

# Menu-Driven Systems

Eric S. Lee

Finance and Management Science  
St. Mary's University  
Halifax, Nova Scotia, Canada  
B3H 3C3

Darrell R. Raymond

Department of Computer Science  
University of Waterloo  
Waterloo, Ontario, Canada  
N2L 3G1

## I. What is a menu?

Menus are ubiquitous in modern microcomputer systems. Though the word "menu" usually conjures up a mental image of a list of words or phrases, the scope of menu-driven systems is much broader: every program that displays user-selectable data is, in part, a menu-driven system. Electronic paint programs, for example, typically display a menu of icons from which various drawing tools or functions can be selected. Desktop systems present a large, two-dimensional menu of icons that represent files, directories, devices, and other system resources. Full-screen word processors and electronic spreadsheets display a "menu" of text or numeric data, which the user can edit directly. Even the lowly telephone is sometimes menu-driven; telephone-accessible public information services present aural menus and the user indicates a selection by dialing the number associated with the desired option.

Despite this variety of forms and styles, all menu-driven systems share two fundamental characteristics:

Menus exploit recognition rather than recall. Menu-driven systems present part or all of their state for the user to investigate and manipulate. Non-menu systems hide their state; users are forced to remember both the state

and the set of operations that are possible at any step in the process.

Menus decouple the functions of input and display. Menu-driven systems display commands or data in substantial detail, but require only a minimal effort to select those commands or data. Non-menu systems couple input and display; there is a tradeoff between short, easily-input commands and longer, more informative ones.

The first characteristic of menu-driven systems reduces mental effort. By explicitly presenting the state of the program and the set of available options, menu-driven systems reduce the user's memory load, promote exploration, and avoid certain types of syntax error. The second characteristic of menu-driven systems reduces physical effort. Since input and display are separated, verbose, informative displays can be presented without a corresponding increase in input.

Menu-driven systems exploit the high information bandwidth of human perceptual systems while making maximal use of the low information bandwidth of human motor systems. The advantages of menu-driven systems spring directly from their close approximation to the peculiar characteristics of the human operator of the computer,

---

This article will appear in the Encyclopedia of Microcomputers, Allen Kent and James G. Williams (eds.), Marcel Dekker, New York.

rather than from any inherent computational advantage. It is not unusual, then, that menu-driven systems result from the designer's concern for a better user interface. Nor is it unexpected that the evolution of menu-driven systems is so intimately connected with the evolution of the personal computer.

## II. Varieties of menu.

There are a large number of ways to construct menus, depending on system capabilities and application. We will briefly describe a number of techniques.

The simplest type of menu is a *text* menu. Text menus are short lists of words or phrases, usually vertically arranged and sometimes taking up the whole space of the display. In the latter case the menu is often called a *frame*. Options on a text menu are often accompanied by a number or letter that serves as a selector for the option.(1) If more advanced input devices such as a mouse, light pen, or touch screen, are available, then the label itself can serve as a selector.(2) Figure 1 shows a simple text menu.

Menus can be either *static* or *dynamic*. A static menu occupies a fixed position and area of the screen for the duration of some activity. A dynamic menu is invisible except when a selection is required; the user calls up the menu, makes a selection, and submerges the menu. Static menus are common in editors, which sometimes present the set of editing operations as a row or column of buttons displayed during the whole editing process. Iconic menus in paint programs and desktop operating systems are also static.

Many systems use a combination of static and dynamic menus, and often static menus are used to obtain dynamic menus. A prevalent example is the *menu bar* and *pull-down* menu, as shown in Figure 2. The menu bar is a static menu; it presents the titles of several dynamic menus that are available. As the cursor is moved over the menu bar, the individual menus are dynamically "pulled down" for inspection by the user. Menu bars are located at the top of the screen and so the menus always appear in the same place; thus, they are dynamically accessed but anchored to a single position. Since there is typically only one screen, menu bars and pull-down menus are best suited to single-tasking environments.

Multitasking environments with extensive processing power and screen area tend to use some form of *pop-up* menu. Pop-up menus appear at

the current location of the cursor in response to a mouse button or function key command. Pop-up menus are more dynamic than pull-down menus because their position is relocatable; as a result, they are better for multitasking environments. Pop-up menus do not require the user to move the cursor to the top of the screen to access the menu bar, an important consideration for large displays. Pull-down menus have the advantage, however, that the top of the screen is a useful border.(3)

Often the number of options to be shown is too large to fit on a single pop-up or pull-down menu. There are two common solutions to this problem. The first solution is to treat the menu as a text window and permit the user to scroll or page through the list of options. The second solution is to use a number of menus and divide the options either sequentially or hierarchically. One sequential technique is the multiple-pane or *deck-of-cards* menu, as shown in Figure 3. In this technique multiple menu panes are popped up with a mouse button, arranged as if they were a set of overlapping cards. By moving the cursor over the panes, the user can make each of the panes pop to the front of the stack. This type of menu is also sometimes called a *slip-off* menu. Deck-of-cards menus are found in the Andrew window system.(4)

The most common hierarchical technique for handling multiple menus is the *pull-right* menu, sometimes known as a *walking* or *cascading* menu, as shown in Figure 4. In this technique the highlighting of a menu option causes the appearance of a subsidiary menu that shows a related group of sub-options; moving to the sub-menu may produce further sub-menus. Pull-right menus can be thought of as an extension of the menu bar technique; just as with the menu bar, each higher level menu is a collection of menu titles; as the user moves the cursor over the title menu, the subsidiary menus show up to the right hand side of the title menu. Unlike deck-of-cards and other sequential approaches, the pull-right menu can easily be extended to several levels of hierarchy. More than one or two levels of pull-right menu may be difficult to manage, however, since the user tends to fall off the menu, losing his or her position in the selection hierarchy. Pull-right menus are found in the NeWS window system.

The virtues of static and dynamic menus are combined in *tear-off*, *pin-up*, *holddown*, and *staydown* menus.(5) These menus can be used like pop-up

menus, but can be made static if the user so chooses. The phrases “tear-off” and “pin-up” come from the metaphor used to explain the menu’s capability; a tear-off menu is like a page torn out of a notepad, while a pin-up menu is like a page pinned to a notepad. Tear-off menus are found in the NextStep system; pin-up menus are found in Open Look.

Another combination of static and dynamic menus is the *cursor* menu.(6) A cursor menu is a small menu that is displayed as a cursor; it is highly dynamic since it moves with the pointing device, but it is static in the sense that it is always visible. Since cursor menus typically describe the set of operations available by using the mouse buttons, they are most appropriate when the operations change according to the current state of a task. Textual cursor menus are used in the KMS system;(7) graphical cursor menus are found in *lector*.(8)

An interesting menu technique is the *circular* or *pie* menu, shown in Figure 5. Pie menus take advantage of two dimensions by arranging menu options as in a pie chart, rather than the usual vertical list. The physiological rationale for pie menus is that the average effort of selecting a menu option is low, since all options are at the same distance from the starting point of the cursor, the centre of the menu. The psychological rationale for pie menus is that two-dimensional position is an added cue to menu structure. Some empirical studies show that pie menus can be efficient, but there are practical problems.(9,10) First, hierarchical structures of pie menus are not as visually effective as for vertical menus, since the multiple pies tend to overlap one another. Second, a pie menu takes more room to display than a corresponding list menu. Laying out a pie chart is NP-hard, so human intervention may be required to construct good, small pie menus.(11)

The internal structure of the modern menu can be quite complex. Menu options are usually organized either alphabetically or functionally; in the latter case the menu may use white space or lines to separate the various functional groupings. Frequency can also be used to organize the menu; either the options can be reordered, or highlighting can be used to indicate frequent options.(12,13) Some menus present a list of options which are not mutually exclusive; in this case it is important to indicate which of the options have already been selected, sometimes by prefacing the selected options with check marks

or other indicative graphics. Alternatively, selection of options or the current state of the program may invalidate some options on the menu; rather than being removed, options are often *dimmed*, that is, printed in a lighter font to show that they are currently unavailable. By dimming the option the system preserves the integrity of the menu and lets the option serve an informative function, if not an operational one. Deletion, on the other hand, reduces the size of the menu and can lead to better performance.(14)

In the case of hierarchical menus, especially pull-right menus, it is often useful to indicate which of the options have sub-menus; this can be done by decorating the label with an ellipsis or an arrow to show that further structure can be obtained. In some menu systems the ellipsis is reserved for dialog boxes associated with a given option. It is also common to indicate a function key shortcut for accessing the function or data associated with the menu option; this is usually done by displaying symbols corresponding to the shortcut keys on the right side of the menu. Some systems provide a default option, which permits selection of a frequently-accessed function without displaying (and navigating) the entire menu.

In addition to these features, menus are often decorated with borders, drop shadows, beveled edges, and other effects intended to improve their graphical appearance.

### III. A menu’s-eye view of the history of microcomputers.

The history of microcomputers is one of attempts to bring increasing amounts of computer power to greater numbers of people at continually lower prices. Menu-driven systems play an important role in this history. Personal computers have facilitated the development of increasingly complex menu-driven systems in two ways. First, as personal computers became pervasive, more people had the computational horsepower necessary to support the extra demands of menu-driven systems. Second, the broadening of the user base meant that software designers were forced to develop simpler, more easily understandable systems, which were typically menu-driven. In turn, the design of menu-driven systems has altered the evolution of the personal computer. An increasing share of computing resources has been directed at enhancing the user’s dialogue with microcomputers, including large display memories, bitmapped screens, a variety of pointing devices, and

extensive system software supporting complex displays.

We can identify three broad phases of development in menu-driven systems. In the first phase, menus were employed as an interface for novices, while expert users still preferred command-line interfaces. This phase is characterized by slow, fixed menu dialogues using primitive interface hardware. In the second phase, developers built systems in which menus were valuable and effective for most users, novice and expert alike. This phase is characterized by a wide variety of menus, and a focus on rapid and flexible menu interfaces supported by powerful interface hardware. In the third and current phase, the pervasiveness of microcomputers is changing the nature of users' tasks; the emphasis will soon be on the production of dynamic, online artifacts rather than static paper documents. In this phase, active, dynamic menus have shifted from being an interface technique to take up a role as a fundamental metaphor for electronic artifacts.

#### **Menus for novices.**

The earliest menu-driven systems employed text menus. Text menus require minimal hardware and software support, usually only a keyboard for input and a simple character-based CRT for output. Text menus can even be used with such primitive devices as keypads and hardcopy terminals. The software support needed for such menus is minimal, as the programmer merely uses the input/output statements of a programming language to present the menu and read the input. The use of the menu is simple, since the dialogue is completely synchronous.

Early text menus were often quite slow because of the slow speed of communications links and display devices. Programmers typically constructed restricted dialogues, in which users answered a fixed set of questions in fixed order. The stilted nature of these interfaces meant that they were usually considered suitable only for novices. For this reason, text menus have been abandoned as a general interface technique in modern microcomputers. Text menus are still found in low-level applications, such as ROM-based setup programs for workstations and other places where the output device is limited.

A number of researchers attempted to build text menu systems which did not suffer from overly stilted dialogues and slow hardware. The earliest notable example was PROMIS, the Problem

Oriented Medical Information System.(15) PROMIS's developers decided that if input and output were made more effective, a large and powerful database could be built as a menu-driven system, retaining the simplicity of menus while still being fast enough for expert users. Effective output was provided by expensive and fast (for their time) character display devices; effective input was provided by using a touch screen to directly select options on the text menu. PROMIS introduced the use of large hierarchies of menu frames, reaching databases as large as 30,000 frames.

Carnegie-Mellon followed up the PROMIS effort with a more general system known as ZOG.(16-18) ZOG stored not only an application's information but also a variety of other kinds of data, including electronic mail, technical papers, documentation, and eventually even its own source code. Each application was converted to an extensive hierarchy of text frames, and the tasks of searching, navigating, displaying, and editing were all reduced to a single task — that of operating a text menu. Research on ZOG brought to light the two perennial problems of text menu hierarchies. The first, the so-called "depth/breadth" problem, is the problem of finding the right shape for the hierarchy.(19) The second, the "disorientation problem", is the problem of getting lost while browsing.(20)

Public access to ZOG-like databases was the goal of *videotex* and *viewdata*.(21) Conceived in the years just before personal computers became an important phenomenon, videotex was an attempt to provide wide access to computer databases by means of the television set and telephone. Videotex systems were designed for general public use, and since the vast majority of potential users had little or no experience with computers, a restricted text menu dialogue structure seemed the best choice. The predictions of broad usage for videotex led many governments and other large institutions to conduct extensive evaluations of videotex systems, but public information systems never did fulfill their advertised potential, due to the competition from the more attractive and flexible microcomputer.(22) Nevertheless, the basic research trends for menu-driven systems were set by this type of application. Enhancing PROMIS-like information hierarchies is still the basic goal of a great deal of current research work.

The first personal computer, the Altair, was announced in January 1975 and the Apple

followed shortly after.(23) The early machines were designed for hobbyists, and so did not have particularly transparent interfaces. It was not until the Apple II that a personal computer was delivered in ready-to-use form; it was quickly followed by Radio Shack's TRS-80 and Commodore's PET. Menu-driven systems were important for these early computers for several reasons. Much of the software was written by hobbyists and hackers who often did not provide documentation; thus the programs that got used were the ones which were partially or wholly menu-driven. Computer games were the most popular activities, and games tended to incorporate menu-driven features to control their options (although Adventure, an early favorite, was command-line oriented). But the most important menu-driven interface was that provided by VisiCalc, the first electronic spreadsheet. This single application provided the main impetus for business computing, and hence for the eventual production of the IBM PC.

#### **Menus for experts.**

The second phase in menu-driven systems was based on the desire to support complex information management tasks. Unlike text menu systems, which were explicitly designed for novices, the second phase resulted in menu-driven systems for sophisticated users. The earliest and most influential system was Engelbart's NLS, later called Augment.(24,25) NLS incorporated so many advances that it took twenty years before they were widely available in personal microcomputers: a partial list includes networked workstations, full screen editing, mixed text and graphics, windowing, electronic mail and database facilities. NLS's most important contribution to the second phase was its philosophy of augmenting knowledge workers, but the most easily appreciated contribution was the mouse, now widely used for manipulating menus. The mouse removed the need for presenting a selection identifier with each menu option, as the position of the option label became its identifier.

NLS was extremely advanced, but it required hardware and software capabilities that simply did not exist elsewhere. The philosophy of augmenting the sophisticated user also needed further development. Members of Engelbart's team went to Xerox's Palo Alto Research Centre and followed up NLS with Alto. Alto combined existing ideas — the mouse and full screen editing techniques of NLS, and the notion of a personal

computer — with bitmapped displays, the result being the first workstation. Alto exhibited several types of menu, including pop-up menus, soft buttons, and icons. Alto led to Star,(26) which provided menu-driven systems for the office environment. One of Star's key concepts was that everything in the system should be visible, and everything visible should be selectable — a menu-driven system in the purest sense. Xerox's sustained work on Alto and Star finally attracted the attention of microcomputer companies. Bit-mapped displays, the mouse, and menu-driven systems for public use appeared in the Apple's Lisa and Macintosh, Commodore's Amiga, and Atari's ST. DOS machines lagged behind for several years, partly because of a lack of hardware, but more significantly because of a lack of system software to facilitate consistency across applications. Eventually, Microsoft's Windows 3.0 provided a framework (and perhaps more importantly, an industrial mandate) in which consistent menu-driven applications could be developed.

Meanwhile, attempts were being made to merge the second phase of menu-driven systems with Unix, a powerful and flexible operating system somewhat notorious for its cryptic command languages. Sun Microcomputers produced SunWindows, the first widely used window system for Unix.(27) SunWindows had pop-up menus and icons, but generally operated as a hybrid; the menu-driven approach was used largely as a pleasant interface to the wide collection of command-based tools in the Unix environment. The Unix workstation — essentially a powerful personal microcomputer with a large memory and high-resolution display — became an instant hit, and fierce price and performance competition began between Sun, Digital Equipment, IBM, Apollo, and many other manufacturers. It rapidly became clear that the manufacturers would need to settle on a single approach to window systems, since proprietary approaches were simply not acceptable to most users of workstations. The rush to be the provider for such a system led to substantial work in developing new systems for supporting menu-based applications, including Andrew, X.10, X.11, New Wave, NeWS, Open Look, Motif, and NextStep. The most important contribution of the Unix workstation community to menu-driven systems was to fill the need for widely available software libraries that simplified the development of powerful menu-driven systems.

### **Integrating menus with artifacts.**

The third and current phase of menu-driven systems is based on the migration of tasks from paper-based to electronic environments. With the personal microcomputer now as common as the electronic calculator, users' activities have changed. Whereas previously the computer was employed to facilitate the production of static paper artifacts, microcomputers are now increasingly employed to produce artifacts which exist largely in electronic form. As a result, system design is turning to support dynamic electronic artifacts. The existence of selectable items in documents and selectable items in iconic interfaces suggests that every instance of an iconic menu interface can be considered a document(28) and every instance of a document can be considered an iconic menu interface.(8) The third phase of menu-driven systems is characterized by the increasingly nebulous distinction between interface, application, and data.

Three types of system are being investigated in the third phase of menu-driven systems. The first is *hypertext*,(29) the notion that a document should contain menu buttons which can be used to jump to related topics in non-linear fashion. This class of systems is directly descended from NLS's structured document editor. The second is *active document* systems, the merging of spreadsheets, databases, and report-generation programs into a single monolithic tool. Active documents contain pointers to dynamically changing data elements, and thus are updated instantly to reflect changes in the status of some monitored activity. The third type of system is *alternate realities*, where an entire world of objects and processes is simulated in a directly manipulable way.(30, 31)

The intertwining of menus with the data they access is largely dependent on extensive and powerful systems software that supports a variety of databases, powerful user interfaces, and application inter-operability. The most common architecture for such systems is object-oriented. Object-oriented systems rely on the identification of objects, the packaging of functionality with a data representation, and the use of hierarchical inheritance to promote code reuse. These features are strikingly similar to the characteristics of menu-driven systems, which also typically identify classes of activity or data and use a hierarchy to organize information. Thus it should not be surprising that object-oriented systems software is extensively employed in the development of

menu-driven interfaces.

### **IV. Research in menu-driven systems.**

The proliferation of menu-driven systems has engaged the attention of human factors researchers. Menu-driven systems are an attractive area for study partly because of their simplicity (thus making it relatively easy to set up experimental situations), partly because of their ubiquity (thus results are potentially of wide general interest), and partly because of a lack of existing guidelines for design; most systems have been built largely on the basis of their designers' intuitions. As a result, the goal of most research on menu-driven systems is the production of design guidelines of general applicability. The body of research on menu-driven systems is probably the most extensive of any topic in human-computer interaction.(19)

#### **IV A. Ease of learning and use.**

Menu-driven systems grew out of a desire for simple, easily learned software. Ease of learning and use involve both *competence* and *effectiveness*. Competence focuses on the types of tasks that can be performed with a system (its expressiveness) and the training necessary to reach a basic level of capability. *Effectiveness* focuses on the efficiency with which tasks are performed by users who have already reached a level of competence; it is often dependent on practice and experience.

It is somewhat curious that so little research has been done on competence analysis of menu-driven systems. One of the most basic characteristics of a menu-driven system is its self-explanatory or self-tutorial nature: the user learns how to use the system simply by following the displayed instructions and choosing from the displayed options. A basic level of competence can be obtained with virtually no training; in one of the few studies on training methods, the result was that no training method was superior to simple exploration of the menu hierarchy.(32) Contrast this result with the lack of success users have in random experimentation with command-driven systems.(33) Improved competence derives naturally from the user's browsing activity, but there appear to be few empirical studies of how this browsing activity proceeds, though many researchers think that it is of significant value in database querying. Furthermore, there are few studies of the expressiveness of menu-driven

systems — what sorts of queries are they capable of answering.(34, 35) Yet it is clear that menu-driven systems are not useful for all queries; for instance, menus are typically poor at finding multiple pages of information that are widely scattered in the database, because each page must be found individually.(36) Boolean query systems, by comparison, can sometimes address such queries.

Research has focused largely on efficiency analysis for several reasons. One is that it is easier to devise testable measures of performance. MacGregor and Lee(34) identified four common performance measures in videotex studies: success rate (what percentage of tasks were completed), probability of error, efficiency (how much above minimum effort was expended), and search time. Another reason is that researchers are often interested in comparisons between menus and alternative approaches, and so tend to study tasks that can be performed by both systems, thus eliminating competence analysis and focusing on effectiveness.

Existing work on effectiveness can be broadly separated into studies of menu-driven databases and menu-driven command systems.(34) Command systems are typically small, use single-word options, and involve searches for specific targets e.g., “print”. Database systems are typically large, use multiple word or phrase options, and involve searches for general or scenario targets e.g., “find an interesting book”. Fourteen studies of menu-driven database systems showed success rates ranging from 57% to 88%, probability of error from 18% to 37%, efficiency rates of 33% to 88%, and search time per page of 9 to 19 seconds. Ten studies of command-like systems showed success rates of 95% and search time per page of 2 to 9 seconds. These results show that for database systems at least, the effectiveness of menu-driven systems can be less than we would like. It is important to note, however, that other database retrieval techniques also suffer poor performance compared with theoretical maximums.(37)

Can we conclude that command menus are much more effective than database menus? The legitimacy of comparing effectiveness between the two types of system is questionable. A single-page menu or small menu hierarchy is relatively easily committed to long-term memory, so extensive tests of such menu systems usually become tests of the effectiveness of the user’s memory, rather than the menu system itself. Furthermore,

the two types of system differ in their sensitivity to mechanical effort. In a command environment the user wants highly efficient mechanical operation; even the physical effort of moving the hands from keyboard to mouse can drastically affect performance.(38) Further discomfort can come from the eye-hand coordination necessary to manage a pop-up or cascading menu.(3) The effectiveness of command menus is very sensitive to small mechanical inefficiencies, and relatively insensitive to gross semantic structuring errors; with sufficient practice subjects have similar effectiveness with random, alphabetical, or functional organizations.(39) In menu-driven databases, on the other hand, the physical effort of using the system is relatively minor compared with the cognitive effort of forming and executing a query. Since long search times per page are common, improving the mechanical aspects of menu querying will not significantly reduce the overall search time, though they may make the system more pleasant.

The most common approach taken to make menu-driven databases easy to learn and use is the adoption of a navigational metaphor. The user is implicitly or explicitly led to think that the use of the menu system is a process of “moving” about the database. This metaphor is intended to improve ease of learning and use by referring the user to a well-known task; however, it has the disadvantage of introducing problems from that well-known task, and of overly restricting search possibilities. The most outstanding problem carried by the navigational analogy is that of disorientation, of becoming lost in the system. Though most people are familiar with navigation, few are good at pathfinding.(36) The disorientation problem is sometimes addressed (following the navigational metaphor) with a map. Apart from the problem that this solves disorientation on one menu by turning to another, it raises the question of people’s effectiveness at mapreading.(40-42)

Another problem arising directly from navigation is the tedium of using the system to find well-known pages, which involves searching from the top of the tree through a well-traversed path. This problem is most easily addressed by simply dropping the navigational metaphor and giving users direct access to well known pages, by means of keywords(43-45) or default paths.(46) A third problem is the problem of cascading errors; any error made in the first few steps of the path is worse than errors made later on.(47) This problem

is directly related to the sequential nature of the navigational approach to querying, and can be overcome by treating the menus as providing a set (rather than a sequence) of questions. A fourth problem is the difficulty of performing searches for multiple targets. The navigational metaphor imposes a single selection paradigm on menu-driven systems; if we remove this restriction and permit multiple selections, then parallel searches are possible.(36)

The navigational metaphor is so deeply embedded in the use of menu-driven databases that the two are often treated as if they were inseparable. Separating the metaphor from the system by means of direct access and default paths can result in improved competence and effectiveness.

#### **IV B. Menu organization.**

Research on menu-driven systems has focused on methods of organizing menus to improve efficiency. Alphabetical and frequency-based techniques have received some attention, but the most thoroughly studied organization is categorization. Alphabetic and frequency-based techniques are generally restricted to within-menu organization, while categorization is often applied across menus.

##### **User search strategy.**

A user's search strategy determines the effectiveness of menu organization. If searches are random, for instance, then no organizational technique is generally advantageous. Researchers have studied several types of search strategy. In redundant search each option is scanned more than once. In exhaustive search each option is scanned exactly once. In self-terminating search options are scanned only until the target option is found. In directed search, knowledge of the organization is used to restrict search to a small part of the menu, so only a few options are scanned.(48-54)

Card(39) postulated random search, based on the assumption that the user cannot remember previous menu locations and so searches randomly (i.e., sampling with replacement). Empirical evidence including eye-movement results was presented to support his claim. MacGregor and Lee(55) re-examined Card's empirical results and showed that they were also in accord with what one would expect from at least some kinds of systematic, sequential search models. MacGregor and Lee proposed that search was probably

systematic with a random component. Several subsequent eye-movement studies provide strong support for this claim.(53, 56) Search seems to be knowledge driven, knowledge limiting search to a constrained region of the menu within which search is essentially random or non-systematic. The fact that the searched area becomes smaller with organization and practice argues for the role of knowledge in directing and limiting the search, while eye-movement analyses indicate that random search is confined to a restricted portion of the menu.

##### **Alphabetical organization.**

If search is knowledge-driven rather than random, then it is reasonable to consider some form of menu organization. One of the easiest types of organization to provide for textual menus is alphabetical. Alphabetical organization is usually confined to ordering options within a single menu, rather than across several menu frames. Even on a single menu, however, we can consider complete alphabetization of all options, or alphabetization of sub-groups, such as rows and columns. The effectiveness of alphabetical ordering seems to depend on the type of search task.

For explicit search tasks (i.e., an identity matching task), complete alphabetization is very effective in improving performance, in each study at least as effective as categorization.(39, 57-60) Hollands and Merikle found an alphabetical ordering to be superior for novices, but the advantage eroded rapidly with practice.

For general search tasks, complete alphabetical ordering is marginally better than random ordering, but quite inferior to any form of categorization.(60) Alphabetically ordering options within categories is no better than a random ordering and is much worse than categorization within categories.(58, 59) As with categorizing, the beneficial effects of alphabetical ordering often diminish with practice.(39, 57, 59) Alphabetization by column is marginally better than by row.(57)

Greenberg and Witten studied alphabetically ordered menu hierarchies using various combinations of delimiters to indicate the range of each option.(61) Delimiters that specified only the upper end of the range appeared to provide the fastest scanning and lowest error rate. Another conclusion was that efficiency deteriorated with depth. Landauer and Nachbar also arrived at this conclusion.(62)

### **Frequency organization.**

The frequency with which options are selected can also be used for organization within a menu. For example, options can be listed in order of the probability that they will be selected by users. A user search strategy that focused on the first few options would be expected to benefit from such an ordering if the probability of selection is not constant across options. The greater the variation in probability of selection, the more likely that this form of organization might be effective.

Greenburg and Witten report a telephone database organized according to frequency which resulted in 60% fewer errors and 35% less scanning time.(63) McDonald *et al.* tested a menu system in which users entered multiple targets on a single menu search.(64) They tested 2 fast food menu layouts; in one, the most frequently co-occurring food options were placed most closely together on the menu, while in the other, the most similar options were placed most closely together on the menu. The frequency arrangement was best for realistic food orders, and the advantage persisted with practice. On the other hand, Somberg reports that with practice, searches on positionally constant menu options were faster than alphabetic, random, or frequency-based orderings.(65)

### **Categorization.**

The most problematic (yet potentially effective) type of organization is categorization. Categorization involves the clustering of menu options into "meaningful" groups and the titling or labeling of these groups. Performance for categorized menu systems usually depends critically on the way in which options are grouped and how the groups are labelled or titled.

Two varieties of categorization are commonly studied in menu system research. The first is hierarchical structures, which organize options into groups across menus. The second is within-menu categorization, which organizes options into groups within a single menu frame. Practical systems generally must consider both types of categorization; the choice between the types depends upon both system characteristics and the relative advantages conferred by the different forms of organization.

### **Who should categorize?**

When forming categories, the question naturally arises of who should do the sorting. Studies on categorization have employed naive users,

experienced users, programmers, system designers, domain experts, the investigator himself, and sorting programs. Rosch suggests that categories are best formed by mapping the cognitive structure or "perceived world structure" as closely as possible.(66) This implies that menu categories should mirror the perceived world structure of typical users, since they are the ones who make search decisions based on menu categories.

Naive or typical users have often generated successful menu category systems. Experts, on the other hand, have generally formed poorer category systems than naive users.(47, 67) Hayhoe and Fischhoff *et al.* present strong evidence that categories formed by scaling analysis of user sortings are superior to expert categories.(68)

Perhaps surprisingly, personalized (i.e., self-created) category systems are often inferior.(64, 68) The quality of personal category systems is predictable, however; Raymond *et al.* showed that quality is correlated with the individual's assessment of the variability of the categories.(69) It seems that experts often have poor cognitive models of typical users, while individual users vary too much over time in the labels assigned to a category. Thus, user-created categories are most reliable when combined with the judgements of many other users.

How can the sorting judgements of many users be combined? A single final categorization can be produced by an expert or by the users themselves, but the most common approach is to use an automatic technique.(58, 64, 67, 70-72) Hayhoe presents an excellent discussion of the use of scaling techniques for creating menu categories.(68) Such techniques are based on analysis of the similarity matrix. This matrix can be constructed in many different ways (e.g., paired comparisons), but the most common approach is to use sorting data. Entries in the matrix represent the frequency with which a sample of people place a given pair of items together in the same group. The scaling techniques extract from the similarity matrix the "best" grouping of options. Each technique uses a different approach to minimax, that is, to minimize the similarity between items in different categories and maximize the similarity between items within the same category. The most popular scaling techniques for analyzing sorting data for menus are hierarchical cluster analysis (HC) and multidimensional scaling

(MDS). Many of these scaling techniques are relatively accessible on statistical packages such as SPSSX or SAS.

One limitation of scaling techniques is that they require an adequate sample size; Hayhoe recommends a minimum of 50 subjects. A second consideration is that the different scaling techniques do not produce the same "optimal" menu categories.

#### **Titling categories.**

In multi-level hierarchical menus, the options on a given menu are typically the category titles for options at the next lower level of the hierarchy. Within a given menu frame, distinct categories have not always been labelled with a title, though the psychological literature suggests the potential utility of titles.(73,74) Empirical research on menus indicate that user performance can be enhanced by category titles that clearly and accurately convey their contents.(47, 75) Moreover, a major cause of category decision errors is vague category titles.(68) Various studies report significantly improved user search performance by modifying misleading category titles on existing menu frames in operational systems.(47, 76, 77)

As with sorting, it seems that the best titles are based on judgments obtained from a sample of users, and not from either experts or from single users. Typical users have generally been found to be as good as or better than experts at titling categories.(47, 52, 67, 68, 75, 78) Furthermore, titles made by the investigator with help from subjects are superior to either titles assigned by experts, investigator-only titles, or personalized titles.(67, 68, 79)

However category titles are formed, care needs to be taken to ensure that titles are distinctive. Schwartz *et al.* describe methods for enhancing distinctiveness among category titles based on Tversky's work.(76)

#### **Effectiveness of categorization.**

Improving categorization techniques has proved to be an effective method of improving search performance, both across and within menus.(47, 57, 58, 75, 80-83) Users have generally performed better with categorized menus than with randomly organized menus,(39, 57-59, 80, 84) though one experiment found no difference between categorized and alphabetic organizations.(85) If the options within categories are further categorized, performance can be

further enhanced for general searches, though not for explicit target searches.(57, 59)

The effectiveness of categorization within a menu depends upon several presentational factors such as proximity, spacing, and row versus columnar organization. Search time for targets is faster if spacing (e.g., a blank line) is provided between different categories within a column on a menu,(57) and if options within a given category are arranged vertically within the same column rather than within a row. Each technique used individually reduces search time by about 25%. Together, spacing and columnar organization reduce search time by 50%. The effectiveness of grouping is markedly dependent upon good layout.(57, 86)

#### **Menu selection errors.**

To find a target, users must be able to correctly identify the category in which it has been classified. Errors arise when menu designers employ terminology and categorizations which do not correspond well with those of the user.(47, 52, 87)

Menu selection errors are caused by one or more of *miscategorization*, *overlapping categories*, and *vague category titles*.(47, 52, 68, 88-90) Miscategorization is the location of desired options in an unexpected category. Category overlap is the perception that two or more categories are appropriate for a given option. Vague titling is the situation in which users are unsure what the contents of a given category might be. Lee *et al.* deliberately created errors of each type in a menu-driven database.(47) All errors significantly degraded user performance. Miscategorization errors were the most serious, however, degrading performance twice as much as the other two types of error.

#### **IV C. Enhancing menu hierarchies.**

The difficulties that arise in generating good categorizations have led to a variety of techniques for enhancing menu-driven systems. The main approaches include identifying and correcting the most erroneous menus, adding descriptors to category options, and adding keyword access.

Empirical testing of typical users can be an effective method of identifying menus exhibiting miscategorization, mistitling, and overlapping categories. Problem menus can then be modified to correct the design errors. The value of modifying problem menus by correcting these kinds of problem is exemplified by marked performance

and preference improvements achieved in two separate studies: search time was reduced by 10% to 37%, errors were reduced by up to 53%, and failure rates were reduced by 45% to 85%.(76, 77)

### **Descriptors.**

A second method of reducing menu selection errors is to clarify the contents of the category options displayed on a menu by adding some form of descriptor to each option. Descriptors most frequently take the form of the options from the next lower level of the menu structure. In effect, two levels of a menu hierarchy are displayed simultaneously on a single frame. Raymond proposes the display and selection of multiple descriptors (even extending past two levels) to permit parallel search.(36)

The evidence for the value of descriptors suggests they are particularly useful at the problematic top levels of menu structures. Errors can be reduced by as much as 40%-60% by the addition of descriptors to the top levels of the hierarchy,(47, 91, 92) though this improvement is somewhat reduced with practice. The value of descriptors is enhanced by making it possible for the user to select both menu levels (i.e., either the category titles or the descriptors for each category) on a menu. Such a modified menu has been shown to reduce errors by 45% relative to a 6-level structure with only two options per level.(81) The use of descriptors does not seem to improve performance at lower database levels.(91, 92)

Descriptors have also taken the form of examples added to each category label. In one experiment, the addition of as many as three examples per category failed to produce any advantage over category names, though three examples can serve to replace the category name without impairing performance.(52)

### **Menu-keywords.**

Two factors are the source of much difficulty for users of menu-only systems. First, a disproportionate number of user errors occur on the top menu levels. Second, such systems often seem inefficient, tedious, and boring for experienced users, who are forced to traverse the same sequence of menu frames. The addition of menu-keywords to a menu-only system can circumvent both problems. In a menu-keyword system, a menu structure is augmented by a special keyword index in which each keyword is associated with a particular menu. The addition of keywords

which access specific menu frames enable users to bypass the problematic upper levels. By directly accessing the lower-level menu frames, menu-keywords permit users to substantially reduce the number of menu frames which must be sequentially accessed, thereby markedly improving performance. Furthermore, with increasing experience, users learn menu-keywords at deeper and deeper levels of the menu structure. In menu-keyword systems, the user always has the option of using only the menu. Thus, both novices and experts will be accommodated. Moreover, such a hybrid system may also avoid the problems for casual users inherent in any keyword-only system, such as user difficulties with Boolean operators and the lack of information about the system state.(34, 52)

Empirical tests of menu-keyword systems for relative novices have shown 36% reduction in search time, 27% reduction in number of frames accessed, and 50% reduction in rate of failure, as well as marked preferences for menu-keywords.(45, 93, 94) In addition, menu-keyword systems are easier to learn and have fewer errors than type-ahead approaches to skipping menu frames.(43)

The performance improvements due to menu-keywords increase with experience. In one study, average search time fell from 340 seconds to 10 seconds for subjects who used keywords frequently; subjects who used keywords infrequently improved only to 230 seconds.(44) In a follow-up study of the effects of experience, users were able to reduce search times by over 50%. Search time and number of menu frames accessed per search were almost perfectly correlated, suggesting that the reduction in search time with experience was attributable to users accessing the menu structure at increasingly deeper levels with increasing experience.

Observing that subjects often assume menu titles are keywords, several researchers have recommended that titles be included in the keyword list for a frame.(43, 44)

### **IV D. Depth versus breadth.**

Most large menu systems are organized hierarchically in a tree-shaped structure varying in breadth, defined as the average number of options per menu, and depth, defined as the average number of menus accessed in retrieving target information. By trading depth for breadth (or vice versa), many alternative menu structures can be

constructed for any given database. For example, for a database of 64 items, a menu hierarchy could consist of 1 menu containing 64 options, 2 levels with 8 options per menu, 3 levels with 4 options per menu, 6 levels with 2 options per menu, and many other structures if menus have a variable number of options. Speed, accuracy, and preference have all been found to vary with breadth and depth. Thus it seems reasonable to consider the problem of determining an optimal breadth and depth for a given size of database.

### Theoretical studies.

Early recommendations were more prescriptive than theoretical or empirical. Shneiderman,(95) for example, prescribed a maximum breadth of seven options per menu, presumably based on Miller’s famous proposition that we best remember items in groups of seven.(96) Similarly, Robertson *et al.* recommended the use of no more than six options per menu, emphasizing the principle of “frame simplicity”.(16) Norman gave an analysis based on “user satisfaction”, which suggested maximal breadth was better for novices and minimal breadth was better for experts.(97)

Theoretical approaches to the depth/breadth issue fall into systems approaches and information theoretic approaches. The systems approaches abstract the essential characteristics of the menu retrieval system, composed of user, hardware, and software components.(48, 49, 57, 62, 98) The Lee and MacGregor model begins with the observation that hierarchical menu structures on  $n$  database items obey an inverse relationship between the breadth or number of options per menu  $b$  and the depth or number of levels in the hierarchy  $d$  :

$$d = \frac{\ln n}{\ln b}$$

The total search time through the index,  $ST$ , is given by the product of the number of menus accessed and the average access time per menu:

$$ST = d(E(I)t + k + c)$$

where  $E(I)$  is the expected number of items examined by a user on one menu frame before making a decision,  $t$  is the time to read or process one option,  $k$  is human response time (including keypress time), and  $c$  is computer response time. Assuming a self terminating search strategy,

$$E(I) = \frac{b+1}{2}$$

Values for  $E(I)$  can also be derived for other user search strategies; for example, in exhaustive search  $E(I) = b$ .

Search time is, in effect, a U-shaped function of the number of options per menu. Search time is greatest for very small and very large values of  $b$ ; the precise values depend on the other parameters of the model.

The number of options per page,  $b$ , that minimizes search time can be found by substituting for  $d$  and setting the derivative of the search time equation with respect to  $b$  equal to zero. The optimum  $b$  (assuming self-terminating search) is given by:

$$b(\ln b - 1) = 1 + 2 \frac{k+c}{t}$$

In general, this equation gives the optimum breadth whenever options are randomly ordered on a menu or whenever the options are organized but the user is unable to take advantage of the organization. For many practical situations optimum breadth is in the range of 4 to 8 options per menu.

If we assume non-random search strategies, grouping can be used to further restrict search.(81) If options are grouped into effective categories within a menu frame, users may first locate the relevant category, and then locate the relevant option within the category. Paap and Roske-Hofstrand(99) assert that the optimum number of groups  $g$  for a menu frame containing  $b$  total options is  $g = \sqrt{b}$ . Assuming that category labels and menu options are processed equally quickly by the user, the expected number of items (options and group labels) examined per frame is  $E(I) = \sqrt{b} + 1$ , and the optimum breadth for self-terminating search is given by:

$$\sqrt{b} \ln(\sqrt{b} - 1) = 1 + \frac{k+c}{t}$$

With grouping, and assuming self-terminating search, Paap and Roske-Hofstrand suggest that optimum breadth ranges from 16 to 78 options for a wide range of practical situations.

It is interesting to note that the optimal number of groups ranges from 4 to 9, with 4 to 9 options within each group, for a wide range of possible systems described by Paap and Roske-Hofstrand. It is no coincidence that this range is virtually the

same as for ungrouped options, since grouping helps for precisely the same reason that hierarchical organization helps — it restricts the options that have to be searched. Items can be organized hierarchically within a single frame, across several frames, or some mixture of the two.

There is, of course, no reason why grouping should be limited to just two levels. Groups may themselves be grouped for an additional reduction in the number of options which need be examined.(48) For example, a set of 64 options on a single menu frame could be grouped in a three-level organization with 4 options in each group. This would require examination of just 7.5 options, on average, rather than 9.0 if only one level of grouping were used.

Fisher *et al.* describe a recursive procedure for identifying the optimal hierarchical menu structure which minimizes user search time.(100) Their equations are derivatives of the Lee and MacGregor and Paap and Roske-Hofstrand search time equations. Their procedure requires, as a starting point or seed, an “easily navigated” menu structure such as might be derived from a hierarchical clustering analysis of sorting data. They then generate all hierarchies derived from the seed where menu frames are eliminated by merging them with a higher level frame. The expected search time is computed for each structure, and the optimum structure is identified. The strength of Fisher’s procedure is that it searches for the optimal menu structure among the set of “semantically well-defined” menu structures. The procedure is, however, computationally intensive.

The preceding analyses assume a linear relation between breadth and search time. Landauer and Nachbar, based on Hick’s law, reject the linear relationship in favour of a log-linear relationship (though they admit the linear relationship may hold for some tasks). Hick’s law states that the average response time in simple decision tasks is a logarithmic function of the number of equally likely options. Thus, Landauer and Nachbar predict search time is minimized by maximizing breadth. It is not clear, however, that menu search in general is a simple decision task.(54, 84) Landauer and Nachbar report a log-linear relationship in searching for specific targets in alphabetically organized lists of words. Many other studies report linear relations.(1, 39, 86, 101, 102) With the exception of Landauer and Nachbar, the system models clearly indicate that the optimum

breadth depends upon such system characteristics as user experience (leading to faster search and decision times), computer response time, and the presence or absence of option grouping.

More recently, an information theoretic analysis of menu structure proposed by Norman and Chin also predicts maximum breadth will be optimal.(103) This prediction is based on the assertion that total uncertainty, which is obtained by summing the choice uncertainties of all but the last menu level, will decrease with increasing breadth. Unfortunately, the theory fails to explain why intermediate breadths are sometimes optimal,(104) or why a random ordering of 64 options on a single menu is worse than that for intermediate breadths.(57)

These results can be accounted for, however, by assuming that the purpose in menu retrieval is to maximize the rate of information transmission. Menu structures of different breadths all transmit the same total amount of information, but they differ in the rate at which it is transmitted to the user. Since the information transmitted is the same for all breadths, rate of transfer depends only upon the amount of time required to transmit the information. Thus, the rate of information transmission is maximized by the speed of transmission (i.e., search time). Predictions from this variation of information theory are similar to those made by system models.

### **Depth versus breadth: empirical results.**

Many empirical studies have examined the effects of varying breadths on user performance and preferences. Most studies examine situations closely resembling command menus, involving small databases, single-word options, and explicit targets in which the target is the actual word at the bottom of the hierarchy.(50, 53, 62, 71, 80, 82, 103, 105-110) These studies are summarized in Table 1.

While optimum breadth is often in the range of 4 to 16 options, it varies considerably with differences in task, user experience, and system characteristics. In some studies, the depth of the menu was confounded with the number of responses (i.e., keypresses): the breadth 64 menu had 1 level and required 1 user selection response, the breadth 8 menu had two levels and required 2 responses, and so on. Using the same 64-item database, Parkinson *et al.* separated these effects experimentally and found that the number of responses was the dominant determinant of search

time.(57) Search time was minimized by minimizing the number of responses, either by increasing breadth or by adding lower level options to the options on a menu frame. While provocative, these conclusions are limited somewhat by the range of values tested in their experiment, since the number of options on a menu varied only from 2 to 6. The results are what one would expect according to systems theory when response time greatly exceeds processing time. This study is also somewhat artificial in that no errors were permitted.

Several studies used more naturalistic search questions, multi-word options, larger databases, and operational systems (or simulations thereof). MacGregor *et al.* used menus with 2, 4, 8, and 16 options per menu adapted from an operational videotex (database) system consisting of 900 documents.(50, 111) The 4 options per menu structure had almost half the error rate of the other three structures. Since a single-page method was used, there was no direct measure of total search time, but extrapolations indicated that 4 options per menu would have minimized total search time.

Tullis failed to find any effect of breadth on either search time or errors in a comparison of two menu structures for accomplishing a set of 24 tasks such as sending messages and printing a file directory.(71) The two menu structures differed in breadth and depth: tasks could be accomplished using 1 to 2 successive menu frames on the broad menu (up to 45 options per menu) or using 1 to 4 menu frames (up to 15 options per menu).

For a database of 256 videotex-like items, Norman and Chin attempted to separate the effects of breadth from those of depth by examining five menu structures equivalent in depth (4 levels) but varying in breadth (4-4-4-4, 8-8-2-2, 2-2-8-8, 2-8-8-2, and 8-2-2-8).(103) Search time did not differ for explicit targets, but did for scenario targets: the 8-2-2-8 menu was faster than the 2-8-8-2 menu. The results for efficiency were similar. Thus, breadth at the bottom and top of a hierarchy appear to have some beneficial effect. Unfortunately, multiple comparison tests were not reported, so it is not possible to tell if other conditions differed significantly.

For a 64-item database, Van Hoe *et al.* tested 15 different menu structures varying in depth (2, 3, 4, or 5 levels) and breadth (increasing breadth at bottom of hierarchy, peaked in centre, and decreasing).(101) Search time systematically

increased with increasing depth. Breadth variations had little effect except at a depth of 2, where the 8-8 condition was significantly faster than either the 4-16 or the 16-4 menu structures. Overall, performance was best for the 8-8 menu structure. Van Hoe *et al.* also found that search time per frame was an increasing linear function of the number of options per menu (not the log-linear relationship predicted by Landauer and Nachbar).

### **What can be learned from depth/breadth studies?**

Depth/breadth tradeoff is the single most studied aspect of menu-driven systems, yet it is probably the most misunderstood. Of ten empirical studies, most show that breadth less than 16 options is either optimal or very close to optimal. There are sound theoretical models which explain this range of breadth, and many operational menu-driven systems have been built within this range — Word Perfect, Lotus, Procomm, Smartcom, and Wordstar have mean breadths of 7.8, 6.0, 5.4 and 4.7 options per menu. The research community, however, generally leans towards the view that broader menus are better. What should the practitioner believe? Practitioners should realize that the question itself is ultimately not an essential one. Most of the thoughtful work on depth/breadth tradeoff warns that search task, system response time, and user characteristics can affect performance more than the choice of menu size. The “optimal” breadth and depth of a menu system are often among the least important factors in organizing a hierarchy.

### **V. The outlook.**

Menu-driven systems are firmly entrenched in the modern microcomputer and show no sign of being discarded in the near future. Sophisticated software is now almost always enhanced by user-selectable icons, buttons, text, figures, or other menu-like artifacts that streamline the user’s task. Important differences between software packages are less often rooted in their functionality, and more frequently found in their menu-driven interface. Indeed, menu-driven interfaces are now considered so significant as to be the subject of legal proceedings.(112) It will be interesting to see how such issues will affect hypertexts, alternate realities, and other systems where the data itself is both the menu and its structure.

Researchers of menu-driven systems should use the emergence of new systems to reconsider their agenda. There has been no lack of work on menu-driven systems, but few solid guidelines have resulted. Research in this area suffers from two basic problems: experts have poor cognitive models of typical users, and individual users vary too much over time. Future work must more carefully control and characterize both the user group and the task, as these often affect the experimental result more than the menu artifact under study. Moreover, the increasing variety and complexity of menu-driven systems suggests that traditional notions of “menu” and “interface” are losing their applicability. We encourage practitioners and researchers to work towards the identification of more fundamental, pervasive phenomena in menu-driven systems.

## References

1. Gary Perlman, "Making the Right Choices With Menus," *INTERACT '84: First IFIP Conference on Human-Computer Interaction*, **1** pp. 291-295 (September 4-7, 1984).
2. Nancy C. Goodwin, "Designing a Multipurpose Menu Driven User Interface to Computer Based Tools," *Proceedings of the Human Factors Society 27th Annual Meeting*, pp. 816-820 (October 10-13, 1983).
3. Neff Walker and John B. Smelcer, "A Comparison of Selection Times from Walking and Pull-Down Menus," *Proceedings of the 1990 SIGCHI Conference on Human Factors in Computing Systems*, pp. 221-225 (April 1-5, 1990).
4. Andrew J. Palay, Wilfred J. Hansen, Michael L. Kazar, Mark Sherman, Maria G. Wadlow, Thomas P. Neuendorffer, Zalman Stern, Miles Bader, and Thom Peters, *The Andrew Toolkit: An Overview*, USENIX Technical Conference, Dallas, Texas (February 9-12, 1988).
5. Miles Macleod and Penelope Tillson, "Pull-Down, Holddown, or Staydown? A Theoretical and Empirical Comparison of Three Menu Designs," *Proceedings of the INTERACT '90 IFIP Conference on Human-Computer Interaction*, pp. 429-434 (August 27-31, 1990).
6. Michael J. Miller, "Multifunctional Cursor for Direct Manipulation User Interfaces," *Proceedings of the CHI '88 Conference on Human Factors in Computing Systems*, pp. 89-94 (May 15-19, 1988).
7. Robert Akscyn, Donald McCracken, and Elise Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations," *Hypertext '87*, pp. 1-20 (November 13-15, 1987).
8. Darrell R. Raymond, "lector — An Interactive Formatter for Tagged Text," CS-90-41, Department of Computer Science, University of Waterloo, Waterloo, Ontario (October 1990).
9. James Boritz, "Of Mice and Menus," CS-90-38, Department of Computer Science, University of Waterloo, Waterloo, Ontario (September 1990).
10. Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman, "An Empirical Comparison of Pie Vs. Linear Menus," *Proceedings of the CHI '88 Conference on Human Factors in Computing Systems*, pp. 95-100 (May 15-19, 1988).
11. Diane L. Souvaine and Christopher J. Van Wyk, "How Hard Can it Be to Draw a Pie Chart?," *Mathematics Magazine*, **63**(3) pp. 165-172 (June 1990).
12. D.L. Fisher, B.G. Coury, and T.O. Tengs, "Optimizing the Set of Highlighted Options on Video Display Terminal Menus," *Proceedings of the Human Factors Society 29th Annual Meeting*, pp. 650-654 (1985).
13. D.L. Fisher, B.G. Coury, T.O. Tengs, and S.A. Duffy, "Minimizing the Time to Search Visual Displays: The Role of Highlighting," *Human Factors*, **31** pp. 167-182 (1989).
14. Ellen P. Francik and Richard M. Kane, "Optimizing Visual Search and Cursor Movement in Pull-Down Menus," *Proceedings of the Human Factors Society 31st Annual Meeting*, **1** pp. 722-726 (October 19-23, 1987).
15. Jan R. Schultz and Layton Davis, "The Technology of PROMIS," *Proceedings of the IEEE*, **67**(9) pp. 1237-1244 (September 1979).
16. G. Robertson, D. McCracken, and A. Newell, "The ZOG Approach to Man-Machine Communication," *International Journal of Man-Machine Studies*, **14** pp. 461-488 (1981).
17. Donald L. McCracken and Robert M. Akscyn, "Experience with the ZOG Human-Computer Interface System," *International Journal of Man-Machine Studies*, **21** pp. 293-310 (1984).
18. Robert M. Akscyn and Donald L. McCracken, "ZOG and the U.S.S. CARL VINSON: Lessons in System Development," *INTERACT '84: First IFIP Conference on Human-Computer Interaction*, **2** pp. 303-308 (September 4-7, 1984).
19. Darrell R. Raymond, "A Survey of Research in Computer-Based Menus," CS-86-61, Department of Computer Science, University of Waterloo, Waterloo, Ontario (November 1986).

20. Marilyn M. Mantei, "Disorientation Behavior in Person-Computer Interaction," *Ph.D. Dissertation*, University of Southern California, (August 1982).
21. Jan Gecsei, *The Architecture of Videotex Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1983).
22. Darrell R. Raymond, "Why Videotex is (still) a Failure," *Canadian Journal of Information Science*, **14**(1) pp. 27-38 (March 1989).
23. Steven Levy, *Hackers: Heroes of the Computer Revolution*, Dell Publishing Co., Inc., New York, N.Y. (1985).
24. Douglas C. Engelbart and William K. English, "A Research Center for Augmenting Human Intellect," *Proceedings of the AFIPS Fall Joint Computer Conference*, **33** pp. 395-410 (December 1968).
25. Douglas C. Engelbart, Richard W. Watson, and James C. Norton, "The Augmented Knowledge Workshop," *Proceedings of the 1973 AFIPS National Computer Conference*, pp. 9-20 (1973).
26. Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Kevin Mackey, "The Xerox Star: A Retrospective," *IEEE Computer*, pp. 11-28 (September 1989).
27. James Gosling, David S.H. Rosenthal, and Michelle J. Arden, *The NeWS Book: An Introduction to the Network/extensible Window System*, Springer-Verlag, New York, N.Y. (1989).
28. Eric A. Bier and Aaron Goodisman, "Documents as User Interfaces," *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography*, pp. 249-262 (September 1990).
29. Jeff Conklin, "Hypertext: An Introduction and Survey," *IEEE Computer*, **20**(9) pp. 17-41 (September 1987).
30. Randall B. Smith, "Experiences with the Alternate Reality Kit: An Example of the Tension Between Literalism and Magic," *Proceedings of the 1987 CHI + GI Conference on Human Factors in Computing Systems and Graphics Interface*, pp. 61-67 (April 5-9, 1987).
31. Randall B. Smith, "The Alternative Reality Kit: An Animated Environment for Creating Interactive Simulations," *Proceedings of the 1986 IEEE Workshop on Visual Languages*, (June 1986).
32. Diana Parton, Keith Huffman, Patty Pridgen, Kent Norman, and Ben Shneiderman, "Learning a Menu Selection Tree: Methods Compared," *Behaviour and Information Technology*, **4**(2) pp. 81-91 (1985).
33. Darrell R. Raymond and Heather J. Fawcett, "Playing Detective with Full Text Searching Software," *SIGDOC '90*, **14**(4) pp. 157-166 (October 31-November 2, 1990).
34. James N. MacGregor and Eric S. Lee, "Performance and Preference in Videotex Menu Retrieval: A Review of the Empirical Literature," *Behaviour and Information Technology*, **6**(1) pp. 43-68 (1987).
35. Thomas Whalen, "The Design of Index Systems for Information Retrieval by Naive Users," *20th International Congress of Applied Psychology*, p. 136 (1982).
36. Darrell R. Raymond, "Personal Data Structuring in Videotex," CS-84-7, Department of Computer Science, University of Waterloo, Waterloo, Ontario (February 1984).
37. David C. Blair and M.E. Maron, "An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System," *Communications of the ACM*, **28**(3) pp. 289-299 (March 1985).
38. Simon P. Davies, Anthony J. Lambert, and John M. Findlay, "The Effects of the Availability of Menu Information During Command Learning in a Word Processing Application," *Behaviour and Information Technology*, **8**(2) pp. 135-144 (1989).
39. Stuart K. Card, "User Perceptual Mechanisms in the Search of Computer Command Menus," *Proceedings of the CHI '82 Conference on Human Factors in Computer Science*, pp. 190-196 (March 15-17, 1982).
40. Patricia A. Billingsley, "Navigation Through Hierarchical Menu Structures: Does it Help to Have a Map?," *Proceedings of the 26th Annual Meeting of the Human Factors Society*, pp. 103-107 (October 25-29, 1982).

41. Diana Parton, Keith Huffman, Patty Pridgen, Kent Norman, and Ben Shneiderman, "Learning A Menu Selection Tree: Training Methods Compared," CAR-TR-66/CS-TR-1409, Human/Computer Interaction Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland (June 1984).
42. Kent L. Norman, Jeffrey P. Schwartz, and Ben Shneiderman, "Memory For Menus: Effects of Study Mode," CAR-TR-69/CS-TR-1412, Human/Computer Interaction Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland (June 1984).
43. Alan Laverson, Kent Norman, and Ben Shneiderman, "An Evaluation of Jump-Ahead Techniques in Menu Selection," *Behaviour and Information Technology*, **6** pp. 97-108 (1987).
44. Eric S. Lee and Glenna Chao, "The Increasing Utility of Incorporating Keywords in Menu Systems as Users Increase in Experience," *Behaviour and Information Technology*, **8**(4) pp. 301-308 (1989).
45. Eric S. Lee, James N. MacGregor, Newman Lam, and Glenna Chao, "Keyword-menu Retrieval: An Effective Alternative to Menu Indexes," *Ergonomics*, **29** pp. 115-130 (1986).
46. M.D. Apperley and R. Spence, "Hierarchical Dialogue Structures in Interactive Computer Systems," *Software — Practice and Experience*, **13** pp. 777-790 John Wiley & Sons, Ltd., (1983).
47. Eric S. Lee, Thom Whalen, Scott McEwen, and Sue Latrémouille, "Optimizing the Design of Menu Pages for Information Retrieval," *Ergonomics*, **27**(10) pp. 1051-1069 (1984).
48. Eric S. Lee and James MacGregor, "Minimizing User Search Time in Menu Retrieval Systems," *Human Factors*, **27**(2) pp. 157-162 (April 1985).
49. Gordon T. Uber, Paul E. Williams, Bradner L. Hisey, and Robert G. Siekert, "The Organization and Formatting of Hierarchical Displays for the On-Line Input of Data," *Proceedings of the 1968 AFIPS Fall Joint Computer Conference*, pp. 219-226 (December 9-11, 1968).
50. James N. MacGregor, Eric S. Lee, and Newman Lam, "Optimizing the Structure of Database Menu Indexes: A Decision Model of Menu Search," *Human Factors*, **28**(4) pp. 387-399 (1986).
51. Eric S. Lee, "The Optimum Number of Alternatives to Display on an Index Page in an Interactive Telidon Database," *Telidon Behavioural Research I*, pp. 128-149 Department of Communications, Government of Canada, (1980).
52. Susan T. Dumais and Thomas K. Landauer, "Using Examples to Describe Categories," *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems*, pp. 112-115 (December 12-15, 1983).
53. André Vandierendonck, Rudy Van Hoe, and Geert De Soete, "Menu Search as a Function of Menu Organization, Categorization, and Experience," *Acta Psychologica*, **69** pp. 231-248 (1988).
54. Kenneth R. Paap, R. Noel, J. McDonald, and Renate Roske-Hofstrand, "Optimal Organizations Guided by Cognitive Networks and Verified by Eye-Movement Analyses," *Proceedings of the Interact '87 Conference on Human-Computer Interaction*, pp. 617-622 (1987).
55. James N. MacGregor and Eric S. Lee, "Menu Search: Random or Systematic?," *International Journal of Man-Machine Studies*, **26** pp. 627-631 (1987).
56. Jeffrey J. Hendrickson, "Performance, Preference, and Visual Scan Patterns on a Menu-Based System: Implications for Interface Design," *Proceedings of the '89 SIG-CHI Conference on Human Factors in Computing Systems*, pp. 217-222 (April 30-May 4, 1989).
57. Stanley R. Parkinson, Norwood Sisson, and Kathleen Snowberry, "Organization of Broad Computer Menu Displays," *International Journal of Man-Machine Studies*, **23** pp. 689-697 (1985).
58. J.G. Hollands and P.M. Merikle, "Menu Organization and User Expertise in Information Search Tasks," *Human Factors*, **29** pp. 577-586 (1987).
59. James E. McDonald, Jim D. Stone, and Linda S. Liebelt, "Searching for Items in Menus: The Effects of Organization and

- Type of Target,” *Proceedings of the 27th Annual Meeting of the Human Factors Society*, pp. 834-837 (October 10-14, 1983).
60. Brad Mehlenbacher, Thomas M. Duffy, and James Palmer, “Finding Information on a Menu: Linking Menu Organization to the User’s Goals,” *Human-Computer Interaction*, **4**(3) pp. 231-250 (1989).
  61. Saul Greenberg and Ian H. Witten, “Comparison of Menu Displays for Ordered Lists,” *Proceedings of the Canadian Information Processing Society Session ’84*, pp. 464-469 (May 9-11, 1984).
  62. Thomas K. Landauer and D.W. Nachbar, “Selection From Alphabetic and Numeric Menu Trees Using a Touch Screen: Depth, Breadth, and Width,” *Proceedings of the CHI ’85 Conference on Human Factors in Computing Systems*, pp. 73-78 (April 14-18, 1985).
  63. Saul Greenberg and Ian H. Witten, “Adaptive Personalized Interfaces — A Question of Viability,” *Behaviour and Information Technology*, **4**(1) pp. 31-45 (1985).
  64. J.E. McDonald, T.S. Dayton, and D.R. McDonald, “Adapting Menu Layout to Tasks,” *International Journal of Man-Machine Studies*, **28** pp. 417-435 (1988).
  65. Benjamin L. Somberg, “A Comparison of Rule-Based and Positionally Constant Arrangements of Computer Menu Items,” *Proceedings of the CHI + GI 1987 Conference on Human Factors in Computing Systems and Graphics Interface*, pp. 255-260 (April 5-9, 1987).
  66. E. Rosch, “Principles of Categorization,” pp. 27-48 in *Cognition and Categorization*, ed. B.B. Lloyd, Erlbaum, Hillsdale, N.J. (1978).
  67. B. Fischhoff, D. McGregor, and L. Blackshaw, “Creating Categories for Databases,” *International Journal of Man-Machine Studies*, **27** pp. 33-63 (1987).
  68. D. Hayhoe, “Sorting-Based Menu Categories,” *International Journal of Man-Machine Studies*, **33** pp. 677-705 (1990).
  69. Darrell R. Raymond, Alberto J. Cañas, Frank Wm. Tompa, and Frank R. Safayeni, *Measuring the Effectiveness of Personal Database Structures*, International Journal of Man-Machine Studies (September 1989).
  70. Kathleen M. Snyder, Alan J. Happ, Lawrence Malcus, Kenneth R. Paap, and James R. Lewis, “Using Cognitive Models to Create Menus,” *Proceedings of the Human Factors Society 29th Annual Meeting*, pp. 655-658 (1985).
  71. Thomas S. Tullis, “Designing a Menu-Based Interface to an Operating System,” *Proceedings of the CHI ’85 Conference on Human Factors in Computing Systems*, pp. 79-84 (April 14-18, 1985).
  72. James E. McDonald, Jim D. Stone, Linda S. Leibel, and John Karat, “Evaluating a Method for Structuring The User-System Interface,” *Proceedings of the 26th Annual Meeting of the Human Factors Society*, pp. 551-555 (October 25-29, 1982).
  73. G. Murphy and D. Medin, “The Role of Theories in Conceptual Coherence,” *Psychological Review*, **92** pp. 289-316 (1985).
  74. Lawrence A. Barsalou, “Ad Hoc Categories,” *Memory & Cognition*, **11**(3) pp. 211-227 Psychonomic Society, Inc., (1983).
  75. George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais, “Statistical Semantics: Analysis of the Potential Performance of Keyword Information Systems,” *The Bell System Technical Journal*, **62**(6) pp. 1753-1806 (July-August 1983).
  76. Jeffrey P. Schwartz and Kent L. Norman, “The Importance of Item Distinctiveness on Performance Using a Menu Selection System,” *Behaviour and Information Technology*, **5**(2) pp. 173-182 (1986).
  77. Ricky E. Savage, James K. Habinek, and Thomas W. Barnhart, “The Design, Simulation, and Evaluation of a Menu Driven User Interface,” *Proceedings of the CHI ’82 Conference on Human Factors in Computer Science*, pp. 36-40 (March 15-17, 1982).
  78. Donald E. Broadbent and Margaret H.P. Broadbent, “The Allocation of Descriptor Terms by Individuals in a Simulated Retrieval System,” *Ergonomics*, **21**(5) pp. 343-354 (1978).

79. Thomas K. Landauer, "Relations Between Cognitive Psychology and Computer System Design," pp. 1-25 in *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, ed. J.M. Carroll, MIT Press, Cambridge, Massachusetts (1987).
80. J.L. Harpster, "Random Versus Ordered Menus in Self-Terminating Menu Searches," *Proceedings of the Human Factors Society 31st Annual Meeting*, **1** pp. 718-721 (October 19-23, 1987).
81. Stanley R. Parkinson, Martin D. Hill, Norwood Sisson, and Cynthia Viera, "Effects of Breadth, Depth, and Number of Responses On Computer Menu Search Performance," *International Journal of Man-Machine Studies*, **28** pp. 683-692 (1988).
82. Kathleen Snowberry, Stanley R. Parkinson, and Norwood Sisson, "Computer Display Menus," *Ergonomics*, **26**(7) pp. 699-712 (1983).
83. P.J. Barnard, J. Morton, J.B. Long, and E.A. Ottley, "Planning Menus for Display: Some Effects of Their Structure and Content on User Performance," *Displays for Man-Machine Systems*, pp. 130-133 (1977).
84. Luc Giroux and Rachel Belleau, "What's on the Menu? The Influence of Menu Content on the Selection Process," *Behaviour and Information Technology*, **5**(2) pp. 169-172 (1986).
85. Jo W. Tombaugh and Scott A. McEwen, "Comparison of Two Information Retrieval Methods on Videotex: Tree-Structure Versus Alphabetical Directory," *Proceedings of the CHI '82 Conference on Human Factors in Computer Science*, pp. 106-110 (March 15-17, 1982).
86. R.W. Backs, L.C. Walrath, and G.A. Hancock, "Comparison of Horizontal and Vertical Menu Formats," *Proceedings of the Human Factors Society 31st Annual Meeting*, pp. 715-717 (October 19-23, 1987).
87. Calvin Kingsley Clauer, "An Experimental Evaluation of Hierarchical Decision-Making for Information Retrieval," IBM RJ 1093 (#17928), IBM Research Laboratory, San Jose, California (September 15, 1972).
88. Thomas Whalen and Candy Mason, "The Use of a Tree-Structured Index Which Contains Three Types of Design Defects," pp. 15-34 in *The Design of Videotex Tree Indexes*, Behavioural Research and Evaluation, Department of Communications, Government of Canada, Ottawa, Ontario (May 1981).
89. R.M. Young and A. Hull, "Cognitive Aspects of the Selection of Viewdata Options by Casual Users," *Proceedings of the Sixth International Conference on Computer Communications*, pp. 571-576 (September 7-10, 1982).
90. R.M. Young and A. Hull, "Categorization Structures in Hierarchical Menus," *Proceedings of the Tenth International Symposium on Human Factors in Telecommunications*, (1983).
91. Kathleen Snowberry, Stanley Parkinson, and Norwood Sisson, "Effects of Help Fields on Navigating Through Hierarchical Menu Structures," *International Journal of Man-Machine Studies*, **22** pp. 479-491 (1985).
92. C. Mayson-Maddison and Eric S. Lee, "User Search Performance and the Use of Descriptors in Two Videotex Menu Indexes," OER81-3600, Department of Communications, Government of Canada, Ottawa, Canada (1982).
93. T. Stewart, "Prestel — How Usable Is It?," pp. 107-117 in *Human Aspects of Telecommunication: Individual and Social Consequences*, ed. E. Witte, Springer-Verlag, Berlin (1980).
94. B.A. Weerdmeester, R.H. van Velthoven, and T.G.M. Vrins, "Keywords for Information Retrieval on Interactive Videotex," *Behaviour and Information Technology*, **4** pp. 103-112 (1985).
95. Ben Shneiderman, *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop, Cambridge, Massachusetts (1980).
96. George A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits On Our Capacity for Processing Information," *The Psychological Review*, **63**(2) pp. 81-97 (March 1956).

97. Donald A. Norman, "Design Principles for Human-Computer Interfaces," *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems*, pp. 1-10 (December 12-15, 1983).
98. David A. Thompson, Lawrence A. Bennisson, and David Whitman, "A Proposed Structure for Displayed Information to Minimize Search Time Through a Data Base," pp. 452-455 in *Introduction to Information Science*, ed. Tefko Saracevic, R.R. Bowker & Co. (1970).
99. Kenneth R. Paap and Renate J. Roske-Hofstrand, "The Optimal Number of Menu Options per Panel," *Human Factors*, **28**(4) pp. 377-385 (1986).
100. D.L. Fisher, E. Yungkurth, and S.M. Moss, "Optimal Menu Hierarchy Design: Syntax and Semantics," *Human Factors*, **32** pp. 665-683 (1990).
101. Rudy Van Hoe, Karel Poupeye, André Vandierendonck, and Geert De Soete, "Some Effects of Menu Characteristics and User Personality on Performance with Menu-Driven Interfaces," *Behaviour and Information Technology*, **9**(1) pp. 17-29 (1990).
102. Benjamin L. Somberg, George J. Boggs, and Maria C. Picardi, "Search and Decision Processes in Human Interaction With Menu-Driven Systems," *The Human Factors Society 26th Annual Meeting*, (October 1982).
103. Kent L. Norman and John P. Chin, "The Effect of Tree Structure on Search in a Hierarchical Menu Selection System," *Behaviour and Information Technology*, **7** pp. 51-65 (1988).
104. Norwood Sisson, Stanley Parkinson, and Kathleen Snowberry, "Design Methodology for Menu Structures," *IEEE 2nd Annual Phoenix Conference on Computers and Communications*, pp. 557-560 (March 14-16, 1983).
105. S.M. Dray, W.G. Ogden, and R.E. Vestewig, "Measuring Performance with a Menu-Selection Human-Computer Interface," *Proceedings of the 25th Annual Meeting of the Human Factors Society*, pp. 746-748 (October 12-16, 1981).
106. Dwight P. Miller, "The Depth/Breadth Tradeoff in Hierarchical Computer Menus," *Proceedings of the 25th Annual Meeting of the Human Factors Society*, pp. 296-300 (October 12-16, 1981).
107. Norwood Sisson, Stanley R. Parkinson, and Kathleen Snowberry, "Considerations of Menu Structure and Communication Rate for the Design of Computer Menu Displays," *International Journal of Man-Machine Studies*, **25** pp. 479-489 (1986).
108. D.F. Wallace, N.S. Anderson, and B. Shneiderman, "Time Stress Effects on Two Menu Selection Systems," *Proceedings of the Human Factors Society 31st Annual Meeting*, **1** pp. 727-731 (October 19-23, 1987).
109. John I. Kiger, "The Depth/Breadth Trade-Off in the Design of Menu-Driven User Interfaces," *International Journal of Man-Machine Studies*, **20** pp. 201-213 (1984).
110. P. Seppala and G. Salvendy, "Impact of Depth of Menu Hierarchy on Performance Effectiveness in a Supervisory Task: Computerized Flexible Manufacturing System," *Human Factors*, **27** pp. 713-722 (1985).
111. James N. MacGregor, Eric S. Lee, and Newman Lam, "The User Search Process in Menu Retrieval and its Implications for the Structuring of Menu Indexes," OER83-01065, Department of Communications, Government of Canada, Ottawa, Canada (1985).
112. Pamela Samuelson, "How to Interpret the Lotus Decision (And How Not To)," *Communications of the ACM*, **33**(11) pp. 27-33 (November 1990).