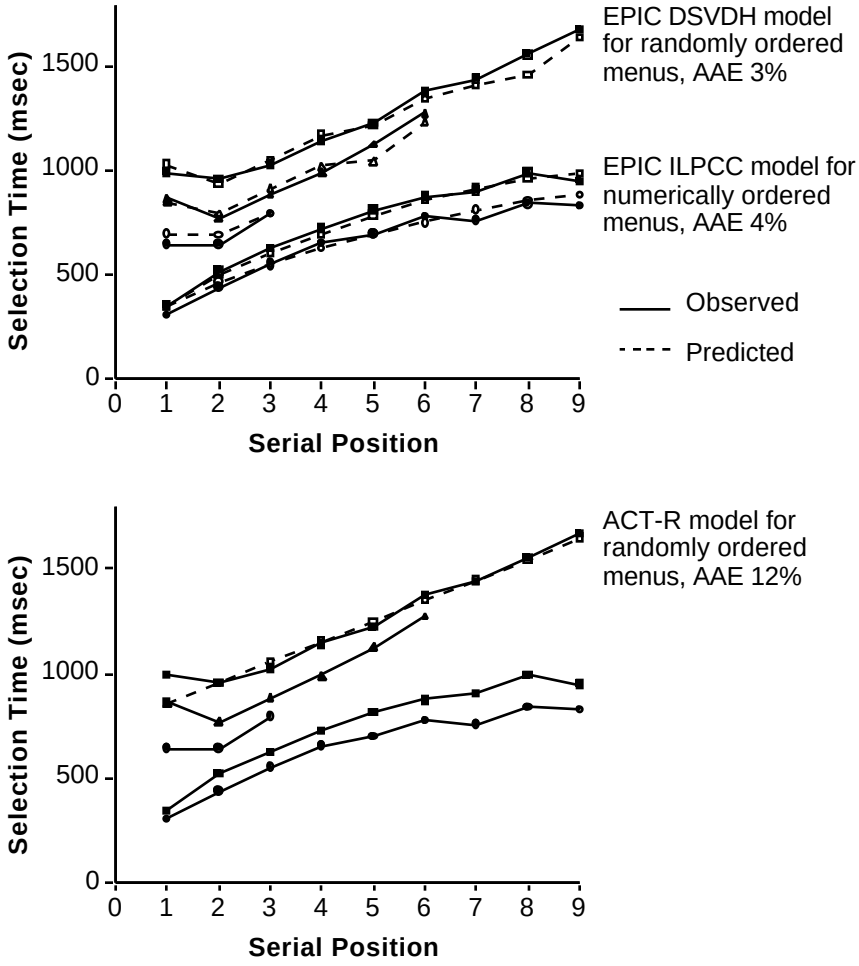

Figure 4.20. Selection times observed by Nilsen and predicted by the EPIC and ACT-R menu selection models, for both randomly and numerically ordered menus. The EPIC models are the Dual Strategy Varying Distance Hybrid (DSVDH) model and the Immediate Look, Point, Check and Correct (ILPCC) model. Also, the average absolute error (AAE) of each model.

suggests that participants did not move the mouse until after they found the target.

On a theoretical level, the EPIC and ACT-R models for randomly ordered menus take different approaches to modeling the data. The EPIC model distributes the activities of perception and visual search more throughout the model, using EPIC's retinal zones and perceptual encodings to account for the availability of simple perceptual features, but the strategic component to account for how a person might use the perceptual information to guide the search. The ACT-R model, however, folds most of the perception and search-

related activity into a single complex encoding of stimulus features, which are then made available to a simple, two-rule, purely serial, purely systematic search strategy.

4.4 Conclusion

The models presented in this chapter provide plausible explanations for the perceptual, cognitive, and motor processing required for selecting a known target item from a pull-down menu, either an unfamiliar randomly ordered menu, or a familiar ordered menu. The next and final chapter will discuss the contributions these models, and implications for future research.

CHAPTER 5
CONTRIBUTIONS AND IMPLICATIONS

This dissertation demonstrates that for a very important practical application in the context of human-computer interaction, a detailed computational model of the mental and physical processes embodied in the human user provides detailed theoretical, empirical, and practical insights with which to understand the user and make fundamental improvements to software and hardware design. Specifically, these insights are relevant to addressing basic issues associated with graphical interface design that relate to computational cognitive modeling, visual search, menu search, and manual motor control. The contributions in these areas, as well as implications for future research, will now be discussed.

5.1 Further Validation of Cognitive Modeling

These models presented in this dissertation contribute to the study of computational cognitive modeling by further validating the principles, assertions, assumptions, and conventions embodied in EPIC. These models apply EPIC to a new set of data and to a new aspect of human performance—visual search. EPIC's ability to explain the data with straightforward strategies and perceptual encodings, and with minor enhancements to the architecture, provides further evidence that the architecture captures the fundamental processes, memories, and timings of human perception, cognition, and action. The menu models demonstrate that EPIC is particularly well-equipped to investigate previously disconnected theories of visual search and eye movements, and to provide a solid foundation for further investigation into visual search and other aspects of human performance.

Besides validating EPIC, the models also stimulate architectural enhancements to EPIC. This serves as a reminder that the EPIC architecture is built from the ground up to represent only the fixed components of the human information processor. Several conditions must exist before an architectural modification will be considered: empirical evidence, sound theoretical foundation, and reliable data that cannot be explained by straightforward inputs to EPIC. The two enhancements proposed in this dissertation—the *visual recoding after saccade* function and the click-and-point compound movement style—result from these conditions being met.

5.2 A Unified Theory of Visual Search

The models in this dissertation integrate research from many different studies and from several different fields, and synthesize previous research into a unified theory of visual search within a unified theory of cognition. These dynamic computational models incorporate and subsume previous descriptive and mathematical models of visual search and eye movement generation, including elements of Treisman's (1986) theory of parallel and serial feature extraction, Russo's (1978) models of cognition and eye movement integration, and various researchers' theories regarding eye movement preparation and execution (such as those reviewed by Rosenbaum, 1991).

The models for randomly ordered menus utilize the parallelism built into the EPIC architecture to combine previous research into a new theory of overlapped eye-movement programming and visual object evaluation. The basic theory is that people conduct the two processes independently and in parallel: (a) moving their eyes around a scene as quickly as possible, using only the object properties of appearance and location, and (b) evaluating whether or not the target object has been viewed yet. These models demonstrate various interactions between the two processes. For example, when it takes longer to identify a target than to program the next saccade, a person might actually foveate the target but continue scanning before realizing he or she just looked at the target; in such a situation, an

eye movement would be needed to return the gaze to the target.

The models provide new theoretical, empirical, and practical insights into fundamental aspects visual search. The models for randomly ordered menus, for example, propose two complementary ideas of how more than one visual object can be processed simultaneously in a visual search task.

The first idea is that visual objects can be pipelined. People can begin the processing required for one object before they have completed the processing of a previous object, and thus have several objects moving from the eye to visual WM at the same time, each item in a different stage of processing.

The second idea is that further parallelism is possible when more than one visual object fits into the fovea simultaneously. This in effect widens the pipeline, allowing several objects to fit into each stage of the pipeline at the same time. This does not assert a "spotlight of attention" within the fovea, as some researchers propose, but instead assumes that everything in the fovea begins moving to the visual WM at the same time. This idea helps to explain why people can scan vertical lists more rapidly than horizontal lists, and has many implications for screen layout design.

The models also offer a broader view of visual search strategies. As discussed in Section 2.3.3, previous research contends that menu search is either random *or* systematic. The Dual Strategy Varying Distance Hybrid (DSVDH) model for randomly ordered menus proposes and demonstrates that it can actually be both. In the DSVDH model, each trial is all random or all the systematic. Future work will determine exactly how elements of both are incorporated into a single search trial. This research has advanced the question from "random or systematic?" to "exactly *how* both random and systematic?"

5.3 New Insights into Menu Search

This dissertation provides new theoretical and empirical insight into the fundamental human information processing involved in the visual search of computer menus, insights

that have implications for fundamental improvements in the design of menus.

Previous theories of menu search, though supported by empirical data, emphasized the cognitive strategies involved in menu search and said little or nothing about the perceptual and motoric processing involved. As a result, previous models were incomplete, made implausible assumptions, and led to lengthy debate regarding some aspects of menu search while remaining silent regarding others. For example, the extensive debate about whether menu search is random or systematic (such as between Card, 1983, and MacGregor and Lee, 1987) implicitly assumed serial consideration of items. The models presented here account for all of the processing required and, as a result, offer new insight.

The models provide theoretical validation for several of the hypotheses about menu search discussed in Section 2.3.3, including (a) that people learn where things are, (b) that search is both random and systematic, and (c) that people terminate the search when they find the target. The models also support the hypothesis that search and selection are *independent* to the extent that, in a high speed menu task, people will not drag the cursor down the menu as each item is considered; but that search and selection are *overlapped* in that mouse movement preparation will occur before the target is found.

The models suggest that the hypothesis that people serially process one menu item at a time is incorrect. Rather, the models support a new hypothesis, that people consider several menu items in parallel. Parallel consideration of objects is not a new notion to cognitive psychology, but this is the first model of menu search model to incorporate the idea.

The models offer the first theory-based explanation for several previously unexplained phenomena in menu selection studies, including (a) the Position 1 effect, (b) the menu length effect, and (c) a longer search time for items lower in a positionally constant menu.

The Position 1 effect is that selecting a target in Position 1 is consistently slower

than in Position 2. This effect first appeared in data collected by Somberg (1982), and more recently in that of Nilsen (1991) and Byrne et al. (1999) The DSVDH model reproduces the effect by incorporating two different features into the model—more than one item fitting into the fovea at a time, and random search. These two features interact, creating a situation in which items at both ends of the menu have an overall lower probability of being in the fovea after any given fixation, and thus will have an overall slower selection time. This is a unique explanation for this phenomena that was made possible by building computational cognitive models, running the models, generating the data, observing the replication of the Position 1 effect, and studying the information processing traces provided by EPIC to determine the source of the effect in the model. When explanations for how random and systematic search are incorporated into a single strategy are developed, even more subtle explanations for the Position 1 effect should become available.

The menu length effect is that a target item can be selected in a shorter menu faster than in a longer menu, even when the target is in the same Serial Position. This effect was reported by Perlman (1984) and Nilsen (1991). The DSVDH model demonstrates that this effect occurs in randomly ordered menus as a result of random search interacting with the menu length.

A longer search time for items lower in a positionally constant menu was observed by Perlman (1984) and Somberg (1987). The effect was observed despite selection being held constant by selecting with a keystroke rather than a mouse movement. The target-location-with-error named location incorporated into the Immediate Look, Point, Check and Correct (ILPCC) model for numerically ordered menus offers a theory-based explanation for this phenomena, as well as a characterization of the error in movements to known locations, error that increases for targets further from the starting position.

Early indications are that the principles embodied in the DSVDH and ILPCC models will be able to contribute to a priori predictions of other menu selection data.

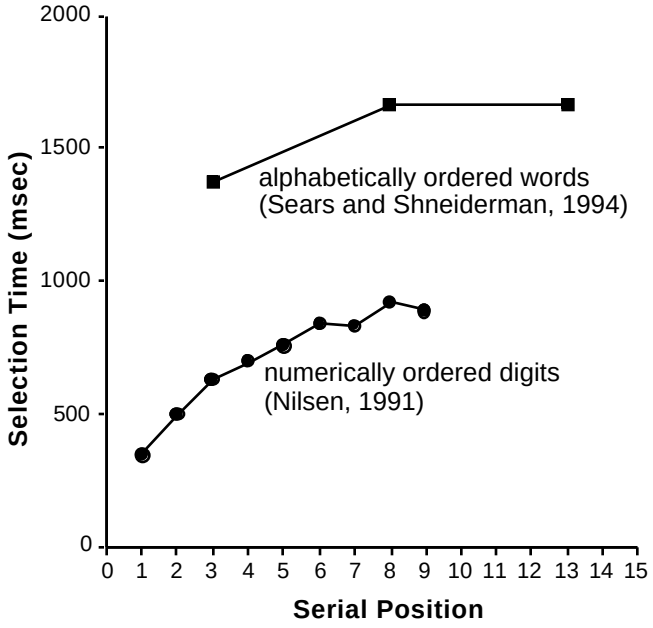

Figure 5.1. Selection time as a function of the Serial Position of the target item, for menus of alphabetically ordered words and numerically ordered digits. Selection time for words and digits increase at the same rate, but words consistently require an additional 850 msec.

Figure 5.1 shows Sears and Shneiderman's (1994) data for alphabetically ordered words next to Nilsen's (1991) for numerically ordered digits. Participants consistently required an additional 850 msec for the word menus. The ILPCC should be able to account for this data after the model is expanded to simulate the additional eye movements and processing that would be required to scan and evaluate words rather than digits.

5.4 Implications for Menu Design Guidelines

When designers better understand the human information processing involved in looking for something on a menu, they will be able to design menus that are easier for people to use. Without scientific theory to guide them, however, designers will work in the dark, using only their intuitions of how people will use their menus. Previous researchers have assumed that people consider each menu item one at a time. Based on menu items used in commercial applications, designers have evidently made the same

assumption.

The menu shown in Figure 5.2, for example, does not fully support parallel consideration of adjacent items. When a person is looking for the command to correct misspellings in a document, “Spell” is a more relevant keyword than “Check.” But “Spell” will not be captured in a maximally efficient foveal sweep of the leftmost word of every menu item, and a slower more thorough scan will be required. The designer’s decision to use “Check Spelling...” instead of “Spelling...” is likely based on an intuition that people will examine each menu item from beginning to end before proceeding to the next item.

New insights such as these provide a scientific basis for evaluating existing and establishing new menu design guidelines. A contradiction between existing guidelines that can now be addressed is whether menu items should be consistent in grammatical style (Mayhew, 1992) or begin with a keyword (Shneiderman, 1992). Perhaps there are benefits to using a consistent grammatical style, such as learnability or simplicity of design. But based on the notion of a maximally efficient foveal sweep, starting each menu item with the keyword should help a user to find a known target more quickly. Perhaps these two guidelines can be combined into “If the system will be used by novices, make menu

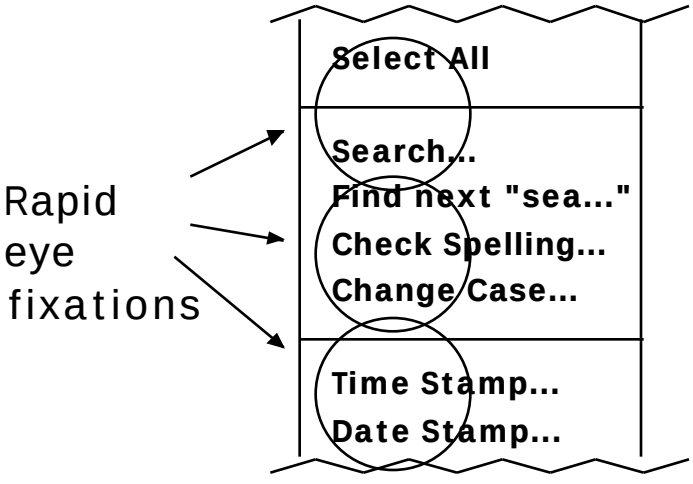

Figure 5.2. If a user is looking for the keyword “Spelling” and makes a maximally efficient foveal sweep down the left edge of these menu items, he or she will miss the target. (This menu is from MORE 3.1 by Symantec.)

items consistent in grammatical style. But if the system will be used primarily by experienced users, line up the keywords above and below each other in the leftmost position of each menu item."

A new guideline might be simply "Make sure initial parts of words are distinctive and not obscured. For example, starting each item with a bullet or other irrelevant marker could increase search time." Aaltonen et al. (1998) observed that non-text items such as checkmarks to indicate status and shortcut key information tend to attract people's attention during menu search. The little icons to the left of the text items in the Microsoft Windows NT menu in Figure 2.2 will hinder an efficient visual search if the user makes a maximally efficient foveal sweep of the icons rather than the words, and if some of the icons are meaningless.

Another guideline that can be proposed based on these models is that menu search should not be constrained by mouse movements. An extreme violation of this guideline is a menu in which every item is invisible except for the menu item where the mouse cursor is currently pointing. Such a menu might seem obviously unusable, but slightly less extreme violations are quite common. Many icon menus, such as the menu shown in Figure 5.3, require the user to move the cursor to the icon in order to display a text description of the item. To use this menu, the user must either spend a lot of time learning the exact meaning of every icon or, more likely, move the cursor over each item, wait for the text description to appear, examine the text, and then move the mouse to the next item. Such a menu does not readily support a high speed visual search.

Interestingly, the models can also inform the design of *auditory* menus. In

Figure 5.3. Some menus require the user to move the mouse cursor to the item to determine what the item is. Such menus result in very slow visual search. (Netscape Navigator 3.04 for the Macintosh)

auditory menus, users listen to options and make selections based on what they hear. One common use of auditory menus is in telephone information systems, as in "For flight arrival and departure information, please press 1...." Auditory menus are also incorporated into eyes-free interfaces that improve accessibility for blind users (Raman, 1997), and are likely to be incorporated into automotive information systems.

Though there are clearly many differences between visual and auditory perception, there are also similarities between the two. For example, vision and audition both impose a delay between the onset of the physical stimulus and the arrival of stimulus properties in WM, and both allow new stimuli to begin being processed while earlier stimuli are still being processed further down the pipeline. These similarities allow the DSVDH model to inform the design of auditory menus.

The DSVDH model demonstrates that people do not decide on one menu item before moving to the next, but instead move to and examine the next item before deciding on the previous item. Such a highly efficient search can made possible with an auditory menu by giving users a "forward" button that interrupts the current option and immediately starts the next, a "backward" button that interrupts and immediately starts the previous option, and a "select" button to select the current option. Resnick and Virzi (1993) built such a system and found that people could select a name from a list significantly faster with their new "skip and scan" menu than with the more typical "For Joe, press 1. For Michelle, press 2....".

The parallel processing models suggest that a "skip and scan" menu system could be optimized if users could be confident that the first ≈250 msec of a option would be enough to identify the option. This could be done by putting the key syllable first, similar to the above recommendation for visual menus, and also by providing other auditory cues such as different voices and sound effects. Users could then press the "forward" button four times a second and, as people seemed to be doing in the visual menu experiments, move through an audio menu at high speed, not waiting to evaluate each item before

moving to the next, and find the target as quickly as possible. Instant access to the "backward" command is critical in such an interface because, as in the menu models above, it is possible that a user might skip over the target before realizing that they had done so, and need to backup to select the target.

5.5 New Insights into Manual Motor Control

The models presented here contribute to an understanding of fundamental aspects of manual motor control, specifically with respect to interface analysis and design. The DSVDH and ILPCC models both provide further validation of standard Fitts' coefficients of 100 for a mouse point and 140 for a mouse point with the mouse button depressed. The value of 100 is the only Fitts coefficient in the literature proposed for *predicting* mouse movement time (Card et al., 1983). The value of 140 is the only coefficient available in the literature for a pure pointing task with the mouse button depressed (Walker et al., 1993). The DSVDH model and the ILPCC model use these coefficients and successfully generate predictions that account for the "button depression effect" observed by Nilsen (1991), as can be seen in Figure 5.4. The models thus provide independent validation that these are appropriate values for using Fitts' law for a priori predictions of movement times for mouse points.

The models of numerically ordered menus explore the possibility that Nilsen's data could be explained by simply raising the Fitts' coefficients. One reason that such exploration is at all feasible is that the literature reports a wide variety of Fitts' coefficients for predicting mouse pointing times. Even though Fitts' law is regarded by human-computer interaction researchers as an equation for predicting mouse pointing time, Fitts' coefficients are usually treated as free parameters and set *after* analyzing a new set of data. There has not been a systematic effort to calibrate Fitts' law for a variety pointing devices and tasks and, as a result, Fitts' law has questionable utility for making a priori predictions. A more systematic effort is needed to collect consistent and reliable coefficients.

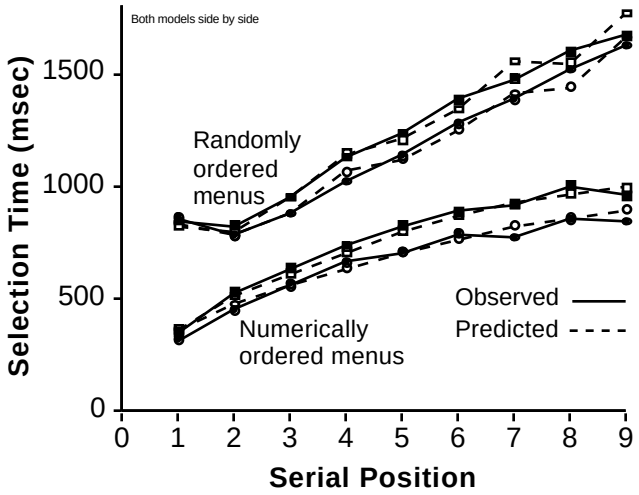

Figure 5.4. The DSVDH and ILPCC models' predictions for, respectively, Nilsen's randomly and numerically ordered menu data. All are collapsed by menu length and expanded by menu style. For each ordering, the "button depression effect" is the difference between the walking (■) and click-open (●) selection times. The button depression effect is explained very well by Fitts' coefficients of 100 and 140.

5.6 Future Modeling of Menu Search

Future improvements to the menu models will increase their predictive power and provide new insights with which to improve software and hardware design. One future research goal is to precisely model the new empirical data from Byrne et al. (1999) and Nilsen and Evans (1999).

Another important research challenge is to build a model that incorporates elements of both random and systematic search into the same trial, and still accounts for Nilsen's (1991) data. The DSVDH model assumes people use either completely random or completely systematic search strategies for each trial. This assertion merits investigation.

One way that a model could incorporate both random and systematic search into a single trial would be to use a strategy that makes a maximally efficient downward foveal sweep, as illustrated in Figure 3.7, but adds random noise to each saccade distance. When a saccade distance is too great, items would getting skipped. An additional sweep would

be required to capture the skipped items. If every item in the menu has a chance of getting skipped, this would add an element of randomness to every position. The randomness would not be part of a deliberate strategy, but a result of a noisy systematic strategy.

A second way that randomness could be incorporated into a maximally efficient foveal sweep would be to limit the number of objects whose movement to visual WM can be initiated during each fixation. For example, even if more than three items fit into the fovea during a fixation, perhaps only three randomly chosen items would begin moving to WM from that fixation; additional items, though in the fovea, would get skipped and require an additional sweep to be perceived.

A third way that randomness could be added to a systematic scan would be to vary the size of the fovea from saccade to saccade. EPIC models currently set the fovea as a circular region with a radius of one degree of visual angle. Varying this radius would allow models, across trials, to simulate a more gradual degradation of property availability around the edges of the fovea and to more realistically model human vision. Nilsen and Evans (1999) suggest this modification to the DSVDH model to explain new empirical data that they collected.

5.7 Future Modeling of General Visual Search

An important component of future research will be to build models of more general visual search tasks. The menu models in this dissertation provide an excellent foundation for building models of more general visual search.

In a more general visual search task, visual objects are more widely dispersed across the screen than in a menu task. Figures 5.5 and 5.6 illustrate more general visual search tasks. The task in Figure 5.5 is a real-world task; in Figure 5.6, an experimental task. Though one might expect that an experiment using the task in 5.6 would be relatively uninformative because of the simplicity of the task, reliable data from such an experiment will reveal fundamental aspects of visual search and guide model-building.

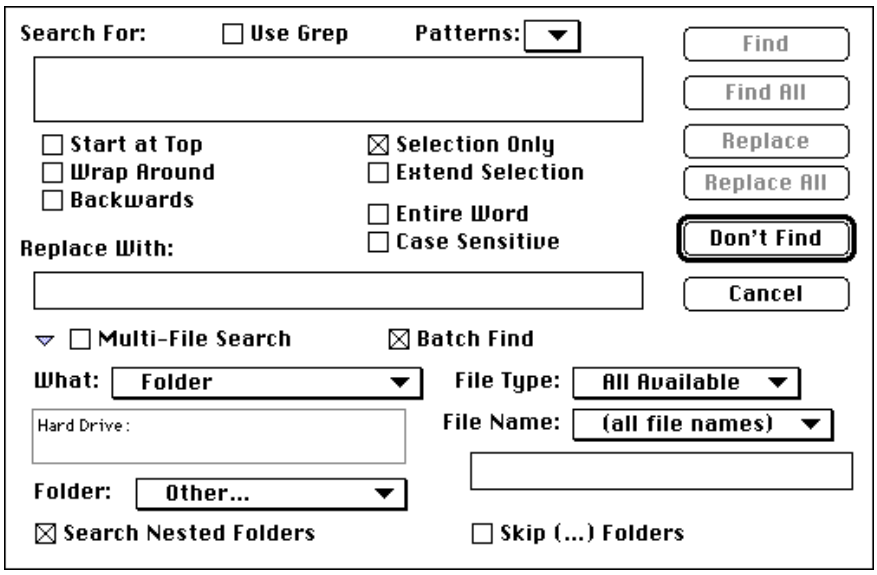

Figure 5.5. Find the "Case Sensitive" check box. This is a more general visual search task than finding an item in a pull-down menu. (From BBEdit 3.5.2 by Bare Bones Software)

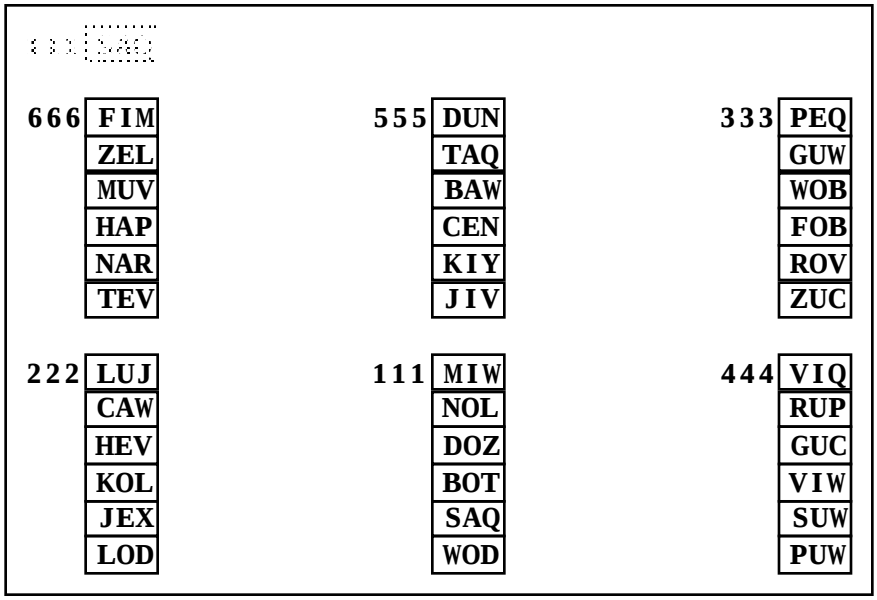

Figure 5.6. Find the "SAQ" object in the "111" group. This is a less real-world task than that shown in Figure 5.5. But reliable data from such a task can reveal fundamental aspects of visual search.

Though visual search has been studied at length (see Section 2.2), there are few if any available experimental data sets that lend themselves to building models of the same detail and accuracy as the menu selection models presented in this dissertation. One

shortcoming of most published data sets is that selection times are usually averaged across all positions on a screen. Nilsen's data were particularly useful for revealing search strategies because they included average selection times for every Serial Position.

Nilsen's experiment was also particularly useful for model-building because it manipulated few factors—ordering, length, and style (walking versus click-open)—and because these few manipulations produced reliable effects in the data that were sufficiently detailed and distinct to guide the model-building process. These effects included the menu length effect, Serial Position effect, Position 1 effect, and others.

A challenge in designing a more general visual search experiment will be to select and manipulate factors that, like Nilsen's, produce reliable effects in the data and guide the model-building process. Here is a list of each factor that might be manipulated, why it is of particular interest, the effect that would likely be observed when varying the factor, and how the effect would likely be explained in a model:

1. *Factor:* Number of items.

   Why of interest: This should produce an effect akin to the menu length effect in Nilsen's experiment, which proved invaluable for revealing underlying search processes.

   Likely effect: More items should increase overall search time.

   How to model: Extend the existing menu models to search in two dimensions.

2. *Factor:* Group titles present or absent.

   Why of interest: This factor will introduce another level of search. There should be one search for the group, and then a second search within the group for the target.

   Likely effect: When group titles are available, search should be faster.

   How to model: When titles are present, conduct an initial search that only examines the group headings. When titles are absent, search every item.

3. *Factor:* Alignment.

   Why of interest: A basic principle of graphic design is that visual objects should be arranged in a highly regular pattern, such as on a grid that is consistent from page to page or screen to screen (Müller-Brockmann, 1981). Varying the extent to which objects are aligned on

a grid might affect aspects of visual search.

Likely effect: Good alignment should be faster.

How to model: When objects are poorly aligned, additional saccades may be required to foveate all objects. Also, there may be fewer identical consecutive saccades, so EPIC will have fewer opportunities to re-use the ocular-motor program from the previous saccade, and more saccade preparation time will be required.

4. *Factor:* Spacing.

   Why of interest: Varying the visual angle between adjacent items should affect the number of eye movements required to examine every item on a screen, and thus overall search time. Nilsen and Evans (1999) recently demonstrated that spacings of one and two degrees do not produce dramatically different search times. More investigation is in order.

   Likely effect: More space between items should increase overall search time.

   How to model: Just extend the existing menu models to search in two dimensions. The models already require more eye movements to get all of the objects into the fovea at least once when menu items are more spread out.

5. *Factor:* Horizontal versus vertical arrangements.

   Why of interest: As discussed in Section 2.3.2, vertical lists of words can be searched more rapidly that horizontal lists. Collecting precise data that demonstrates this phenomena could lead to a detailed explanation of the source of the phenomena.

   Likely effect: Vertical lists will be faster.

   How to model: Just extend the existing menu models to search in two dimensions. Horizontal lists will require more eye movements to get all of the objects into the fovea at least once.

6. *Factor:* The type of objects searched.

   Why of interest: This should help to reveal why some kinds of objects can be found more rapidly than others, such as icons versus text.

   Likely effect: Some kinds objects will be found more quickly than others.

   How to model: Calibrate the perceptual encoding availabilities and delays for each kind of object.

7. *Factor:* Use of primary features.

   Why of interest: Primary features such as size, color, shape, or orientation are known to “pop out.” A comprehensive search model would have to account for this phenomena.

   Likely effect: Targets that can be distinguished by a single primary feature will be found quickly.

> *How to model:* Make primary features available outside of the fovea. Interrupt any search in progress as soon as a primary feature distinguishing the target appears in visual WM.

Modeling new data sets will be challenging. But lessons learned from the menu models will help to guide the process. Once a sufficient body of visual search data has been modeled, the various models and strategies can be combined into a visual search prediction tool.

5.8 Building a Screen Layout Analysis Tool

Insights gained from the modeling of more general visual search tasks will inform the design and construction of a screen layout analysis tool that will contribute to fundamental improvements in the design of human-computer interfaces. This tool will evaluate screen layouts based on the amount of time required to find information on the screen, and provide software designers with useful feedback as they design their software. The tool will take as input a definition of a screen layout and a visual search task. The tool will provide as output a prediction of the time required for the user to execute the task. Designers will use this information to compare alternative designs and to meet system requirements specifications.

The current menu models provide the basics for the search engine of such a tool. The inputs to the tool—a screen layout and a search task—could be translated into inputs to an EPIC model. The EPIC model could then generate a prediction that would be reported to the designer.

Such an approach to building a predictive tool would improve on the systems developed by Tullis (1988) and Sears (1993), which were discussed in Section 2.2.4. The tool would improve on Tullis' Display Analysis Program (DAP) by making more specific predictions for a wider variety of tasks. DAP only predicts for alphanumeric screens, and only makes one time prediction per screen. The tool would improve on Sears' Layout Appropriateness (LA) by predicting search times rather than just computing an arbitrary

cost for each layout that is then compared to the cost of a theoretically LA-optimal layout.

The proposed tool would most closely resemble Lohse's (1993) tool for Understanding Cognitive Information Engineering (UCIE). UCIE predicts how long a person will take to answer a question by studying a graph or table. Just like the models presented here, UCIE accounts for specific, detailed aspects of perception and cognition such as eye movements and the relative level of difficulty to acquire information in each glance. UCIE generates its predictions by executing a simulation of all of the processing required for a task.

The proposed tool would improve upon UCIE by incorporating more subtle and detailed aspects of visual search, such as the ability to perceive multiple objects with the same fixation, to pipeline several visual objects to visual WM at the same time, and to overlap perceptual and motor activities. A tool based on more detailed models should provide more accurate predictions of visual search.

The tool will work as follows. First, the designer will propose a screen layout and a specific task or set of tasks. For example, given the web page shown in Figure 5.7, how long will it take for someone to find all of the links to other pages? Second, an EPIC model will be semi-automatically constructed and executed as follows:

1. A digital bitmap of the screen layout is imported into the tool and encoded into visual objects or regions, such as those shown in Figure 5.8. In early versions of the tool, the designer will manually specify the boundaries. In later versions, the process will be more automated.
2. The features of the screen objects and regions defined in step 1 are defined, features such as color and shape. In early versions of the tool, the designer will manually enter these descriptions into a text file. In later versions, this step will eventually be somewhat automated.
3. The designer lists all of the meanings of non-text objects that he or she expects the user to know, such as meanings of icons. This forces the designer to articulate his or her assumptions, which can then be critiqued separately.
4. Based on the visual information provided in steps 1, 2, and 3, a description of each item is deposited into EPIC's physical space.
5. Standard visual recoding delays, taken from previous modeling efforts, are imposed for each visual feature.

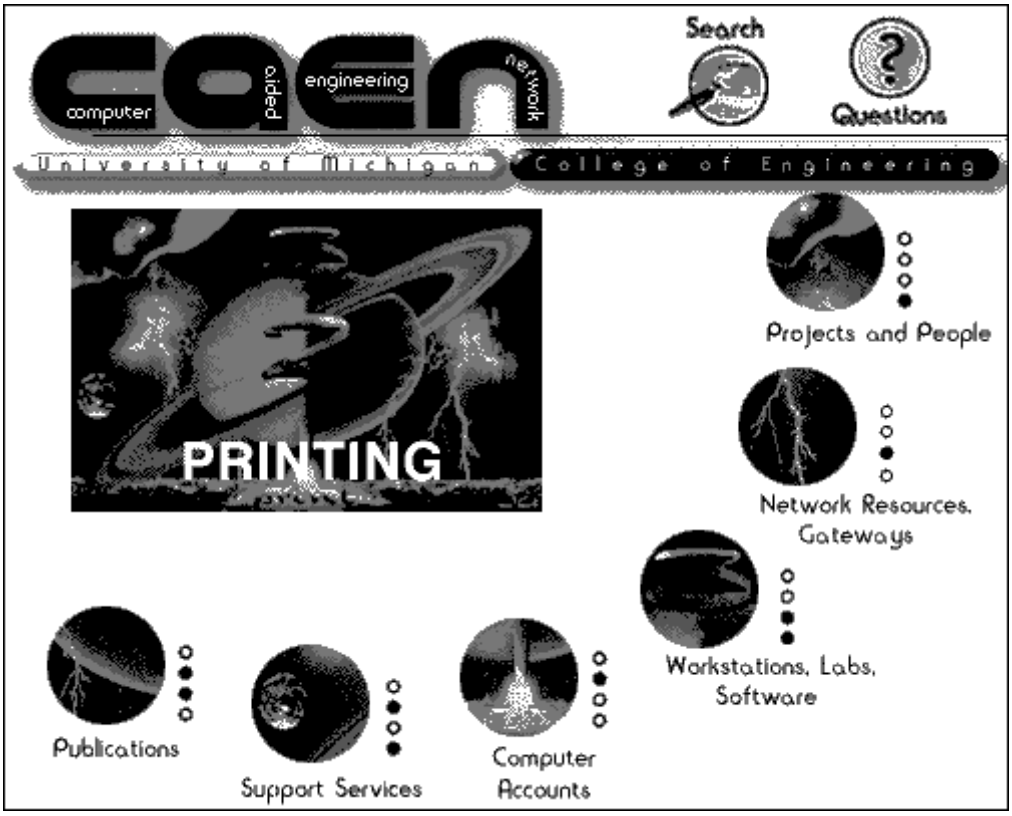

Figure 5.7. A screen layout that could be used as an input to a screen layout analysis tool. This is the main web page for undergraduate engineering computer support services at the University of Michigan in early 1997. (http://www.engin.umich.edu/caen/, 3/10/97)

6. A general visual search strategy is applied to the task. The strategy includes the following components:
   a. For the most part, conduct maximally efficient foveal sweeps of the layout but, occasionally, make a random jump to a randomly chosen item on the screen.
   b. Move the eyes as quickly as possible without waiting to evaluate each item.
   c. Consider items in parallel whenever possible.
   d. Decide where to move eyes next based on proximity, whether or not the items have been in the fovea yet, and primary features such as size, shape, and color.
   e. Halt the search as soon as the target text or the icon corresponding to the target text is found.

The EPIC model will execute and the search strategy will halt when the task is completed. The model will be executed several times and the predicted task execution times from all executions will be averaged. The tool will then report this average prediction as

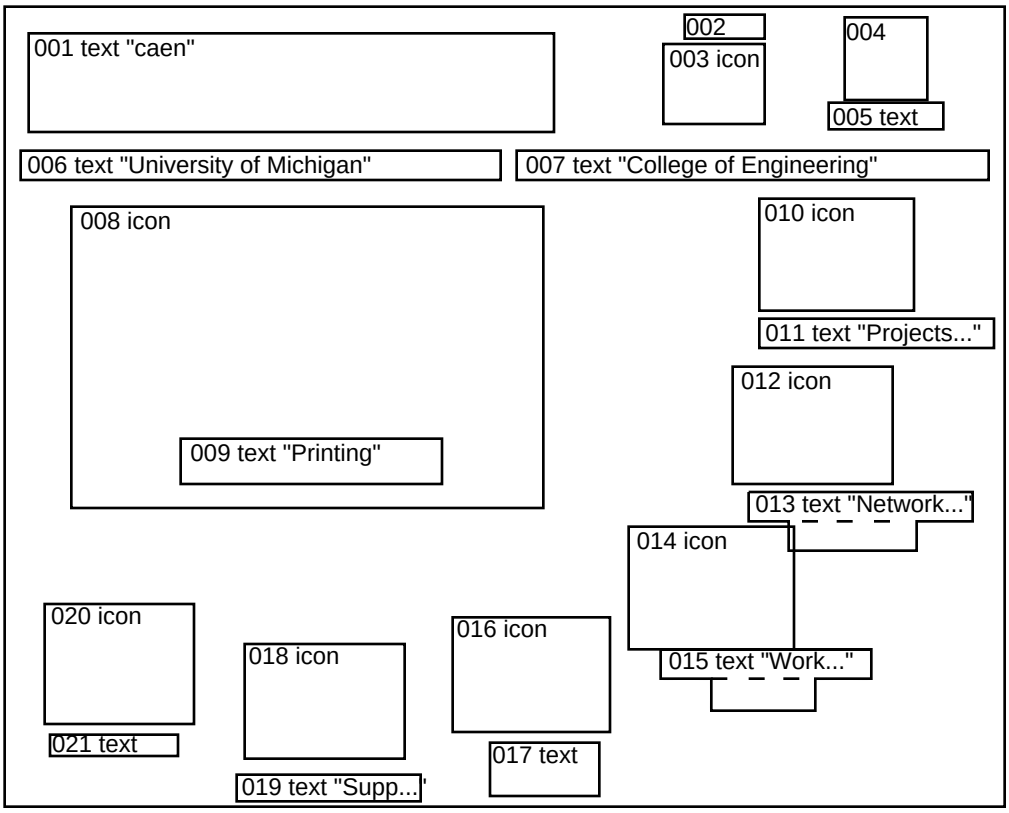

Figure 5.8. The visual regions and objects corresponding to the screen layout shown in Figure 5.7. This is spatial information that would be used by the screen layout analysis tool to predict visual search times.

the output of the tool. Different screen layouts can be compared and the most efficient one chosen.

5.9 Concluding Remarks

When using a computer, a substantial amount of time is spent looking for things in menus and elsewhere on the computer screen. As the field of human-computer interaction gains a better understanding of the perceptual, cognitive, and motor processes that people use when they conduct a visual search, researchers can better advise practitioners how to design menus and computer layouts that are easier to use. The advice can be in the form of explanations of human processes, theoretically-based guidelines, and model-based predictive tools.

By assembling what is currently known about visual search and menu search, and presenting the first empirically validated computational models of the perceptual, cognitive, and motor processes involved in the visual search of pull-down menus, this dissertation makes a contribution towards providing a scientific foundation for designing computer menus and screen layouts that are easier to search, and thus computers that are easier to use.

APPENDIX A

THE SERIAL PROCESSING RANDOM SEARCH STRATEGY

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.2.2.PRS By Anthony Hornof
;;  8/22/96
;;  Random, fixate on each, serial decision.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))
 (DELDB (VISUAL ?OBJECT DETECTION ONSET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (USE-ONLY-ONE ?OBJECT)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-TO-GO-BOX
```

```
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
```

```
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)
 (DELDB (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU))
 (ADDDB (STEP SACCADE TO FIRST RANDOMLY CHOSEN ITEM))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU))
 (ADDDB (STEP SACCADE TO FIRST RANDOMLY CHOSEN ITEM))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-POINT
;; As soon as possible after press or first punch, prepare for mouse movement.
IF
((GOAL DO MENU TASK)
 (STEP SACCADE TO FIRST RANDOMLY CHOSEN ITEM)
 (VISUAL ?FIRST-ITEM IS-BELOW NOTHING)
 (VISUAL ?FIRST-ITEM IS-ABOVE ?SECOND-ITEM)
 (WM CURSOR IS ?CURSOR)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PREPARE POINT RIGHT ?CURSOR ?SECOND-ITEM)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SACCADE-TO-FIRST-RANDOMLY-CHOSEN-ITEM
;; Saccade to any item.
IF
((GOAL DO MENU TASK)
 (STEP SACCADE TO FIRST RANDOMLY CHOSEN ITEM)
 (VISUAL ?OBJECT IN-MENU YES)
 (RANDOMLY-CHOOSE-ONE ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 )
THEN
((DELDB (STEP SACCADE TO FIRST RANDOMLY CHOSEN ITEM))
 (ADDDB (STEP RANDOM VISUAL SEARCH))
 (ADDDB (WM CURRENT-ITEM IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(TARGET-IS-NOT-LOCATED-SACCADE-TO-ANOTHER-RANDOMLY-CHOSEN-ITEM
;; If this is NOT the target, then continue random search.
IF
((GOAL DO MENU TASK)
 (STEP RANDOM VISUAL SEARCH)
 (WM CURRENT-ITEM IS ?CURRENT-OBJECT)
 (VISUAL ?CURRENT-OBJECT LABEL ?NT)  ;; Wait for text to appear.
 (NOT (WM TARGET-TEXT IS ?NT))       ;; It is not the target text.
 (VISUAL ?NEXT-OBJECT IN-MENU YES)        ;; Get ready for next random saccade.
 (NOT (WM CURRENT-ITEM IS ?NEXT-OBJECT))
 (RANDOMLY-CHOOSE-ONE ?NEXT-OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
```

```
THEN
((DELDB (WM CURRENT-ITEM IS ?CURRENT-OBJECT))
 (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(TARGET-IS-LOCATED-MOVE-GAZE-AND-CURSOR-TO-TARGET
;; Decides you found the item during the visual sweep.
IF
((GOAL DO MENU TASK)
 (STEP RANDOM VISUAL SEARCH)
 (WM CURRENT-ITEM IS ?TARGET-OBJECT) ;; To distinguish from the precue.
 (VISUAL ?TARGET-OBJECT LABEL ?T)    ;; Wait for text to appear.
 (WM TARGET-TEXT IS ?T)              ;; It IS the target text
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP RANDOM VISUAL SEARCH))
 (ADDDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (SEND-TO-MOTOR OCULAR MOVE ?TARGET-OBJECT)
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;these rules clean up whatever needs to be cleaned up after the response
(CLEANUP-STEP-CLEANUP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP))
THEN
((DELDB (STEP CLEANUP))))

(CLEANUP-TARGET-OBJECT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
```

```
 (WM TARGET-OBJECT IS ?OBJECT))
THEN
((DELDB (WM TARGET-OBJECT IS ?OBJECT))))

(CLEANUP-CURRENT-ITEM
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM CURRENT-ITEM IS ?X))
THEN
((DELDB (WM CURRENT-ITEM IS ?X))))

(CLEANUP-PRECUE
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM PRECUE IS ?X))
THEN
((DELDB (WM PRECUE IS ?X))))

(CLEANUP-TARGET-TEXT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-TEXT IS ?X))
THEN
((DELDB (WM TARGET-TEXT IS ?X))))
```

APPENDIX B

THE SERIAL PROCESSING SYSTEMATIC SEARCH STRATEGY

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.2.1.PRS By Anthony Hornof
;;  8/21/96
;;  Systematic top-to-bottom, fixate on each, serial decision.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))
 (DELDB (VISUAL ?OBJECT DETECTION ONSET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (USE-ONLY-ONE ?OBJECT)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-TO-GO-BOX
```

```
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
```

```
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)
 (DELDB (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU))
 (ADDDB (STEP FIX GAZE ON TOP MENU ITEM))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP PRESS OR PUNCH MOUSE-BUTTON TO SHOW MENU))
 (ADDDB (STEP FIX GAZE ON TOP MENU ITEM))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-POINT
;; As soon as possible after press or first punch, prepare for mouse movement.
IF
((GOAL DO MENU TASK)
 (STEP FIX GAZE ON TOP MENU ITEM)
 (VISUAL ?OBJECT DETECTION ONSET)
 (VISUAL ?OBJECT IS-BELOW NOTHING)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (VISUAL ?OBJECT IS-ABOVE ?PLY-PREPARE-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PREPARE POINT RIGHT ?CURSOR-OBJECT ?PLY-PREPARE-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(FIX-GAZE-ON-TOP-ITEM
;; Move eye to FIRST item on list.
IF
((GOAL DO MENU TASK)
 (STEP FIX GAZE ON TOP MENU ITEM)
 (VISUAL ?OBJECT DETECTION ONSET)
 (VISUAL ?OBJECT IS-BELOW NOTHING)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP FIX GAZE ON TOP MENU ITEM))
 (ADDDB (STEP VISUAL-SEARCH))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)
 (ADDDB (WM CURRENT-ITEM IS ?OBJECT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(TARGET-IS-NOT-LOCATED-SACCADE-ONE-ITEM
;; If this is NOT the target, then continue down the list.
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SEARCH)
 (WM CURRENT-ITEM IS ?OBJECT)
 (VISUAL ?OBJECT IS-ABOVE ?NEXT-OBJECT)
 (NOT (VISUAL ?OBJECT IS-ABOVE NOTHING))
 (MOTOR OCULAR PROCESSOR FREE)
 (VISUAL ?OBJECT LABEL ?NT)     ;; Wait for text to appear.
 (NOT (WM TARGET-TEXT IS ?NT))  ;; It is not the target text.
 )
```

```
THEN
((DELDB (WM CURRENT-ITEM IS ?OBJECT))
 (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(TARGET-IS-LOCATED-MOVE-GAZE-AND-CURSOR-TO-TARGET
;; Decides you found the item during the visual sweep.
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SEARCH)
 (WM CURRENT-ITEM IS ?TARGET-OBJECT) ;; To distinguish from the precue.
 (VISUAL ?TARGET-OBJECT LABEL ?T)    ;; Wait for text to appear.
 (WM TARGET-TEXT IS ?T)              ;; It IS the target text
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP VISUAL-SEARCH))
 (ADDDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (SEND-TO-MOTOR OCULAR MOVE ?TARGET-OBJECT)
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;these rules clean up whatever needs to be cleaned up after the response
(CLEANUP-STEP-CLEANUP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP))
THEN
((DELDB (STEP CLEANUP))))

(CLEANUP-TARGET-OBJECT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
```

```
 (WM TARGET-OBJECT IS ?OBJECT))
THEN
((DELDB (WM TARGET-OBJECT IS ?OBJECT))))

(CLEANUP-CURRENT-ITEM
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM CURRENT-ITEM IS ?X))
THEN
((DELDB (WM CURRENT-ITEM IS ?X))))

(CLEANUP-PRECUE
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM PRECUE IS ?X))
THEN
((DELDB (WM PRECUE IS ?X))))

(CLEANUP-TARGET-TEXT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-TEXT IS ?X))
THEN
((DELDB (WM TARGET-TEXT IS ?X))))
```

APPENDIX C

THE PARALLEL PROCESSING RANDOM SEARCH STRATEGY

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.2.6.PRS By Anthony Hornof
;;  Random, anticipatory first saccade, next item always out of fovea,
;;           parallel decision
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR OCULAR DISABLE CENTERING)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))
 (DELDB (VISUAL ?OBJECT DETECTION ONSET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (USE-ONLY-ONE ?OBJECT)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(MOVE-CURSOR-TO-GO-BOX
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP CLICK GO BOX AND MOVE GAZE TO FIRST RANDOM MENU LOCATION))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK GO BOX AND MOVE GAZE TO FIRST RANDOM MENU LOCATION)
 (WM GO BOX IS ?OBJECT)
```

```
 (MOTOR MANUAL PROCESSOR FREE)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)
 (ADDDB (WM CURRENT-ITEM IS START-POSITION))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK GO BOX AND MOVE GAZE TO FIRST RANDOM MENU LOCATION)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (ADDDB (WM CURRENT-ITEM IS START-POSITION))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SACCADE-TO-RANDOM-LOCATION
;; Saccade to the randomly chosen named location of the first saccade.
IF
((GOAL DO MENU TASK)
 (STEP CLICK GO BOX AND MOVE GAZE TO FIRST RANDOM MENU LOCATION)
 (MOTOR OCULAR PROCESSOR FREE)
 )
THEN
((DELDB (STEP CLICK GO BOX AND MOVE GAZE TO FIRST RANDOM MENU LOCATION))
 (ADDDB (STEP RANDOM SEARCH FOR TARGET))
 (SEND-TO-MOTOR OCULAR MOVE FIRST-FIXATION-LOCATION)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-POINT
;; As soon as possible after punch, prepare for mouse movement.
IF
((GOAL DO MENU TASK)
 (STEP RANDOM SEARCH FOR TARGET)
 (WM CURRENT-ITEM IS START-POSITION)
 (VISUAL ?FIRST-ITEM IS-BELOW NOTHING)         ; critical path
 (VISUAL ?FIRST-ITEM IS-ABOVE ?SECOND-ITEM)    ; critical path
 (WM CURSOR IS ?CURSOR)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PREPARE POINT RIGHT ?CURSOR ?SECOND-ITEM)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(FIRST-SACCADE-TO-RANDOM-ITEM-NOT-IN-FOVEA
;; Make first saccade to random item not currently in the fovea.
IF
((GOAL DO MENU TASK)
 (STEP RANDOM SEARCH FOR TARGET)
 (WM CURRENT-ITEM IS START-POSITION)
 (VISUAL ?OBJECT IN-MENU YES)
 (VISUAL ?OBJECT FOVEA NO)            ;; Next object can't be in the fovea now.
 (RANDOMLY-CHOOSE-ONE ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 )
THEN
((DELDB (WM CURRENT-ITEM IS START-POSITION))
 (ADDDB (WM CURRENT-ITEM IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SACCADE-TO-RANDOM-ITEM-NOT-IN-FOVEA
;; Saccade to random item not currently in the fovea.
IF
((GOAL DO MENU TASK)
 (STEP RANDOM SEARCH FOR TARGET)
 (WM CURRENT-ITEM IS ?LAST-OBJECT)
 ;; Wait until it has been established that the new item is in the fovea.
 ;; Otherwise, the foveal info of the other objects is not correct yet either.
 (VISUAL ?LAST-OBJECT FOVEA YES)
 (VISUAL ?OBJECT IN-MENU YES)
 (VISUAL ?OBJECT FOVEA NO)  ;; Object can't be in the fovea now.
 (RANDOMLY-CHOOSE-ONE ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (WM CURRENT-ITEM IS ?LAST-OBJECT))
 (ADDDB (WM CURRENT-ITEM IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(TARGET-IS-LOCATED-STOP-SCANNING
;; Decides you found the item during the visual sweep.  10/26/95 -ajh
;; This extra rule prevents two ocular commands being sent at same time.
IF
((GOAL DO MENU TASK)
 (STEP RANDOM SEARCH FOR TARGET)
 (WM TARGET-TEXT IS ?T)
 (VISUAL ?TARGET-OBJECT LABEL ?T)
 (VISUAL ?TARGET-OBJECT IN-MENU YES)   ;; Don't react to the precue!
 )
THEN
((DELDB (STEP RANDOM SEARCH FOR TARGET))
 (ADDDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
 (ADDDB (WM TARGET-OBJECT IS ?TARGET-OBJECT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SCANNING-IS-STOPPED-MOVE-GAZE-AND-CURSOR-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET)
 (WM TARGET-OBJECT IS ?TARGET-OBJECT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
 (ADDDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (SEND-TO-MOTOR OCULAR MOVE ?TARGET-OBJECT)
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
```

```
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;these rules clean up whatever needs to be cleaned up after the response
(CLEANUP-STEP-CLEANUP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP))
THEN
((DELDB (STEP CLEANUP))))

(CLEANUP-TARGET-OBJECT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-OBJECT IS ?OBJECT))
THEN
((DELDB (WM TARGET-OBJECT IS ?OBJECT))))

(CLEANUP-CURRENT-ITEM
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM CURRENT-ITEM IS ?X))
THEN
((DELDB (WM CURRENT-ITEM IS ?X))))

(CLEANUP-PRECUE
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM PRECUE IS ?X))
THEN
((DELDB (WM PRECUE IS ?X))))

(CLEANUP-TARGET-TEXT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-TEXT IS ?X))
THEN
((DELDB (WM TARGET-TEXT IS ?X))))
```

APPENDIX D

THE PARALLEL PROCESSING SYSTEMATIC SEARCH STRATEGY

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.2.7.PRS By Anthony Hornof
;;  8/24/96
;;  Systematic top-to-bottom, fixate on each, serial decision.
;;  Eye starts on any randomly chosen item that insures the first item
;;    will fall in fovea.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR OCULAR DISABLE CENTERING)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))
 (DELDB (VISUAL ?OBJECT DETECTION ONSET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (USE-ONLY-ONE ?OBJECT)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))
```

```
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-TO-GO-BOX
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP CLICK GO BOX AND MOVE GAZE TO FIRST MENU LOCATION))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Start trial.  Have subject wait long enough for ocular motor
;; processor to be free, to make sure the next rule is not delayed
;; due to pre-trial activity.  The subject can wait here anyway as
```

```
;; they memorize the precue.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK GO BOX AND MOVE GAZE TO FIRST MENU LOCATION)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)
 (ADDDB (STEP PREPARE POINT TO TARGET))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK GO BOX AND MOVE GAZE TO FIRST MENU LOCATION)
 (WM GO BOX IS ?OBJECT)
 (MOTOR MANUAL PROCESSOR FREE)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (ADDDB (STEP PREPARE POINT TO TARGET))
 (DELDB (WM GO BOX IS ?OBJECT))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SACCADE-TO-FIRST-LOCATION
;; Move eye to named location of the first fixation.
IF
((GOAL DO MENU TASK)
 (STEP CLICK GO BOX AND MOVE GAZE TO FIRST MENU LOCATION)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP CLICK GO BOX AND MOVE GAZE TO FIRST MENU LOCATION))
 (ADDDB (STEP VISUAL-SWEEP))
 (SEND-TO-MOTOR OCULAR MOVE FIRST-FIXATION-LOCATION)
 (ADDDB (WM CURRENT-ITEM IS FIRST-FIXATION))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-POINT-NEW
;; As soon as possible after press or first punch, prepare for mouse movement.
IF
((GOAL DO MENU TASK)
 (STEP PREPARE POINT TO TARGET)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP PREPARE POINT TO TARGET))
 (SEND-TO-MOTOR MANUAL PREPARE POINT RIGHT ?CURSOR-OBJECT ITEM-LOCATION-3)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SACCADE-TO-NEXT-SWEEP-ITEM
;; Saccade down in the list and stay in sweeping mode.
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SWEEP)
 (WM CURRENT-ITEM IS ?OBJECT)
 (NOT (VISUAL ?OBJECT IS-ABOVE NOTHING))  ;; sit on last item
 (DIFFERENT ?OBJECT ?NEXT-OBJECT)  ;; wait until new next is prepared
 (VISUAL GLOBAL-FEATURE NEXT-SWEEP-ITEM ?NEXT-OBJECT)
```

```
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (WM CURRENT-ITEM IS ?OBJECT))
 (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(TARGET-IS-LOCATED-STOP-SCANNING
;; Decides you found the item during the visual sweep.  10/26/95 -ajh
;; This extra rule prevents two ocular commands being sent at same time.
IF
((GOAL DO MENU TASK)
 (STEP VISUAL-SWEEP)
 (WM TARGET-TEXT IS ?T)
 (VISUAL ?TARGET-OBJECT LABEL ?T)
 (NOT (VISUAL ?TARGET-OBJECT IN-MENU NO))   ;; Don't react to the precue!
 )
THEN
((DELDB (STEP VISUAL-SWEEP))
 (ADDDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
 (ADDDB (WM TARGET-OBJECT IS ?TARGET-OBJECT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SCANNING-IS-STOPPED-MOVE-GAZE-AND-CURSOR-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET)
 (WM TARGET-OBJECT IS ?TARGET-OBJECT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
 (ADDDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (SEND-TO-MOTOR OCULAR MOVE ?TARGET-OBJECT)
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP RELEASE OR PUNCH MOUSE BUTTON)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP RELEASE OR PUNCH MOUSE BUTTON))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;these rules cleans stuff up after the response
(CLEANUP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-OBJECT IS ?OBJECT)
 (WM CURRENT-ITEM IS ?X)
 (WM PRECUE IS ?Y)
 (WM TARGET-TEXT IS ?Z))
THEN
((DELDB (WM CURRENT-ITEM IS ?X))
 (DELDB (WM TARGET-OBJECT IS ?OBJECT))
 (DELDB (STEP CLEANUP))
 (DELDB (WM PRECUE IS ?Y))
 (DELDB (WM TARGET-TEXT IS ?Z))))
```

APPENDIX E

THE IMMEDIATE LOOK, POINT, AND CLICK STRATEGY

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.3.22.PRS By Anthony Hornof
;;  5/28/98
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR OCULAR DISABLE CENTERING)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))
 (DELDB (VISUAL ?OBJECT DETECTION ONSET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (VISUAL ?OBJECT IN-MENU NO)    ;; so it does not look at a menu item
  (USE-ONLY-ONE ?OBJECT)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(MOVE-CURSOR-TO-GO-BOX
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP PREPARE EXACTLY CORRECT EYE MOVEMENT))
 (DELDB (WM GO BOX IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-EXACTLY-CORRECT-EYE-MOVEMENT
;; Prepare the eye movement to the exactly correct location.
IF
((GOAL DO MENU TASK)
 (STEP PREPARE EXACTLY CORRECT EYE MOVEMENT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
```

```
((DELDB (STEP PREPARE EXACTLY CORRECT EYE MOVEMENT))
 (ADDDB (STEP GET SET))
 (SEND-TO-MOTOR OCULAR PREPARE TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-SET
;; Make sure both processors are free.
IF
((GOAL DO MENU TASK)
 (STEP GET SET)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP GET SET))
 (ADDDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE GAZE DIRECTLY TO TARGET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK ON GO BOX)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)
 (DELDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE CURSOR TO TARGET))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK ON GO BOX)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE CURSOR TO TARGET))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-DIRECTLY-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE DIRECTLY TO TARGET)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE DIRECTLY TO TARGET))
 (SEND-TO-MOTOR OCULAR MOVE TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-DIRECTLY-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE CURSOR TO TARGET)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP MOVE CURSOR TO TARGET))
 (ADDDB (STEP CLICK ON TARGET))
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT TARGET-LOCATION-CORRECT)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK ON TARGET)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP CLICK ON TARGET))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK ON TARGET)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP CLICK ON TARGET))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;these rules clean up whatever needs to be cleaned up after the response
(CLEANUP-STEP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP))
THEN
((DELDB (STEP CLEANUP))))

(CLEANUP-PRECUE
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM PRECUE IS ?X))
THEN
((DELDB (WM PRECUE IS ?X))))

(CLEANUP-TARGET-TEXT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-TEXT IS ?X))
THEN
((DELDB (WM TARGET-TEXT IS ?X))))
```

APPENDIX F

THE IMMEDIATE LOOK, POINT, AND CLICK STRATEGY WITH SPECIAL CASE FOR POSITION 1

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.3.24.PRS By Anthony Hornof
;;  9/9/98
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR OCULAR DISABLE CENTERING)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))
 (DELDB (VISUAL ?OBJECT DETECTION ONSET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (VISUAL ?OBJECT IN-MENU NO)    ;; so it does not look at a menu item
  (USE-ONLY-ONE ?OBJECT)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-TO-GO-BOX
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP DECIDE IF SPECIAL CASE))
 (DELDB (WM GO BOX IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(DECIDE-YES-SPECIAL-CASE
IF
((GOAL DO MENU TASK)
 (STEP DECIDE IF SPECIAL CASE)
 (WM TARGET-TEXT IS 1))
THEN
```

```
((DELDB (STEP DECIDE IF SPECIAL CASE))
 (ADDDB (STEP CLICK ON GO BOX - SPECIAL CASE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(DECIDE-NO-SPECIAL-CASE
IF
((GOAL DO MENU TASK)
 (STEP DECIDE IF SPECIAL CASE)
 (NOT (WM TARGET-TEXT IS 1)))
THEN
((DELDB (STEP DECIDE IF SPECIAL CASE))
 (ADDDB (STEP PREPARE EXACTLY CORRECT EYE MOVEMENT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; RULES FOR *YES* SPECIAL CASE - TARGET POSITION 1.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU--TARGET-IS-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS CLICK-OPEN)
 (STEP CLICK ON GO BOX - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON GO BOX - SPECIAL CASE))
 (ADDDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU--TARGET-IS-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS WALKING)
 (STEP CLICK ON GO BOX - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON GO BOX - SPECIAL CASE))
 (ADDDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(PREPARE-TO-PUNCH-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS CLICK-OPEN)
 (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PREPARE PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-TO-RELEASE-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS WALKING)
 (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
```

```
THEN
((DELDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PREPARE RELEASE MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(PUNCH-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS CLICK-OPEN)
 (STEP CLICK ON TARGET - SPECIAL CASE)
 (VISUAL ?OBJECT IN-MENU YES)
 (VISUAL ?OBJECT LABEL 1)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))
 (SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS WALKING)
 (STEP CLICK ON TARGET - SPECIAL CASE)
 (VISUAL ?OBJECT IN-MENU YES)
 (VISUAL ?OBJECT LABEL 1)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))
 (SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; RULES FOR *NO* SPECIAL CASE - TARGET POSITIONS 2 THROUGH 9.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-EXACTLY-CORRECT-EYE-MOVEMENT
;; Prepare the eye movement to the exactly correct location.
IF
((GOAL DO MENU TASK)
 (STEP PREPARE EXACTLY CORRECT EYE MOVEMENT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP PREPARE EXACTLY CORRECT EYE MOVEMENT))
 (ADDDB (STEP GET SET))
 (SEND-TO-MOTOR OCULAR PREPARE TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-SET
;; Make sure both processors are free.
IF
((GOAL DO MENU TASK)
 (STEP GET SET)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
```

```
((DELDB (STEP GET SET))
 (ADDDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE GAZE DIRECTLY TO TARGET))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK ON GO BOX)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)
 (DELDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE CURSOR TO TARGET))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK ON GO BOX)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE CURSOR TO TARGET))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-DIRECTLY-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE DIRECTLY TO TARGET)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE DIRECTLY TO TARGET))
 (SEND-TO-MOTOR OCULAR MOVE TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-DIRECTLY-TO-TARGET
IF
((GOAL DO MENU TASK)
 (STEP MOVE CURSOR TO TARGET)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP MOVE CURSOR TO TARGET))
 (ADDDB (STEP CLICK ON TARGET))
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK ON TARGET)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP CLICK ON TARGET))
```

```
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK ON TARGET)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP CLICK ON TARGET))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;these rules clean up whatever needs to be cleaned up after the response
(CLEANUP-STEP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP))
THEN
((DELDB (STEP CLEANUP))))

(CLEANUP-PRECUE
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM PRECUE IS ?X))
THEN
((DELDB (WM PRECUE IS ?X))))

(CLEANUP-TARGET-TEXT
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-TEXT IS ?X))
THEN
((DELDB (WM TARGET-TEXT IS ?X))))
```

APPENDIX G

THE IMMEDIATE LOOK, POINT, CHECK AND CORRECT STRATEGY WITH SPECIAL CASE FOR POSITION 1

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NILSEN.1.3.34.PRS By Anthony Hornof
;;  9/1/98
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(CHOICE-Start
IF
((GOAL DO MENU TASK)
 (NOT (WM MENU TASK UNDERWAY)))
THEN
((ADDDB (WM MENU TASK UNDERWAY))
 (SEND-TO-MOTOR OCULAR DISABLE REFLEX)
 (SEND-TO-MOTOR OCULAR DISABLE CENTERING)
 (SEND-TO-MOTOR MANUAL RESET MEMORY)
 (ADDDB (STEP IDENTIFY-CURSOR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(START-CURSOR-TRACKING
IF
((GOAL DO MENU TASK)
 (STEP IDENTIFY-CURSOR)
 (VISUAL ?OBJECT SHAPE CROSS))
THEN
((DELDB (STEP IDENTIFY-CURSOR))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (WM CURSOR IS ?OBJECT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(LOOK-AT-GO-BOX
 ;; Just looks at the visual object that appears next.
 IF
 ((GOAL DO MENU TASK)
  (STEP WAIT FOR GO BOX)
  (VISUAL ?OBJECT DETECTION ONSET)
  (VISUAL ?OBJECT IN-MENU NO)    ;; so it does not look at a previous menu item
  (USE-ONLY-ONE ?OBJECT)   ;; If cursor is in GO box, the precue is already there.
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP WAIT FOR GO BOX))
  (ADDDB (STEP VERIFY GO BOX TEXT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(VERIFY-GO-BOX
 ;; Verify the new thing that appeared really is the GO box.
 IF
 ((GOAL DO MENU TASK)
  (STEP VERIFY GO BOX TEXT)
  (VISUAL ?OBJECT LABEL GO)
  (MOTOR OCULAR PROCESSOR FREE))
 THEN
 ((DELDB (STEP VERIFY GO BOX TEXT))
  (ADDDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (WM GO BOX IS ?OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-CURSOR-TO-GO-BOX
 IF
 ((GOAL DO MENU TASK)
  (STEP MOVE CURSOR TO GO BOX)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (WM GO BOX IS ?TARGET-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE)
  )
 THEN
 ((SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)
  (DELDB (STEP MOVE CURSOR TO GO BOX))
  (ADDDB (STEP MOVE GAZE TO TARGET PRECUE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-TARGET-PRECUE
;; After the precue has appeared with its onset, move the eyes to it.
;; But only move eyes to it if it is just above the GO-BOX.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO TARGET PRECUE)
 (WM GO BOX IS ?GO-BOX)
 (VISUAL ?OBJECT IS-ABOVE ?GO-BOX)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO TARGET PRECUE))
 (ADDDB (STEP GET TARGET PRECUE))
 (ADDDB (WM PRECUE IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-TARGET-PRECUE
;; Put the text of the precue object into WM as the target text.
IF
((GOAL DO MENU TASK)
 (STEP GET TARGET PRECUE)
 (WM PRECUE IS ?OBJECT)
 (VISUAL ?OBJECT LABEL ?PRECUE-TEXT))
THEN
((DELDB (STEP GET TARGET PRECUE))
 (ADDDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (WM TARGET-TEXT IS ?PRECUE-TEXT))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-BACK-TO-GO-BOX
;; Move gaze back in preparation to start the trial.
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE BACK TO GO BOX)
 (WM GO BOX IS ?OBJECT)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE BACK TO GO BOX))
 (ADDDB (STEP DECIDE IF SPECIAL CASE))
 (DELDB (WM GO BOX IS ?OBJECT))
 (SEND-TO-MOTOR OCULAR MOVE ?OBJECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(DECIDE-YES-THIS-IS-A-SPECIAL-CASE
IF
((GOAL DO MENU TASK)
 (STEP DECIDE IF SPECIAL CASE)
 (WM TARGET-TEXT IS 1))
THEN
```

```
((DELDB (STEP DECIDE IF SPECIAL CASE))
 (ADDDB (STEP CLICK ON GO BOX - SPECIAL CASE))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(DECIDE-NO-THIS-IS-NOT-A-SPECIAL-CASE
IF
((GOAL DO MENU TASK)
 (STEP DECIDE IF SPECIAL CASE)
 (NOT (WM TARGET-TEXT IS 1)))
THEN
((DELDB (STEP DECIDE IF SPECIAL CASE))
 (ADDDB (STEP PREPARE EYE MOVEMENT TO LOCATION WITH ERROR))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; RULES FOR *YES* SPECIAL CASE - TARGET POSITION 1.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU--TARGET-IS-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS CLICK-OPEN)
 (STEP CLICK ON GO BOX - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON GO BOX - SPECIAL CASE))
 (ADDDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU--TARGET-IS-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS WALKING)
 (STEP CLICK ON GO BOX - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON GO BOX - SPECIAL CASE))
 (ADDDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-TO-PUNCH-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS CLICK-OPEN)
 (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PREPARE PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-TO-RELEASE-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS WALKING)
 (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
```

```
((DELDB (STEP PREPARE TO CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (SEND-TO-MOTOR MANUAL PREPARE RELEASE MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS CLICK-OPEN)
 (STEP CLICK ON TARGET - SPECIAL CASE)
 (VISUAL ?OBJECT IN-MENU YES)
 (VISUAL ?OBJECT LABEL 1)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))
 (SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(RELEASE-MOUSE-BUTTON-ON-FIRST-MENU-ITEM
IF
((GOAL DO MENU TASK)
 (STRATEGY MENU STYLE IS WALKING)
 (STEP CLICK ON TARGET - SPECIAL CASE)
 (VISUAL ?OBJECT IN-MENU YES)
 (VISUAL ?OBJECT LABEL 1)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON TARGET - SPECIAL CASE))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))
 (SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; RULES FOR *NO* SPECIAL CASE - TARGET POSITIONS 2 THROUGH 9.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-EYE-MOVEMENT-TO-LOCATION-WITH-ERROR
IF
((GOAL DO MENU TASK)
 (STEP PREPARE EYE MOVEMENT TO LOCATION WITH ERROR)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP PREPARE EYE MOVEMENT TO LOCATION WITH ERROR))
 (ADDDB (STEP GET SET))
 (SEND-TO-MOTOR OCULAR PREPARE TARGET-LOCATION-WITH-ERROR)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(GET-SET
;; Make sure both processors are free.  Note that TWO steps are added here.
IF
((GOAL DO MENU TASK)
 (STEP GET SET)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP GET SET))
 (ADDDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP MOVE GAZE TO LOCATION WITH ERROR))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "PRESS/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PRESS-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK ON GO BOX)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP ATTEMPT TO POINT TO TARGET))
 (SEND-TO-MOTOR MANUAL PERFORM PRESS MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PUNCH-MOUSE-BUTTON-TO-SHOW-MENU
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK ON GO BOX)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP CLICK ON GO BOX))
 (ADDDB (STEP ATTEMPT TO POINT TO TARGET))
 (SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(MOVE-GAZE-TO-LOCATION-WITH-ERROR
IF
((GOAL DO MENU TASK)
 (STEP MOVE GAZE TO LOCATION WITH ERROR)
 (MOTOR OCULAR PROCESSOR FREE))
THEN
((DELDB (STEP MOVE GAZE TO LOCATION WITH ERROR))
 (SEND-TO-MOTOR OCULAR MOVE TARGET-LOCATION-WITH-ERROR)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(POINT-TO-TARGET-LOCATION-WITH-ERROR
;; Note that TWO steps are added here.
IF
((GOAL DO MENU TASK)
 (STEP ATTEMPT TO POINT TO TARGET)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP ATTEMPT TO POINT TO TARGET))
 (ADDDB (STEP PREPARE RELEASE OR PUNCH))
 (ADDDB (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT))
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT TARGET-LOCATION-WITH-
ERROR)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-TO-RELEASE-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP PREPARE RELEASE OR PUNCH)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PREPARE RELEASE MOUSE-BUTTON)
 (DELDB (STEP PREPARE RELEASE OR PUNCH))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(PREPARE-TO-PUNCH-MOUSE-BUTTON-ON-TARGET
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP PREPARE RELEASE OR PUNCH)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PREPARE PUNCH MOUSE-BUTTON)
 (DELDB (STEP PREPARE RELEASE OR PUNCH))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(FIRST-POINT-MISSES-MENU-ENTIRELY-SO-MAKE-A-SECOND-SACCADE-AND-POINT
IF
((GOAL DO MENU TASK)
 (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (VISUAL ?CURSOR-OBJECT POINTS-TO NOTHING)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT))
 (ADDDB (STEP CLICK ON TARGET AFTER SECOND POINT))
 (SEND-TO-MOTOR OCULAR MOVE TARGET-LOCATION-CORRECT)
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(FIRST-POINT-MISSES-TARGET-SO-MAKE-A-SECOND-SACCADE-AND-POINT
IF
((GOAL DO MENU TASK)
 (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT)
 (WM TARGET-TEXT IS ?TEXT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (VISUAL ?CURSOR-OBJECT POINTS-TO ?SOMETHING)
 (VISUAL ?SOMETHING LABEL ?OTHER-TEXT)
 (DIFFERENT ?TEXT ?OTHER-TEXT)
 (MOTOR OCULAR PROCESSOR FREE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT))
 (ADDDB (STEP CLICK ON TARGET AFTER SECOND POINT))
 (SEND-TO-MOTOR OCULAR MOVE TARGET-LOCATION-CORRECT)
 (SEND-TO-MOTOR MANUAL PERFORM POINT RIGHT ?CURSOR-OBJECT TARGET-LOCATION-CORRECT)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(FIRST-POINT-LANDS-ON-TARGET-SO-RELEASE-MOUSE-BUTTON
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT)
 (VISUAL ?TARGET-OBJECT LABEL ?TEXT)
 (VISUAL ?CURSOR-OBJECT POINTS-TO ?TARGET-OBJECT)
 (WM TARGET-TEXT IS ?TEXT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(FIRST-POINT-LANDS-ON-TARGET-SO-PUNCH-MOUSE-BUTTON
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT)
 (VISUAL ?TARGET-OBJECT LABEL ?TEXT)
 (VISUAL ?CURSOR-OBJECT POINTS-TO ?TARGET-OBJECT)
 (WM TARGET-TEXT IS ?TEXT)
 (WM CURSOR IS ?CURSOR-OBJECT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP FIGURE OUT WHERE FIRST POINT LANDED AND WHAT TO DO NEXT))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; THESE TWO RULES SHOULD MATCH EXCEPT FOR "RELEASE/PUNCH" AND "MENU STYLE".
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SECOND-POINT-LANDS-ON-TARGET-SO-RELEASE-MOUSE-BUTTON
IF
((STRATEGY MENU STYLE IS WALKING)
 (GOAL DO MENU TASK)
 (STEP CLICK ON TARGET AFTER SECOND POINT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM RELEASE MOUSE-BUTTON)
 (DELDB (STEP CLICK ON TARGET AFTER SECOND POINT))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(SECOND-POINT-LANDS-ON-TARGET-SO-PUNCH-MOUSE-BUTTON
IF
((STRATEGY MENU STYLE IS CLICK-OPEN)
 (GOAL DO MENU TASK)
 (STEP CLICK ON TARGET AFTER SECOND POINT)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM PUNCH MOUSE-BUTTON)
 (DELDB (STEP CLICK ON TARGET AFTER SECOND POINT))
 (ADDDB (STEP WAIT FOR GO BOX))
 (ADDDB (STEP CLEANUP))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;These rules clean up whatever needs to be cleaned up after the response
(CLEANUP-STEP
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP))
THEN
((DELDB (STEP CLEANUP))))

(CLEANUP-PRECUE
IF
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM PRECUE IS ?X))
THEN
((DELDB (WM PRECUE IS ?X))))

(CLEANUP-TARGET-TEXT
IF
```

```
((GOAL DO MENU TASK)
 (STEP CLEANUP)
 (WM TARGET-TEXT IS ?X))
THEN
((DELDB (WM TARGET-TEXT IS ?X))))
```

BIBLIOGRAPHY

BIBLIOGRAPHY

Aaltonen, A., Hyrskykari, A., & Räihä, K.-J. (1998). 101 spots, or how do users read menus? *Proceedings of CHI 98,* New York: ACM, 132-139.

Abrams, R. A., Meyer, D. E., & Kornblum, S. (1989). Speed and accuracy of saccadic eye movements: Characteristics of impulse variability in the oculomotor system. *Journal of Experimental Psychology: Human Perception and Performance,* 15(3), 529-543.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Lebiere, C. (Eds.). (1998). *The Atomic Components of Thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction,* 12(4), 439-462.

Antes, J. R. (1974). The time course of picture viewing. *Journal of Experimental Psychology*, 103(1), 62-70.

Apple Computer. (1993). *Macintosh Human Interface Guidelines*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Arani, T., Karwan, M. H., & Drury, C. G. (1984). A variable-memory model of visual search. *Human Factors,* 26(6), 631-639.

Arend, U., Muthig, K. P., & Wandmacher, J. (1987). Evidence for global feature superiority in menu selection by icons. *Behaviour and Information Technology*, 6(4), 411-426.

Backs, R. W., Walrath, L. C., & Hancock, G. A. (1987). Comparison of Horizontal and Vertical Menu Formats. *Proceedings of the Human Factors Society 31st Annual Meeting,* Santa Monica, CA: Human Factors Society, 715-717.

Battig, W. F., & Montague, W. E. (1969). Category Norms for Verbal Items in 56 Categories. *Journal of Experimental Psychology Monographs,* 80(3).

Bednall, E. S. (1992). The effect of screen format on visual list search. *Ergonomics,* 35, 369-383.

Biederman, I. (1972). Perceiving real-world scenes. *Science,* 177, 77-79.

Biederman, I., Glass, A. L., & Stacey, E. W., Jr. (1973). Searching for objects in real-world scenes. *Journal of Experimental Psychology*, 97, 22-27.

Bloomfield, J. (1972). Visual search in complex fields: Size differences between target disc and surrounding discs. *Human Factors,* 14, 139-148.

Boff, K. R., & Lincoln, J. E. (1988). Visual Search, *Engineering Data Compendium: Human Perception and Performance.*: AAMRL, Wright-Patterson AFB, OH, 1549-1602.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text editing skill: A cognitive complexity analysis. *Human-Computer Interaction*, 5, 1-48.

Boynton, R. M., & Boss, D. E. (1971). The effect of background luminance and contrast upon visual field location. *Illuminating Engineering*, 66, 173-186.

Byrne, M. D. (1993). Using icons to find documents: Simplicity is critical. *Proceedings of INTERCHI '93,* New York: ACM, 446-453.

Byrne, M. D., Anderson, J. R., Douglass, S., & Matessa, M. (1999). Eye tracking the visual search of click-down menus. *Proceedings of CHI 99,* New York: ACM, 402-409.

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

Card, S. K. (1983). Visual search of computer command menus. In H. Bouma & D. G. Bouwhuis (Eds.), *Attention and Performance X: Control of Language Processes*. London: Lawrence Erlbaum Associates, 97-108.

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

Carswell, C. M. (1992). Choosing specifiers: An evaluation of the basic tasks model of graphical perception. *Human Factors*, 34(5), 535-554.

Carter, E. C., & Carter, R. C. (1981). Color and conspicuousness. *Journal of the Optical Society*, 71, 723-729.

Cave, K. R., & Wolfe, J. M. (1990). Modeling the role of parallel processing in visual search. *Cognitive Psychology*, 22, 225-271.

Chong, R. S. (1998a). Modeling dual task performance improvement: Casting executive process knowledge acquisition as strategy refinement. (Computer Science Technical Report CSE-TR-378-98). Ann Arbor, Michigan: The University of Michigan. Also, Ph.D. dissertation in Computer Science and Engineering, The University of Michigan.

Chong, R. S. (1998b). Modeling dual-task performance improvement with EPIC-Soar. *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum.

Clancy, T. (1984). *The Hunt for Red October*. Annapolis, Maryland: Naval Institute Press.

Cleveland, W. S., & McGill, R. (1985). Graphical perception and graphical methods for analyzing scientific data. *Science*, 229, 828-833.

Cohen, K. M. (1981). The development of strategies of visual search. In D. F. Fisher, R. A. Monty, & J. W. Senders (Eds.), *Eye movements, Cognition, and Visual Perception*. Hillsdale, NJ: Lawrence Erlbaum Associates, 271-288.

Crosby, M. E., & Peterson, W. (1991). Using eye movements to classify search strategies. *Proceedings of the Human Factors Society 35th Annual Meeting,* Santa Monica, CA: Human Factors Society, 1476 - 1480.

Dosher, B. A. (1998). Models of visual search: Looking for a face in the crowd. In D. Scarborough & S. Sternberg (Eds.), *Methods, Models, and Conceptual Issues: An Invitation to Cognitive Science*. Cambridge, Massachusetts: MIT Press, 455-521.

Drury, C. G., & Clement, M. R. (1978). The effect of area, density, and number of background characters on visual search. *Human Factors*, 20(5), 597-602.

Engel, F. L. (1977). Visual conspicuity, visual search, and fixation tendencies of the eye. *Vision Research*, 17, 95-108.

Findlay, J. M. (1992). Programming of stimulus-elicited saccadic eye movements. In K. Rayner (Ed.), *Eye Movements and Visual Cognition: Scene Perception and Reading*. New York: Springer-Verlag.

Fisher, D. F., Monty, R. A., & Senders, J. W. (Eds.). (1981). *Eye movements, Cognition, and Visual Perception*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Francik, E. P., & Kane, R. M. (1987). Optimizing visual search and cursor movement in pull-down menus. *Proceedings of the Human Factors Society 31st Annual Meeting,* Santa Monica, CA: Human Factors Society, 722-726.

Galitz, W. O. (1993). *User-Interface Screen Design*. Wellesley, MA: QED Information Sciences, Inc.

Giroux, L., & Bellau, R. (1986). What's on the menu? the influence of menu content on the selection process. *Behaviour and Information Technology*, 5, 169-172.

Gordon, I. (1968). Interactions between items in visual search. *Journal of Experimental Psychology*, 76, 348-355.

Han, S. H., Jorna, G. C., Miller, R. H., & Tan, K. C. (1990). A comparison of four input devices for the Macintosh interface. *Proceedings of the Human Factors Society 34th Annual Meeting,* Santa Monica, CA: Human Factors Society, 267-271.

Hollands, J. G., & Merikle, P. M. (1987). Menu organization and user expertise in information search tasks. *Human Factors*, 29(5), 577-586.

Hornof, A. J., & Kieras, D. E. (1997). Cognitive modeling reveals menu search is both random and systematic. *Proceedings of ACM CHI 97: Conference on Human Factors in Computing Systems,* New York: ACM, 107-114.

Hornof, A. J., & Kieras, D. E. (1999). Cognitive modeling demonstrates how people use anticipated location knowledge of menu items. *Proceedings of ACM CHI 99: Conference on Human Factors in Computing Systems*, New York: ACM, 410-417.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3, 320-351.

Jonides, J., & Gleitman, H. (1972). A conceptual category effect in visual search: O as letter or as digit. *Perception & Psychophysics*, 12, 457-460.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12(4), 391-438.

Kieras, D. E., Wood, S. D., Abotel, K., & Hornof, A. (1995). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. *Proceedings of the ACM Symposium on User Interface Software and Technology, UIST '95,* New York: ACM, 91-100.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Transactions on Computer-Human Interaction*, 4(3), 230-275.

Klatzky, R. (1980). *Human Memory*. (2nd ed.). San Fransisco: W. H. Freeman.

Koopman, B. O. (1946). Search and Screening. (OEG Report No. 56). Washington, D. C.: Operations Evaluation Group, Office of the Chief of Naval Operations, Navy Department.

Krendel, E. S., & Wodinsky, J. (1960). Search in an Unstructured Visual Field. *Journal of the Optical Society of America*, 50(6), 562-568.

Laird, J., Rosenbloom, P., & Newell, A. (1986). *Universal subgoaling and chunking.* Boston: Kluwer Academic Publishers.

Landauer, T. K., & Nachbar, D. W. (1985). Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth and width. *Proceedings of CHI '85,* New York: ACM.

Lee, E., & MacGregor, J. (1985). Minimizing user search time in menu retrieval systems. *Human Factors*, 27(2), 157-162.

Loftus, G. R. (1976). A framework for a theory of picture recognition. In R. A. Monty & J. W. Senders (Eds.), *Eye Movements and Psychological Processes*. Hillsdale, NJ: Lawrence Erlbaum.

Lohse, G. L. (1993). A cognitive model for understanding graphical perception. *Human-Computer Interaction*, 8, 353-388.

MacGregor, J., & Lee, E. (1987). Menu search: Random or systematic? *International Journal of Man-Machine Studies*, 26(5), 627-631.

MacGregor, J., Lee, E., & Lam, N. (1986). Optimizing the structure of database menu indexes: A decision model of menu search. *Human Factors*, 28(4), 387-399.

MacKenzie, I. S., & Buxton, W. (1992). Extending Fitts' law to two-dimensional tasks. *Proceedings of CHI '92*, New York: ACM, 219-226.

Mayes, J. T., Draper, S. W., McGregor, A. M., & Oatley, K. (1988). Information Flow in a User Interface: The Effect of Experience and Context on the Recall of MacWrite Screens. In D. M. Jones & R. Winder (Eds.), *People and Computers IV*. Cambridge, UK: Cambridge University Press, 275-289.

Mayhew, D. J. (1992). *Principles and Guidelines in Software User Interface Design*. Englewood Cliffs, New Jersey: Prentice Hall.

Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., & Smith, J. E. K. (1988). Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review*, 95(3), 340-370.

Meyer, D. E., & Kieras, D. E. (1997a). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, 104(1), 3-65.

Meyer, D. E., & Kieras, D. E. (1997b). A computational theory of executive cognitive processes and multiple-task performance: Part 2. Accounts of psychological refractory-period phenomena. *Psychological Review*, 104(4), 749-791.

Meyer, D. E., & Kieras, D. E. (1999). Précis to a practical unified theory of cognition and action: Some lessons from EPIC computational models of human multiple-task performance. In D. Gopher & A. Koriat (Eds.), *Attention and Performance XVII. Cognitive Regulation of Performance: Interaction of Theory and Application*. Cambridge, MA: MIT Press, 17-88.

Meyer, D. E., Smith, J. E. K., Kornblum, S., Abrams, R. A., & Wright, C. (1990). Speed-accuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action. In M. Jeannerod (Ed.), *Attention and Performance XIII*. Hillsdale, N. J.: Lawrence Erlbaum, 173-226.

Monty, R. A., & Senders, J. W. (1976). *Eye Movements and Psychological Processes*. Hillsdale, NJ: Lawrence Erlbaum.

Müller-Brockmann, J. (1981). *Grid Systems in Graphic Design*. New York: Hastings House Publishers, Inc.

Neisser, U. (1963). Decision-time without reaction-time: experiments in visual scanning. *American Journal of Psychology*, 76, 376-385.

Neisser, U. (1976). *Cognition and Reality*. San Fransisco: W. H. Freeman & Co.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, Massachusetts: Harvard University Press.

Nilsen, E., & Evans, J. (1999). Exploring the divide between two unified theories of cognition: Modeling visual attention in menu selection. *CHI 99 Extended Abstracts,* New York: ACM, 288-289.

Nilsen, E. L. (1991). Perceptual-motor control in human-computer interaction. (Tech. Rep. No. 37). Ann Arbor, Michigan: The Cognitive Science and Machine Intelligence Laboratory, The University of Michigan. Also: Ph.D. dissertation in Psychology, The University of Michigan, 1991.

Norman, K. L. (1991). *The Psychology of Menu Selection: Designing Cognitive Control of the Human/Computer Interface*. Norwood, N. J.: Ablex.

Paap, K. R., & Roske-Hofstrand, R. J. (1988). Design of menus, *Handbook of Human Computer Interaction*. Amsterdam: Elsevier Science Publishers, 205-235.

Parton, D., Huffman, K., Pridgem, P., Norman, K., & Shneiderman, B. (1985). Learning a menu selection tree: training methods compared. *Behaviour and Information Technology,* 4, 81-91.

Perlman, G. (1984). Making the right choices with menus. *Proceedings of Interact '84,* Elsevier Science Publishers, 317-321.

Prinz, W., Nattkemper, D., & Ullmann, T. (1992). Moment-to-moment control of saccadic eye movements: Evidence from continuous search. In K. Rayner (Ed.), *Eye Movements and Visual Cognition: Scene Perception and Reading*. New York: Springer-Verlag.

Raman, T. V. (1997). *Auditory User Interfaces: Toward the Speaking Computer*. Boston: Kluwer Academic Publishers.

Rayner, K. (1992). *Eye Movements and Visual Cognition: Scene Perception and Reading*. New York: Springer-Verlag.

Resnick, P., & Virzi, R. A. (1993). Skip and scan: Cleaning up telephone interfaces. *Proceedings of CHI '93,* New York: ACM Press.

Rosenbaum, D. A. (1991). *Human motor control*. New York: Academic Press.

Russo, J. E. (1978). Adaptation of cognitive processes to the eye movement system. In J. W. Senders, D. F. Fisher, & R. A. Monty (Eds.), *Eye Movements and the Higher Psychological Functions*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 89-109.

Scott, D. (1993). Visual search in modern human-computer interfaces. *Behaviour and Information Technology*, 12(3), 174-189.

Sears, A. (1993). Layout appropriateness: A metric for evaluating user interface widget layout. *IEEE Transactions on Software Engineering*, 19(7), 707-719.

Sears, A., & Shneiderman, B. (1994). Split Menus: Effectively Using Selection Frequency to Organize Menus. *ACM Transactions on Computer-Human Interaction,* 1(1), 27-51.

Senders, J. W., Fisher, D. F., & Monty, R. A. (Eds.). (1978). *Eye Movements and the Higher Psychological Functions*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Shneiderman, B. (1992). *Designing the User Interface: Strategies for effective human-computer interaction.* (Second ed.). Reading, Massachusetts: Addison-Wesley.

Smith, M. R. (Ed.). (1985). *Military Enterprise and Technological Change*. Cambridge, Massachusetts: MIT Press.

Smith, S. L., & Mosier, J. N. (1986). Guidelines for designing user interface software. (Technical Report ESD-TR-86-278). Hanscom Air Force Base, MA: USAF Electronic Systems Division.

Somberg, B. L. (1987). A comparison of rule-based and positionally constant arrangements of computer menu items. *Proceedings of CHI '87,* New York: ACM, 79-84.

Somberg, B. L., Boggs, G. J., & Picardi, M. C. (1982). Search and decision processes in human interaction with menu-driven systems. Presented at the Human Factors Society 26th Annual Meeting. Paper did not appear in the conference proceedings, but was supplied by the author.

Somberg, B. L., & Picardi, M. C. (1983). Locus of the information familiarity effect in the search of computer menus. *Proceedings of the Human Factors Society 27th Annual Meeting,* Santa Monica, CA: Human Factors Society, 826-830.

Sontag, S., & Drew, C. (1998). *Blind Man's Bluff: The Untold Story of American Submarine Espionage*. New York: Public Affairs.

Staggers, N. (1993). Impact of screen density on clinical nurses' computer task performance and subjective screen satisfaction. *International Journal of Man-Machine Studies*, 39, 775-792.

Treisman, A. (1986). Features and objects in visual processing. *Scientific American,* 255, 114B-125.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire, Connecticut: Graphics Press.

Tullis, T. S. (1988). A System for Evaluating Screen Formats: Research and Application. In R. Hartson & D. Hix (Eds.), *Advances in Human-Computer Interaction*. (Vol. 2). Norwood, NJ: Ablex, 214-286.

Vandierendonck, A., Van Hoe, R., & De Soete, G. (1988). Menu search as a function of menu organization, categorization and experience. *Acta Psychologica,* 69(3), 231-248.

Vartabedian, A. G. (1971). The effects of letter size, case, and generation method on CRT display search time. *Human Factors*, 13, 363-368.

Walker, N., Meyer, D. E., & Smelcer, J. B. (1993). Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction. *Human Factors*, 35(3), 431-458.

Wiesmeyer, M. D. (1992). *An Operator-Based Model of Human Covert Visual Attention*. Ph.D. dissertation in Computer Science and Engineering, The University of Michigan, Ann Arbor, Michigan.

Williams, J. R. (1988). The effects of case and spacing on menu option search time. *Proceedings of the Human Factors Society 32nd Annual Meeting*, Santa Monica, CA: Human Factors Society, 341-43.

Williams, L. G. (1966a). The effect of target specification on objects fixated during visual search. *Perception & Psychophysics*, 1, 315-318.

Williams, L. G. (1966b). Target conspicuity and visual search. *Human Factors*, 8(1), 80-92.

Wolf, C. E. (1986). BNA "HN" command display: Results of user evaluation. Unpublished technical report. Tullis (1988) cites as available from Cynthia Wolf Connally, Unisys Corporation, 19 Morgan, Irvine, CA 92718. But Cynthia Wolf is evidently no longer with Unisys, and Unisys no longer at this location.

Wood, S. D. (1993). Issues in the implementation of a GOMS-model design tool. Unpublished report, University of Michigan.

Woodward, R. M. (1972). Proximity and direction of arrangement in numeric displays. *Human Factors*, 14(4), 337-343.