Exhibit 1014 – Part 3

2.12 Legal Issues

As user interfaces have become prominent, serious legal issues have emerged. Privacy issues are always a concern whenever computers are used to store data or to monitor activity. Medical, legal, financial, military, or other data often have to be protected to prevent unapproved access, illegal tampering, inadvertent loss, or malicious mischief. Physical security to prohibit access is fundamental; in addition, privacy protection can involve user-interface issues for encryption and decryption, password access, file-access control, identity checking, and data verification. Effective protection should provide a high degree of privacy with a minimum of interference.

A second issue is safety and reliability. User interfaces for aircraft, automobiles, medical equipment, military systems, or nuclear-reactor control rooms can affect life-or-death decisions. If an air-traffic controller is temporarily confused by the contents of the display, that could lead to disaster. If the user interface for such a system is demonstrated to be difficult to understand, it could leave the designer, developer, and operator open to a law suit alleging improper design. Designers should strive to make high-quality and well-tested interfaces that adhere to state-of-the-art design guidelines. Documentation of testing and usage should be maintained to provide accurate data on actual performance. Unlike architecture or engineering, user-interface design is not yet an established profession with clear standards.

A third issue is copyright protection for software and information (Clapes, 1989; Menell, 1989; Gilbert, 1990; NRC, 1991). Software developers who have spent time and money to develop a package are frustrated in attempting to recover their costs and make a profit if potential users pirate (make illegal copies of) the package, rather than buy it. Various technical schemes have been tried to prevent copying, but clever hackers can usually circumvent the barriers. It is unusual for a company to sue an individual for copying a program, but cases have been brought against corporations and universities. Site-license agreements are one solution, because they allow copying within a site once the fees have been paid. More complicated situations arise in the context of access to online information. If a customer of an online information service pays for time to access to the database, does the customer have the right to extract and store the retrieved information electronically for later use? Can the customer send an electronic copy to a colleague, or sell a bibliography carefully culled from a large commercial database? Do individuals, their employers, or network operators own the information contained in electronic-mail messages?

A fourth issue is freedom of speech in electronic environments. Is the right to make controversial statements through electronic mail or bulletin-

board systems? Are such statements protected by the First Amendment? Are networks like street corners, where freedom of speech is guaranteed, or are networks like television broadcasting, where community standards must be protected? Should network operators be responsible for or prohibited from eliminating offensive or obscene jokes, stories, or images? Controversy has raged over whether a network operator can prohibit electronic-mail messages that were used to organize a rebellion against the network operators. Another controversy emerged over whether the network operator should suppress racist electronic-mail remarks or postings to a bulletin board. If libelous statements are transmitted, can a person sue the network as well as the source?

The most controversial issue for user-interface designers is that of copyright and patent protection for user interfaces. When user interfaces comprised coded commands in all-capital letters on a Teletype, there was little that could be protected. But the emergence of artistically designed graphic user interfaces with animations and extensive online help has led developers to file for copyright protection. This activity has led to many controversies:

- *What material is eligible for copyright?* Since fonts, lines, boxes, shading, and colors cannot usually be accorded copyrights, some people claim that most interfaces are not protectable. Advocates of strong protection claim that the ensemble of components is a creative work, just like a song that is composed of uncopyrightable notes or a poem of uncopyrightable words. Although standard arrangements, such as the rotated-L format of spreadsheets, are not copyrightable, collections of words, such as the Lotus 1-2-3 menu tree, have been accepted as copyrightable. Maybe the most confusing concept is the separation between ideas (not protectable) and expressions (protectable). Generations of judges and lawyers have wrestled with the issue; they agree only that there is "no bright shining line" between idea and expression, and that the distinction must be decided in each case. Most informed commentators would agree that the idea of working on multiple documents at once by showing multiple windows simultaneously is not protectable, but that specific expressions of windows (border decorations, animations for movement, and so on) might be protectable. A key point is that there should be a variety of ways to express a given idea. When there is only one way to express an idea—for example, a circle for the idea of a wedding ring—the expression is not protectable.
- *Are copyrights or patents more appropriate for user interfaces?* Traditionally, copyright is used for artistic, literary, and musical expressions, whereas patent is used for functional devices. There are interesting crossovers,

such as copyrights for maps, engineering drawing, and decorations on teacups, and patents for software algorithms. Copyrights are easy to obtain (just put a copyright notice on the user interface and file a copyright application), are rapid, and are not verified. Patents are complex, slow, and costly to obtain, since they must be verified by the Patent and Trademark Office. Copyrights last 75 years for companies, and life plus 50 years for individuals. Patents last for only 17 years but are considered more enforceable. The strength of patent protection has raised concerns over patents that were granted for what appear to be fundamental algorithms for data compression and display management. Copyrights for printed user manuals and online help can also be obtained.

- *What constitutes copyright infringement?* If another developer copies your validly copyrighted user interface exactly, that is clearly a case of infringement. More subtle issues arise when a competitor makes a user interface that has elements strikingly similar, by your judgment, to your own. To win a copyright-infringement case, you must convince a jury of "ordinary observers" that the competitor actually saw your interface and that the other interface is "substantially similar" to yours.
- *Should user interfaces be copyrighted?* There are many respected commentators who believe that user interfaces should not be copyrighted. They contend that user interfaces should be shared and that it would impede progress if developers had to pay for permission for every user-interface feature that they saw and wanted to include in their interface. They claim also that copyrights interfere with beneficial standardization and that unnecessary artistic variations would create confusion and inconsistency. Advocates of copyrights for user interfaces wish to recognize creative accomplishments and, by allowing protection, to encourage innovation while ensuring that designers are rewarded for their works. Although ideas are not protectable, specific expressions would have to be licensed from the creator, presumably for a fee, in the same way that each photo in an art book must be licensed and acknowledged, or each use of a song, play, or quote must be granted permission. Concern over the complexity and cost of this process and the unwillingness of copyright owners to share is legitimate, but the alternative of providing no protection might slow innovation.

In the current legal climate, interface designers must respect existing expressions and would be wise to seek licenses or cooperative agreements to share user interfaces. Placing a copyright notice on the title screen of a system and in user manuals seems appropriate. Of course, proper legal counsel should be obtained.

2.13 Practitioner's Summary

Designing user interfaces is a complex and highly creative process that blends intuition, experience, and careful consideration of numerous technical issues. Designers are urged to begin with a thorough task analysis and a careful specification of the user communities. Explicit recording of task objects and actions based on a task analysis can lead to construction of useful metaphors or system images. Identification of computer objects and actions guides designers to develop simpler concepts that benefit novice and expert users. Next, designers create consistent and meaningful syntactic forms for input and display. Extensive testing and iterative refinement are necessary parts of every development project.

Design principles and guidelines are emerging from practical experience and empirical studies. Organizations can benefit by reviewing available guidelines documents and then constructing a local version. A guidelines document records organizational policies, supports consistency, aids the application of dialog-management tools, facilitates training of new designers, records results of practice and experimental testing, and stimulates discussion of user-interface issues.

2.14 Researcher's Agenda

The central problem for psychologists, human-factors professionals, and computer scientists is to develop adequate theories and models of the behavior of humans who use interactive systems. Traditional psychological theories must be extended and refined to accommodate the complex human learning, memory, and problem-solving required in these applications. Useful goals include descriptive taxonomies, explanatory theories, and predictive models.

A first step might be to investigate thoroughly a limited task for a single community, and to develop a formal notation for describing task actions and objects. Then, the mapping to computer actions and objects could be made precisely. Finally, the linkage with syntax would follow. This process would lead to predictions of learning times, performance speeds, error rates, subjective satisfaction, or human retention over time, for competing designs.

Next, the range of tasks and user communities could be expanded to domains of interest such as word processing, information retrieval, or data entry. More limited and applied research problems are connected with each of the hundreds of design principles or guidelines that have been proposed. Each validation of these principles and clarification of the breadth of

applicability would be a small but useful contribution to the emerging mosaic of human performance with interactive systems.

References

Bailey, Robert W., *Human Performance Engineering: Using Human Factors/Ergonomics to Achieve Computer Usability* (Second Edition) Prentice-Hall, Englewood Cliffs, NJ (1989).

Brown, C. Marlin, *Human–Computer Interface Design Guidelines*, Ablex, Norwood, NJ (1988).

Brown, P., and Gould, J., How people create spreadsheets, *ACM Transactions on Office Information Systems 5* (1987), 258–272.

Card, Stuart K., Theory-driven design research, In McMillan, Grant R., Beevis, David, Salas, Eduardo, Strub, Michael H., Sutton, Robert, and Van Breda, Leo (Editors), *Applications of Human Performance Models to System Design*, Plenum Press, New York (1989), 501–509.

Card, Stuart K., Mackinlay, Jock D., and Robertson, George G., The design space of input devices, *Proc.CHI '90 Conference: Human Factors in Computing Systems*, ACM, New York (1990), 117–124.

Card, Stuart, Moran, Thomas P., and Newell, Allen, The keystroke-level model for user performance with interactive systems, *Communications of the ACM 23* (1980), 396–410.

Card, Stuart, Moran, Thomas P., and Newell, Allen, *The Psychology of Human–Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ (1983).

Clapes, Anthony Lawrence, *Software, Copyright, and Competition: The "Look and Feel" of the Law*, Quorum, New York (1989).

Computer Science and Telecommunications Board National Research Council, *Intellectual Property Issues in Software*, National Academy Press, Washington, DC (1991).

Eason, K. D., Dialogue design implications of task allocation between man and computer, *Ergonomics 23*, 9 (1980), 881–891.

Egan, Dennis E., Individual differences in human–computer interaction. In Helander, Martin (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988), 543–568.

Elkerton, Jay and Palmiter, Susan L., Designing help using a GOMS model: An information retrieval evaluation, *Human Factors 33*, 2 (1991), 185–204.

Foley, J. D. & Wallace, V. L. "The Art of Natural Graphic Man–Machine Conversation," Proc. IEEE, 62 (4) Ap. 1974, pp. 462–70.

Gaines, Brian R., The technology of interaction—dialogue programming rules, *International Journal of Man–Machine Studies 14* (1981), 133–150.

Galitz, Wilbert O., *Handbook of Screen Format Design* (Third Edition), Q. E. D. Information Sciences, Wellesley, MA (1989).

Gilbert, Steven W., Information technology, intellectual property, and education, *EDUCOM Review 25* (1990), 14–20.

Grudin, Jonathan, The case against user interface consistency, *Communications of the ACM 32*, 10 (1989), 1164–1173.

Hansen, Wilfred J., User engineering principles for interactive systems, *Proc. Fall Joint Computer Conference 39*, AFIPS Press, Montvale, NJ (1971), 523–532.

Kieras, David, Towards a practical GOMS model methodology for user interface design. In Helander, Martin (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988), 135–157.

Kieras, David, and Polson, Peter G., An approach to the formal analysis of user complexity, *International Journal of Man–Machine Studies 22* (1985), 365–394.

Lockheed Missiles and Space Company, *Human Factors Review of Electric Power Dispatch Control Centers: Volume 2: Detailed Survey Results*, Prepared for Electric Power Research Institute, Palo Alto, CA (1981).

Menell, Peter S., An analysis of the scope of copyright protection for application programs, *Stanford Law Review 41* (1989), 1045–1104.

National Research Council, *Intellectual Property Issues in Software*, National Academy Press, Washington D.C. (1991).

Norcio, Anthony F. and Stanley, Jaki, Adaptive human–computer interfaces: A literature survey and perspective, *IEEE Transactions on Systems, Man, and Cybernetics 19* (1989), 399–408.

Norman, Donald A., Design rules based on analyses of human error, *Communications of the ACM 26*, 4 (1983), 254–258.

Norman, Donald A., *The Psychology of Everyday Things*, Basic Books, New York (1988).

Norman, Kent L., Models of the mind and machine: Information flow and control between humans and computers, *Advances in Computers 32* (1991), 119–172.

Payne, S. J. and Green, T. R. G., Task-Action Grammars: A model of the mental representation of task languages, *Human–Computer Interaction 2* (1986), 93–133.

Payne, S. J. and Green, T. R. G., The structure of command languages: An experiment on Task-Action Grammar, *International Journal of Man–Machine Studies 30* (1989), 213–234.

Reisner, Phyllis, Formal grammar and design of an interactive system, *IEEE Transactions on Software Engineering SE-5* (1981), 229–240.

Reisner, Phyllis, What is consistency? In Diaper et al. (Editors), *INTERACT '90: Human–Computer Interaction*, North-Holland, Amsterdam, The Netherlands (1990), 175–181.

Rubinstein, Richard, and Hersh, Harry, *The Human Factor: Designing Computer Systems for People*, Digital Press, Burlington, MA (1984).

Sears, Andrew, *Widget-Level Models of Human–Computer Interaction: Applying Simple Task Descriptions to Design and Evaluation*, PhD. Dissertation, Department of Computer Science, University of Maryland, College Park, MD (1992).

Sheridan, Thomas B., Task allocation and supervisory control. In Helander, M. (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988), 159–173.

Shneiderman, Ben, *Software Psychology: Human Factors in Computer and Information Systems*, Little, Brown, Boston, MA (1980).

Shneiderman, Ben, A note on the human factors issues of natural language interaction with database systems, *Information Systems 6*, 2 (1981), 125–129.

Shneiderman, Ben, System message design: Guidelines and experimental results. In *Directions in Human–Computer Interaction*, Badre, A. and Shneiderman, B. (Editors), Ablex, Norwood, NJ (1982), 55–78.

Shneiderman, Ben, Direct manipulation: A step beyond programming languages, *IEEE Computer 16*, 8 (1983), 57–69.

Smith, Sid L. and Mosier, Jane N., *Guidelines for Designing User Interface Software*, Report ESD-TR-86-278, Electronic Systems Division, the MITRE Corporation, Bedford, MA (1986). Available from National Technical Information Service, Springfield, VA.

Tufte, Edward, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT (1983).

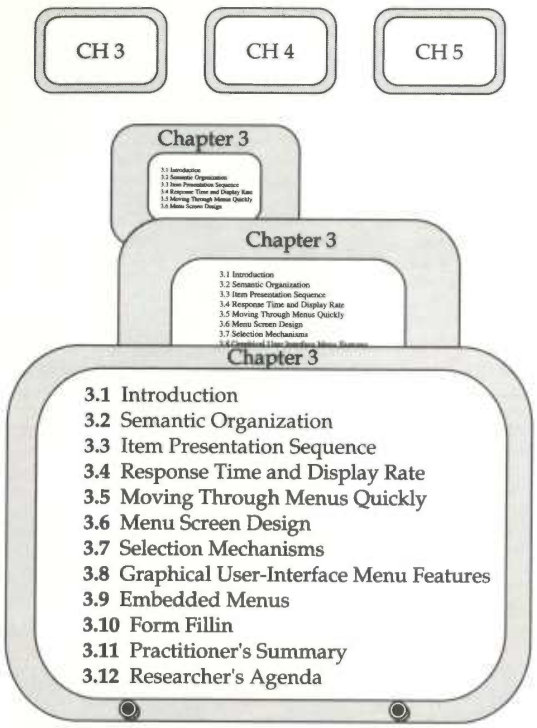
Tufte, Edward, *Envisioning Information*, Graphics Press, Cheshire, CT (1990).

CHAPTER 3

Menu Selection and Form Fillin

A man is responsible for his choice and must accept the consequences, whatever they may be.

W. H. Auden, *A Certain World*



3.1 Introduction

Menu selection is attractive because it can eliminate training and memorization of complex command sequences. When the menu items are written using familiar terminology, users can select an item easily, and can indicate their choice with one or two keypresses or use of a pointing device. This simplified interaction style reduces the possibility of keying errors and structures the task to guide the novice and intermittent user. With careful design and high-speed interaction, menu selection can become appealing to expert frequent users, as well. The success of menu selection in the Macintosh, Microsoft Windows 3.0, or OSF/Motif is an indication of the widespread appeal of seeing choices and of selecting by pointing.

Menu selection is often contrasted with use of command language, but the distinctions are sometimes blurred. Typically, menu selection requires a single keystroke or mouse click, whereas commands may be lengthy; but

how would you classify a menu in which users have to type a six- or eight-letter item? Typically, menu selection displays the choices, whereas commands must be memorized, but how would you classify a menu that offered four numbered choices and accepted 10 more generic commands that are not displayed? The command menu bar in LOTUS 1-2-3 is a success because is allows the duality that novices treat it as a menu and walk through the levels, whereas experienced users construct commands by typing ahead several levels of menu choices. If light can be a wave and a particle, then why should an interface not be a menu and a command?

Rather than debate terminology, it is more useful to maintain an awareness of how much information is on the display at the moment the selection is made, what are the form and content of item selection, and what task domain knowledge is necessary for users to succeed. Menu selection is especially effective when users have little training, use the system only intermittently, are unfamiliar with the terminology, and need help in structuring their decision-making process.

However, if a designer uses menu selection, this choice does not guarantee that the system will be appealing and easy to use. Effective menu-selection systems emerge only after careful consideration of and testing for numerous design issues, such as semantic organization, menu-system structure, number and sequence of menu items, titling, prompting format, graphic layout and design, phrasing of menu items, display rates, response time, shortcuts through the menus for knowledgeable frequent users, on-line help, and selection mechanisms (keyboard, pointing devices, touchscreen, voice, etc.) (Norman, 1991).

3.2 Semantic Organization

The primary goal for menu designers is to create a sensible, comprehensible, memorable, and convenient semantic organization relevant to the user's tasks. We can learn some lessons by following the semantic decomposition of a book into chapters, a program into modules, the animal kingdom into species, or a Sears catalog into sections. Hierarchical decompositions—natural and comprehensible to most people—are appealing because every item belongs to a single category. Unfortunately, in some applications, an item may be difficult to classify as belonging to one category, and the temptation to duplicate entries or to create a network increases. In spite of their limitations, tree structures have an elegance that should be appreciated.

Restaurant menus separate appetizers, soups, main dishes, desserts, and drinks to help customers organize their selections. Menu items should fit logically into categories and have readily understood meanings.

Restauranteurs who list dishes with idiosyncratic names such as "veal Monique," generic terms such as "house dressing," or unfamiliar labels such as "wor shu op" should expect that waiters will spend ample time explaining the alternatives, or should anticipate that customers will become anxious because of the unpredictability of their meals.

Similarly, for computer menus, the categories should be comprehensible and distinctive so that the users are confident in making their selections. Users should have a clear idea of what will happen when they make a selection. Computer menus are more difficult to design than are restaurant menus, because computer displays typically have less space than do printed menus; display space is a scarce resource. In addition, the number of choices and the complexity is greater in many computer applications, and computer users may not have helpful waiters to turn to for an explanation.

The importance of meaningful organization of menu items was demonstrated in a study with 48 novice users (Liebelt et al., 1982). Simple menu trees with three levels and 16 target items were constructed in both meaningfully organized and disorganized forms. Error rates were nearly halved and user think time (time from menu presentation to user's selection of an item) was reduced for the meaningfully organized form. In a later study, semantically meaningful categories—such as food, animals, minerals, and cities—led to shorter response times than did random or alphabetic organizations (McDonald et al., 1983). This experiment tested 109 novice users who worked through 10 blocks of 26 trials. The authors conclude that "these results demonstrate the superiority of a categorical menu organization over a pure alphabetical organization, particularly when there is some uncertainty about the terms." With larger menu structures, the effect is even more dramatic, as has been demonstrated by studies with extensive videotext databases (Lee and Latremouille, 1980; McEwen, 1981; Perlman, 1984).

These results and the SSOA model suggest that the key to menu-structure design is first to consider the semantic organization that results from the task. The task terms and structure come first; number of items on the display becomes a secondary issue.

Menu-selection applications range from trivial choices between two items to complex information systems with 300,000 displays. The simplest applications consist of a single menu, but even with this limitation, there are many variations (Figure 3.1). The second group of applications includes a linear sequence of menu selections; the progression of menus is independent of the user's choice. Strict tree structures make up the third and most common group. Acyclic (menus are reachable by more than one path) and cyclic (structures with meaningful paths that allow users to repeat menus) networks constitute the fourth group. These groupings describe the semantic organization; special traversal commands may enable users to jump around the branches of a tree, to go back to the previous menu, or to go to the beginning of a linear sequence.

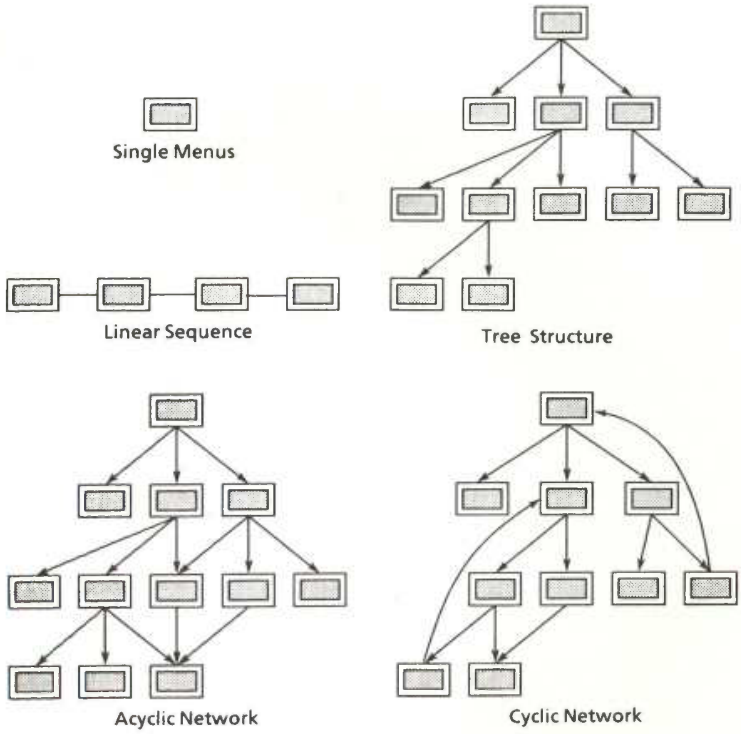


Figure 3.1

Menu systems can use simple single or linear sequences of menus. Tree-structured menus are the most common structure. Traversing deep trees or more elaborate acylic or cyclic menu structures can become difficult for some users.

3.2.1 Single menus

In some situations, a single menu is sufficient to accomplish a task. Single menus may have two or more items, may require two or more screens, or may allow multiple selections. Single menus may pop up on the current work area or may be permanently available (in a separate window or on a data tablet) while the main display is changed. Different guidelines apply for each situation.

Binary menus The simplest case is a binary menu with yes–no or true–false choices, such as is found in many home computer games. An example is `DO YOU WANT INSTRUCTIONS (Y,N)?` Even this simple example can be

improved. First, novice users might not understand the (Y,N) prompt—which is really an abbreviated form of the menu of choices. Second, this common query leaves the user without a clear sense of what will happen next. Typing Y might produce many pages of instructions, and the user might not know how to stop a lengthy output. Typing N is also anxiety producing, because users have no idea of what the program will do. Even simple menus should offer clear and specific choices that are predicable and thus give the user the sense of control:

```
Your choices are

   1 -- Get 12 lines of brief instructions.
   2 -- Get 89 lines of complete instructions.
   3 -- Go on to playing the game.

Type 1, 2, or 3, and press RETURN:
```

Since this version has three items, it is no longer a binary menu. It offers more specific items, so the user knows what to expect, but it still has the problem that users must take instructions now or never. Another strategy might be this:

```
At any time, you may type

      ? -- Get 12 lines of brief instructions.
     ?? -- Get 89 lines of complete instructions.

Be sure to press RETURN after every command
Ready for game playing commands:
```

This example calls attention to the sometimes narrow distinction between commands and menu selection; the menu choices have become more command-like since the user must now recall the ? or ?? syntax.

Menu items can be identified by single-letter mnemonics, as in this photo-library retrieval system:

```
Photos are indexed by film type
       B Black and white
       C Color
Type the letter of your choice
   and press RETURN:
```

The mnemonic letters in this menu are often preferred to numbered choices (see Section 3.7). The mnemonic-letter approach requires additional caution in avoiding collisions and increases the effort of translation to foreign languages, but its clarity and memorability are an advantage in many applications.

In graphic user interfaces, the user can point to selection buttons with a mouse or other cursor-control device. Choosing the orientation for output can be done with a pair of icons. The selected item is the darker one (inverted). In the following example, choosing between `Cancel` and `OK` can be by mouse click, but the thickened border on `OK` could indicate that this selection is the default, and that a RETURN keypress will select it.

These simple examples demonstrate alternative ways to identify menu items and to convey instructions to the user. No optimal format for menus has emerged, but consistency across menus in a system is extremely important.

Multiple-item menus and radio buttons Single menus may have more than two items. One example is an online quiz displayed on a touchscreen:

```
Who invented the telephone?
   Thomas Edison
   Alexander Graham Bell
   Lee De Forest
   George Westinghouse
Touch your answer.
```

Another example is a list of options in a document-processing system:

```
EXAMINE, PRINT, DROP, OR HOLD?
```

The quiz example has distinct, comprehensible items, but the document-processing example shows an implied menu selection that could be confusing to novice users. There are no explicit instructions, and it is not apparent that single-letter abbreviations are acceptable. Knowledgeable and expert users may prefer this short form of a menu selection, usually called a *prompt*, because of its speed and simplicity.

In graphic user interfaces, such selections are often called *radio buttons* since only one item can be selected at a time. This choice of paper size for printing shows `US Letter` as the selected item:

```
SUPERDUPERWRITER MAIN MENU                    PAGE 1

 1  Edit
 2  Copy
 3  CReate
 4  Delete
 5  Print
 6  View index

Type the number/letter
 or M for more choices.
 Then Press RETURN
```

```
SUPERDUPERWRITER MAIN MENU                    PAGE 2

 7  Alter line width
 8  New character set
 9  Search
10  Set passWord
11  Set cursor Blink rate
12  Set beep Volume

Type the number/letter
   or P to go back to Page 1.
 Then Press RETURN
```

Figure 3.2

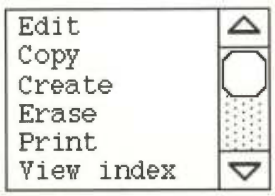
This text-oriented extended menu operates in a traditional keyboard style with numeric selection and with mnemonic letters.

Extended menus Sometimes, the list of menu items may require more than one display but allow only one meaningful item to be chosen. One solution is to create a tree-structured menu, but sometimes the desire to limit the system to one conceptual menu is strong. The first portion of the menu is displayed with an additional menu item that leads to the next display in the extended menu sequence. A typical application is in word-processing systems, where common choices are displayed first, but infrequent or advanced features are kept on the second display (Figure 3.2). Sometimes, the extended screen menu will continue for many screens. Extended menus provide a justification for the more elaborate scrolling capabilities found in most graphical user interfaces (Figure 3.3).

Pull-down and pop-up menus *Pull-down menus* are constantly available to the user via selections along a top menu bar; *pop-up menus* appear on the display in response to a click with a pointing device such as a mouse. The Xerox Star, Apple Lisa, and Apple Macintosh (Figure 3.4) made these possibilities widely available, and their versions have become standardized (Windows 3.0, IBM CUA SAA, OSF/Motif). Common items in the menu bar are `File`, `Edit`, `Search`, `Font`, `Format`, `View`, and `Help`. The user can make a selection by moving the pointing device over the menu items, which respond by

Figure 3.3

This graphical extended menu uses pointing and clicking to select an item, and scrolling to move among the items. Contrast to Figure 3.2.



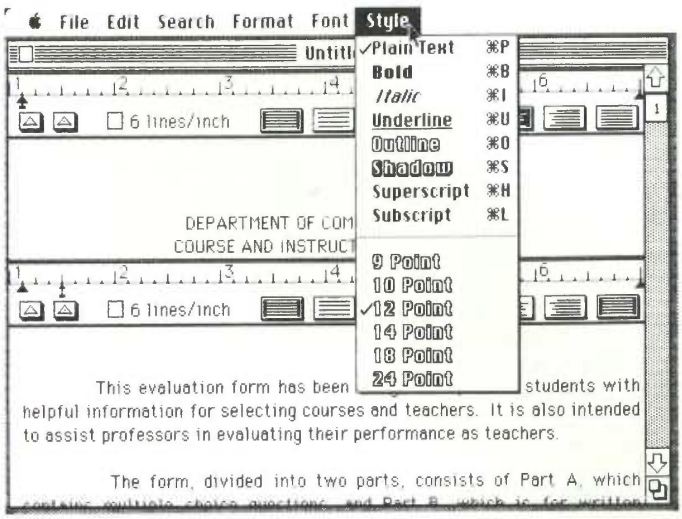


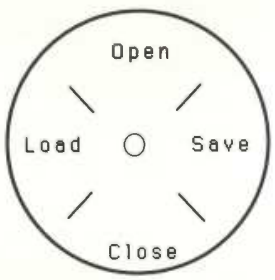
Figure 3.4

The pull-down menu on an early Apple Macintosh MacWrite program enabled users to select font variations and size. (Photo courtesy of Apple Computer, Inc.)

highlighting (reverse video, a box surrounding the item, and color all have been used). This Macintosh menu bar shows the available pull-down menus:

File Edit Font Size Style Format Spelling View

The contents of the pop-up menu may depend on where the cursor is when the pointing device is clicked. Since the pop-up menu covers a portion of the display, there is strong motivation to keeping the menu text small. Hierarchical sequences of pop-up menus are also used. Pop-up menus can also be organized in a circle to form *pie menus* (Callahan et al., 1988):



The pie menus are convenient because selection is more rapid and, with practice, can be done without visual attention.

Permanent menus Single menus can be used for permanently available commands that can be applied to a displayed object. For example, the Bank Street Writer, a word processor designed for children, always shows a fragment of the text and this menu:

```
ERASE     MOVE       FIND      TRANSFER
UNERASE   MOVEBACK   REPLACE   MENU
```

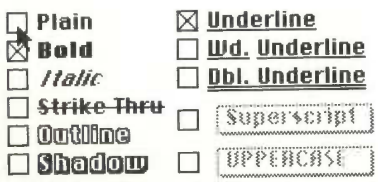
Moving the left and right arrow keys causes items to be highlighted sequentially in reverse video. When the desired command is highlighted, pressing the RETURN key initiates the action. Similarly, PRODIGY provides this menu on the bottom of the display:

```
NEXT BACK MENU PATH JUMP ACTION HELP EXIT
```

Only the permissible items are shown at any moment, however.

Other applications of permanent menus include paint programs, such as PC Paintbrush (Color Plate 1), PenPlay II (Figure 3.5 and Color Plate 2), computer-assisted design packages, or other graphics systems that display an elaborate menu of commands to the side of the object being manipulated. Mice, touchscreens, or other cursor-action devices allow users to make selections without keyboards.

Multiple-selection menus or check boxes A further variation on single menus is the capacity to make multiple selections from the choices offered. For example, a political-interest survey might allow multiple choice on one display (Figure 3.6). A multiple-selection menu with mouse clicks as the selection method is a convenient strategy for handling multiple binary choices, since the user is able to scan the full list of items while deciding. In the following Macintosh example, `Bold` and `Underline` have been selected; `Superscript` and `UPPERCASE` (grayed out) become available as a pop-up menu after the check box is selected:



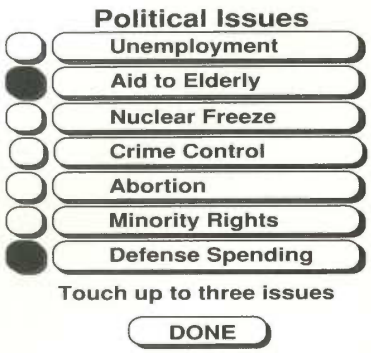
Summary Even the case of single menus provides a rich domain for designers and human-factors researchers. Questions of wording, screen

Figure 3.5

PenPlay II display shows menus of tool icons on the left and palette at bottom. Free-hand drawing on a touchscreen surface with a finger is accompanied by sounds. Gestures create varied patterns, including raindrops, snowflakes, needles, and colored circles, all of which can be recorded and replayed. (Implemented by Andrew Sears. Available from the University of Maryland Office of Technology Liaison, College Park, MD.)

Figure 3.6

This multiple-selection touchscreen menu enables users to make up to three selections of political issues.



layout, and selection mechanism all arise even in the simple case of choosing from one set of items. Still more challenging questions emerge during design of sequences and trees of menus.

3.2.2 Linear sequences and multiple menus

Often, a series of interdependent menus can be used to guide users through a series of choices in which they see the same sequence of menus no matter what choices they make. For example, a document-printing package might have a linear sequence of menus to choose print parameters, such as device, line spacing, and page numbering (Figure 3.7). Another familiar example is an online examination that has a sequence of multiple-choice test items, each made up as a menu.

With high-resolution screens and pointing devices, it is possible to include several menus on a single display, thereby simplifying the user interface and speeding usage (Figure 3.8).

Movement through the menus Linear sequences guide the user through a complex decision-making process by presenting one decision at a time. The document-printing example could be improved by offering the user several menus on the screen at once. If the menus do not fit on one screen, then there should be a mechanism for going back to previous screens to review or change choices made earlier. A second improvement is to display previous

```
Do you want the document printed at
  T - your Terminal
  P - the computer center line Printer
  L - the computer center Laser printer
Type your choice and press RETURN:

Do you want
  1 - single spacing
  2 - double spacing
  3 - triple spacing
Type your choice and press RETURN:

Do you want
  N - No page numbering
  T - page numbering on the Top, right justified
  B - page numbering at the Bottom, centered
Type your choice and press RETURN:
```

Figure 3.7

A linear sequence of menus allows the user to select three print parameters for a document: printing device, line spacing, and page numbering.

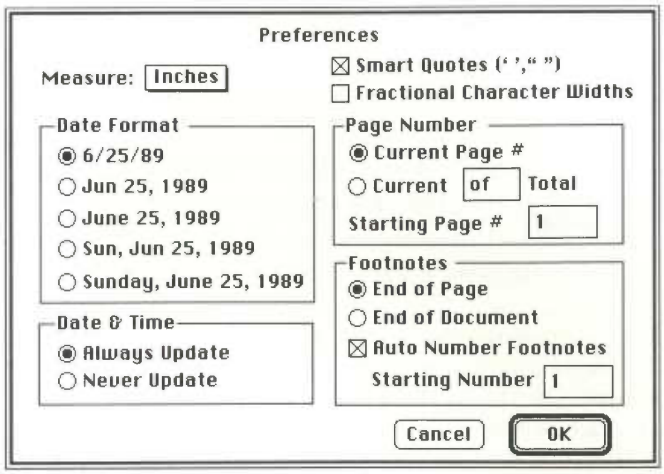


Figure 3.8

A set of independent menus collected on a single dialog box within Claris's MacWrite II. (Courtesy of Claris Corp., Santa Clara, CA.)

choices, so users can see what decisions they have made. A third improvement is to let the users know how many and which menus they have not yet seen.

Summary Linear sequences and multiple menus provide simple and effective means for guiding the user through and structuring a decision-making process. The user should be given a clear sense of progress or position within the menus, and the means for going backward to earlier choices (and possibly terminating or restarting the sequence).

Choosing the order and layout of menus in a linear sequence is often straightforward, but care must be taken to match user expectations. One strategy is to place the easy decisions first (or in the upper left in a multiple menu), to relieve users of simple concerns quickly, enabling them to concentrate on more difficult choices.

3.2.3 Tree-structured menus

When a collection of items grows and becomes difficult to maintain under intellectual control, people form categories of similar items, creating a tree structure (Clauer, 1972; Brown, 1982; Norman, 1991). Some collections can be partitioned easily into mutually exclusive groups with distinctive identifiers.

Familiar examples include these groupings:

- Male, female
- Animal, vegetable, mineral
- Spring, summer, autumn, winter
- Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday
- Less than 10, between 10 and 25, greater than 25
- Percussion, string, woodwind, brass
- Fonts, size, style, spacing

Even these groupings may occasionally lead to confusion or disagreement. Classification and indexing are complex tasks, and, in many situations, there is no single solution that is acceptable to everyone. The initial design can be improved as a function of feedback from users. Over time, as the structure is improved and as users gain familiarity with it, success rates will improve.

In spite of the associated problems, tree-structured menu systems have the power to make large collections of data available to novice or intermittent users. If each menu has eight items, then a menu tree with four levels has the capacity to lead an untrained user to the correct frame out of a collection of 4096 frames.

If the groupings at each level are natural and comprehensible to users, and if users know for what they are looking, then the menu traversal can be accomplished in a few seconds—more quickly than flipping through a book. On the other hand, if the groupings are unfamiliar and users have only a vague notion for what they are looking, they may get lost in the tree menus for hours (Robertson et al., 1981; Norman and Chin, 1988).

Terminology from the user's task domain can orient the user. Instead of using a title, such as `MAIN MENU OPTIONS`, that is vague and emphasizes the computer domain, use terms such as `FRIENDLIBANK SERVICES` or simply `GAMES`.

Depth versus breadth The *depth*, or number of levels, of a menu tree depends, in part, on the *breadth*, or number of items per level. If more items are put into the main menu, then the tree spreads out and has fewer levels. This shape may be advantageous, but only if clarity is not compromised substantially and if a slow display rate does not consume the user's patience. Several authors have urged using four to eight items per menu, but, at the same time, they urge using no more than three to four levels. With large menu applications, one or both of these guidelines must be compromised.

Several empirical studies have dealt with the depth-breadth trade-off and the evidence is quite strong that breadth should be preferred over depth. In fact, there is reason to encourage menu trees to be limited to three levels;

when the depth goes to four or five there appears to be a greater chance of users becoming lost or disoriented.

Kiger (1984) grouped 64 items in these menu-tree forms:

8 x 2	Eight items on each of two levels
4 x 3	Four items on each of three levels
2 x 6	Two items on each of six levels
4 x 1 + 16 x 1	A four-item menu followed by a 16-item menu
16 x 1 + 4 x 1	A 16-item menu followed by a four-item menu

The deep narrow tree, 2 x 6, produced the slowest, least accurate, and least preferred version; the 8 x 2 was among those rated highest for speed, accuracy, and preference. The 22 subjects performed 16 searches on each of the five versions.

Landauer and Nachbar (1985) confirmed the advantage of breadth over depth and developed predictive equations for traversal times. They varied the number of items per level from 2, 4, 8, to 16 to reach 4096 target items of numbers or words (Figure 3.9). The times for the task with words ranged from 23.4 seconds down to 12.5 seconds as the breadth increased and the

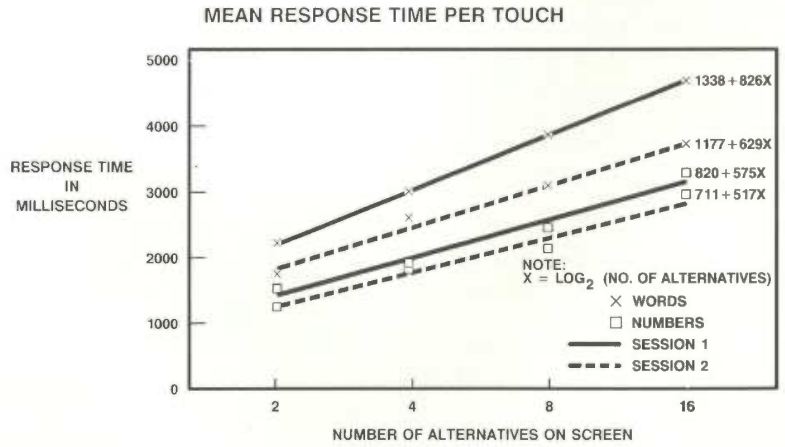


Figure 3.9

The advantage of broader shallower trees was demonstrated in a study with menus with branching factors of 2, 4, 8, and 16 items. (Adapted with courtesy of Bellcore, Morristown, NJ.)