Mark Kostabi, *Message Center (Gold Rush)*, 1996
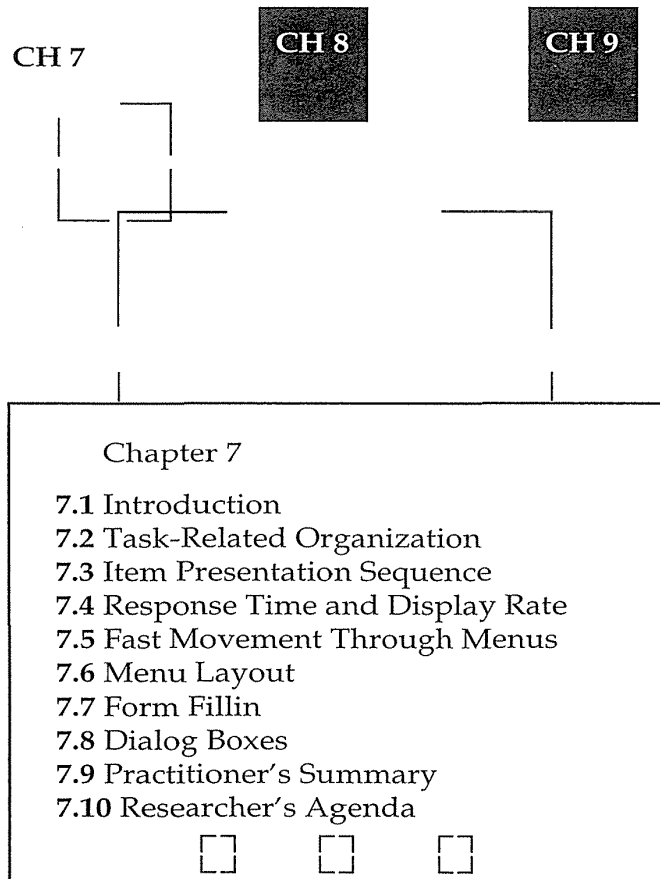
Menu Selection, Form Fillin, and Dialog Boxes

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

W. H. Auden, *A Certain World*

CH 7   CH 8   CH 9

Chapter 7

7.1 Introduction
7.2 Task-Related Organization
7.3 Item Presentation Sequence
7.4 Response Time and Display Rate
7.5 Fast Movement Through Menus
7.6 Menu Layout
7.7 Form Fillin
7.8 Dialog Boxes
7.9 Practitioner's Summary
7.10 Researcher's Agenda

7.1 Introduction

When designers cannot create appropriate direct-manipulation strategies, menu selection and form fillin are attractive alternatives. Whereas early systems used full-screen menus with numbered items, modern menus are usually pulldowns, check boxes or radio buttons in dialog boxes, or embedded links on World Wide Web pages, all selectable by mouse clicks. When the menu items are written with familiar terminology and are organized in a convenient structure and sequence, users can select an item easily.

Menus are effective because they offer the cues to elicit user recognition, rather than forcing the user to recall the syntax of a command from memory. Users indicate their choices with a pointing device or keystroke and get feedback indicating what they have done. Simple menu selection is especially effective when users have little training, use the system intermittently, are

unfamiliar with the terminology, or need help in structuring their decision-making process. With careful design of complex menus and high-speed interaction, menu selection can become appealing even to expert frequent users.

However, just because a designer uses menu selection, form fillin, and dialog boxes, there is no guarantee that the interface will be appealing and easy to use. Effective interfaces emerge only after careful consideration of and testing for numerous design issues, such as task-related organization, phrasing of items, sequence of items, graphic layout and design, response time, shortcuts for knowledgeable frequent users, on line help, error correction, and selection mechanisms (keyboard, pointing devices, touchscreen, voice, and so on) (Norman, 1991).

This chapter starts with menus, then moves on to cover form fillin and this method's integration into dialog boxes. The examples are drawn from pull-down menus, full-screen displays, embedded links of the World Wide Web, and graphical dialog boxes. Menu items can be textual, graphic, or auditory.

7.2   Task-Related Organization

The primary goal for menu, form-fillin, and dialog-box designers is to create a sensible, comprehensible, memorable, and convenient organization relevant to the user's tasks. We can learn a few lessons by following the decomposition of a book into chapters, a program into modules, or the animal kingdom into species. 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 only one category, and designers are tempted to create duplicate pointers, thus forming a network.

Restaurant menus separate appetizers, soups, salads, main dishes, desserts, and beverages to help customers organize their selections. Menu items should fit logically into categories and have readily understood meanings. Restaurateurs 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 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. 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 (Norman and Chin, 1989).

The importance of meaningful organization of menu items was demonstrated in an early 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, 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 concluded 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.

These results and the OAI model suggest that the key to menu-structure design is first to consider the task-related objects and actions. For a music-concert ticketing system, the menus might separate out types of music (classical, folk, rock, jazz, and so on), concert locations, or dates, and might offer actions such as browsing lists, searching by performer name, or locating inexpensive performances. The interface objects might be dialog boxes with check boxes for types of music and scrolling menus of concert locations. Performer names might be in a scrolling list or typed in via form fillin.

Menu-selection applications range from trivial choices between two items to complex information systems that offer thousands of displays. The simplest applications consist of a single menu, but even within this limited format, there are many variations (Fig. 7.1). 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 that are reachable by more than one path) and cyclic (structures with meaningful paths that allow users to repeat menus) networks constitute the fourth group. In addition, 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.

7.2.1 Single menus

In some situations, a single menu is sufficient to accomplish a task. Single menus may have two or more items, or may allow multiple selections. Single menus may pop up on the current work area or may be permanently available (on a frame, in a separate window, or on a data tablet) while the main display is changed.
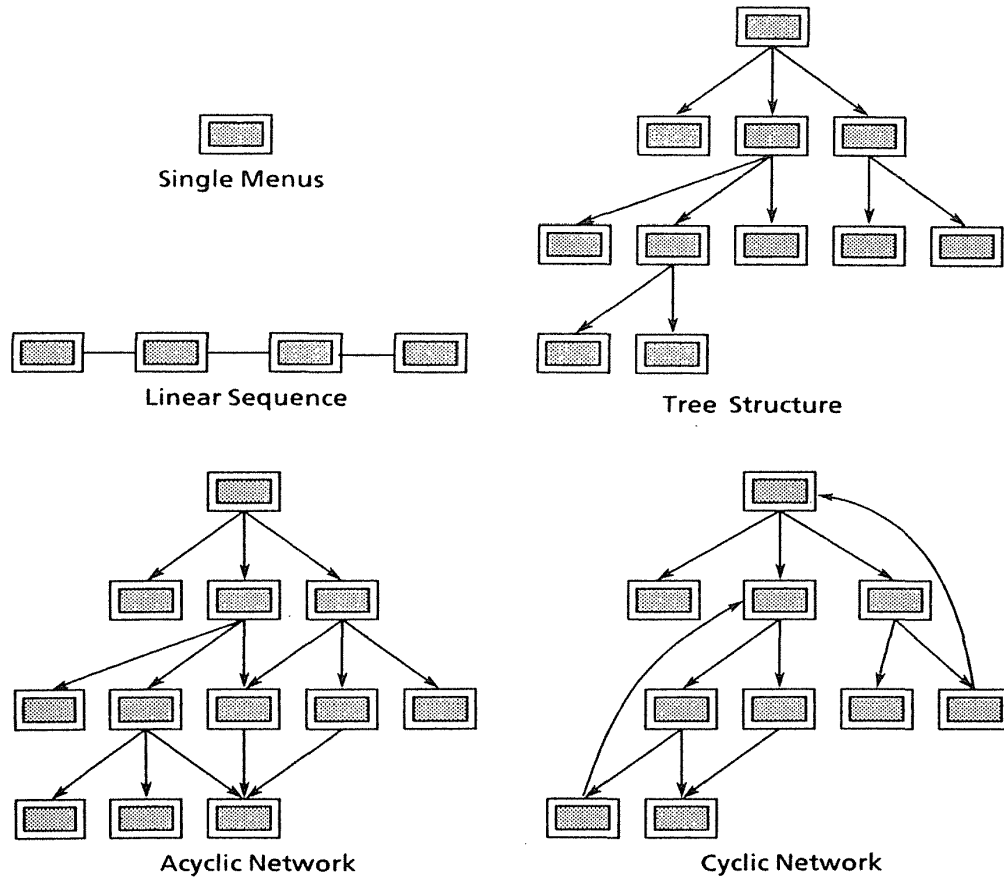
Figure 7.1

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

Binary menus The simplest case is a *binary menu* with, for example, yes-no, true-false, or male–female choices. In keyboard-oriented systems, menu items can be identified by single-letter mnemonics, as they are 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:
```

Users often prefer *mnemonic letters,* such as those in this menu, to numbered choices (see Section 7.5). 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 GUIs, dialog boxes offer users selection buttons, often called *radio buttons.* Selection is made with a mouse or other cursor-control device. This box has two radio buttons.
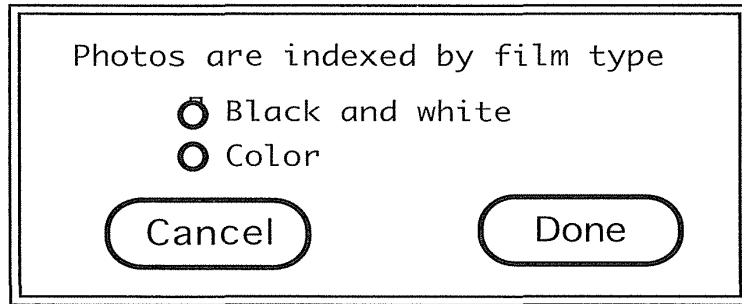
While earlier systems used text only, modern systems can show items graphically. For example, users can choose the orientation for output by selecting one of a pair of icons. The selected item is the darker (inverse highlighted) one.

In the following example, users can choose between Cancel and OK by a mouse click, but the thickened border on OK indicates that this selection is the default, and that pressing RETURN 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 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 GUIs, radio buttons support single item selection from a multiple-item menu. This choice of paper size for printing shows US Letter as the selected item:

Paper: ⊙ **US Letter**   ○ **A4 Letter**
　　　　　　 ○ **US Legal**   ○ **B5 Letter**
　　　　　　 ○ **No. 10 Envelope**

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 (Fig. 7.2). 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;
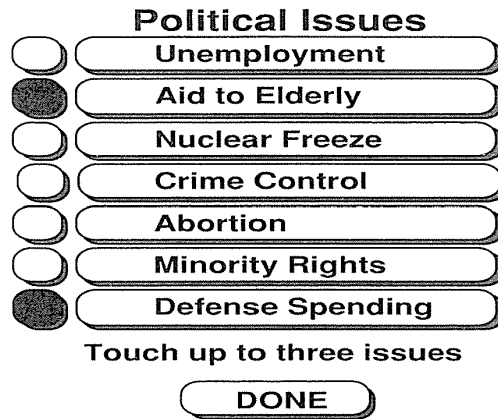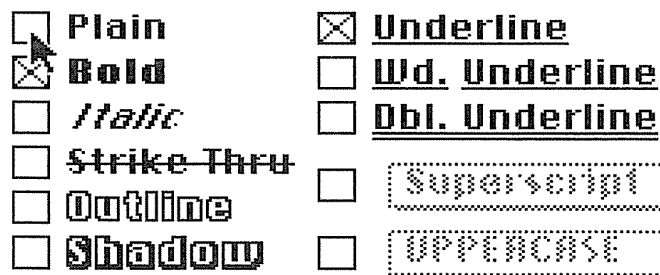
Political Issues

◯	Unemployment
⬤	Aid to Elderly
◯	Nuclear Freeze
◯	Crime Control
◯	Abortion
◯	Minority Rights
⬤	Defense Spending

Touch up to three issues

(DONE)

Figure 7.2

A multiple-selection touchscreen menu. Users can select up to three political issues.

Superscript and UPPERCASE (grayed out) become available on a pop-up menu after the check box is selected:

☐	Plain	☒	Underline
☒	Bold	☐	Wd. Underline
☐	*Italic*	☐	Dbl. Underline
☐	Strike Thru	☐	Superscript
☐	Outline		
☐	Shadow	☐	UPPERCASE

Pull-down and pop-up menus *Pull-down menus* are constantly available to the user via selections along a top menu bar. The Xerox Star, Apple Lisa, and Apple Macintosh (Fig. 7.3) made these possibilities widely available, and their versions have become standardized (Windows, IBM OS/2, OSF/Motif). Common items in the menu bar are File, Edit, Font, Format, View, Window, and Help. The users make a selection by moving the pointing device over the
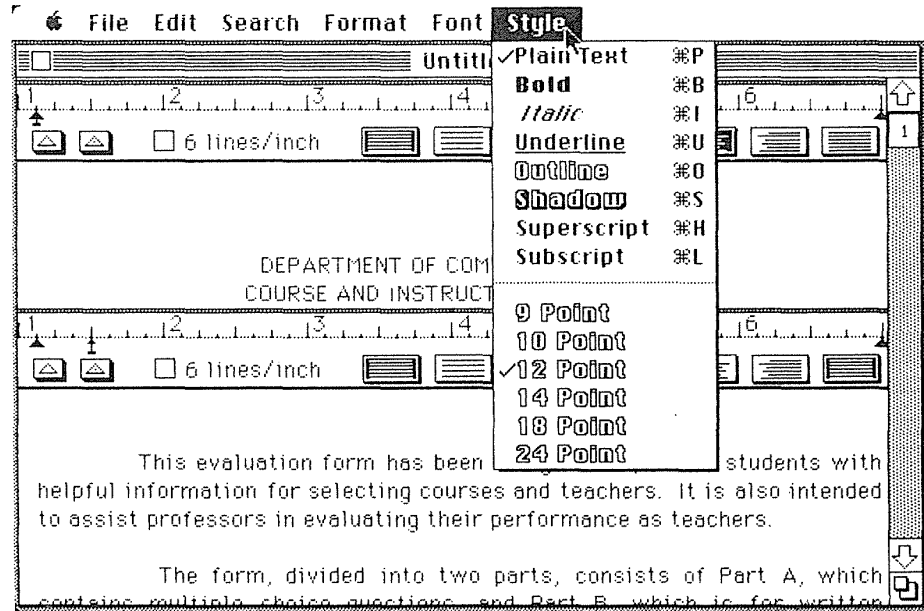
Figure 7.3

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

menu items, which respond by highlighting (reverse video, a box surrounding the item, and color all have been used). Since positional constancy is such a strong principle, when an item is not available for selection it is preferable to gray it out rather than to remove it from the list. This Macintosh menu bar shows the available pull-down menus:

⚫ File   Edit   Font   Size   Style   Format   Spelling   View

In Windows, pull-down menu items are also selectable with a keystroke sequence.

Pop-up menus appear on the display in response to a click with a pointing device such as a mouse. 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 keep 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):
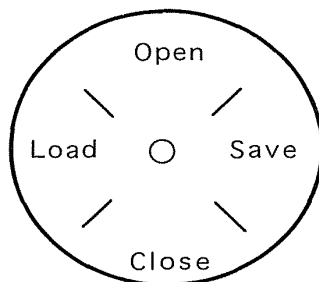
Pie menus are convenient because selection is more rapid and, with practice, can be done without visual attention. Improvements to appearance and behavior were made in a pie menus variant called marking menus in Alias StudioPaint V3 (Tapia and Kurtenbach, 1995).

Scrolling and two-dimensional menus (fast and vast)   Sometimes the list of menu items may be longer than the 20 to 60 lines that can reasonably fit on a display. One solution is to create a tree-structured menu, but sometimes the desire to limit the system to one conceptual menu is strong. A typical application is selecting a state from the 50 states in the United States. The first portion of the menu is displayed with an additional menu item that leads to the next display in the menu sequence. The scrolling (or paging) menu might continue with dozens or thousands of items using the list-box capabilities found in most GUIs. Alternatively, a multiple-column menu might be used, with the 50 states arranged in five columns of 10 items each (Fig. 7.4). These "fast and vast" *two-dimensional menus* give users an excellent overview of the choices, reduce the number of actions, and allow rapid selection. Multiple-column menus are especially useful in World Wide Web page design to minimize the scrolling needed to see a long list and to give users a single-screen overview of the full set of choices.

Alphasliders   When the menu items become too numerous to show on the screen at once without obscuring other items, more compact strategies are needed. One approach is the *alphaslider,* which uses multiple levels of granularity in moving the slider thumb (scroll box) and therefore can support tens or hundreds of thousands of items (Ahlberg and Shneiderman, 1994). The following alphaslider covers the 10,000 actors in a film database (Color Plate B4). The dark upper part of the thumb jumps over 40 actors for each move of

```
┌─────────────────────────────────────────────────────────────────┐
│  Select multiple states for travel information:                   │
│                                                                   │
│  Alabama      Hawaii     Massachusetts  New Mexico    South Dakota │
│  Alaska       Idaho      Michigan      ┌─────────┐   Tennessee    │
│  Arizona      Illinois   Minnesota     │New York │   Texas        │
│            ┌─────────┐                 └─────────┘                │
│  Arkansas  │Indiana  │  Mississippi    North Carolina  Utah       │
│  California └─────────┘  Missouri       North Dakota   Vermont     │
│            Iowa                        Ohio                        │
│  Colorado   Kansas     Montana         Oklahoma       Virginia    │
│  Connecticut Kentucky  Nebraska        Oregon         Washington  │
│  Delaware   Louisiana  Nevada          Pennsylvania   West Virginia│
│  Florida    Maine      New Hampshire   Rhode Island   Wisconsin   │
│            ┌─────────┐ ┌──────────┐                               │
│  Georgia   │Maryland │ │New Jersey│    South Carolina  Wyoming     │
│            └─────────┘ └──────────┘                               │
│                                   (  Ok  )     ( Cancel )         │
└─────────────────────────────────────────────────────────────────┘
```
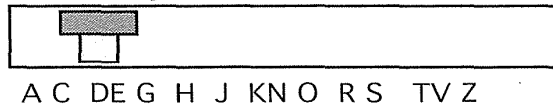
Figure 7.4

A "fast and vast" two-dimensional menu that allows rapid multiple selection from the list of 50 states. This version shows a menu with five columns of 10 states each, arranged in alphabetical order down the columns.

the mouse, and the lighter smaller lower part allows movement through each actor's name:

Actor: Connery, Sean

```
┌──────────────────────────────────────────┐
│  ▓▓▓▓                                      │
│   ┌┐                                       │
│   └┘                                       │
└──────────────────────────────────────────┘
```

A C DE G H J KN O R S   TV Z

The index at the bottom of the alphaslider gives users an idea of where to jump to start a new search.

Embedded links All the menus discussed thus far might be characterized as *explicit menus* in that there is an orderly enumeration of the menu items with little extraneous information. In many situations, however, the menu items might be *embedded* in text or graphics and still be selectable. This is the basis for hypertext designs (see Chapter 16).

In a textual database with articles about people, events, and places for a museum application, it is natural to allow users to retrieve detailed information by selecting a name in context (Koved and Shneiderman, 1986). Selectable names are highlighted, and users click with a mouse (Fig. 7.5). The names, places, or phrases are menu items embedded in meaningful text that

```
WASHINGTON, DC: THE NATION'S CAPITAL          PAGE 2 OF 3

   Located between Maryland and Virginia, Washington, DC
embraces the White House and the Capitol, a host of
government offices as well as the Smithsonian museums.
Designed by Pierre L'Enfant, Washington, DC is a graceful
city of broad boulevards, national monuments, the rustic
Rock Creek Park, and the National Zoo.
   First-time visitors should begin at the mall by walking
from the Capitol towards the Smithsonian museums and on
```

BACK PAGE   NEXT PAGE   RETURN TO "NEW YORK CITY"   EXTRA

Figure 7.5

Embedded links in an early version of Hyperties. The links improve comprehensibility over numbered menu lists and lowered anxiety for novice users. A reverse-video selector box initially covers the NEXT PAGE action. Users move the selector box over highlighted links or actions, then select by pressing RETURN. A mouse allows them to make a selection by merely clicking on the highlighted link or action. (Created in 1983 by Human–Computer Interaction Laboratory, University of Maryland, College Park, MD; distributed and refined by Cognetics Corporation, Princeton Junction, NJ.)

informs users and helps to clarify the meaning of the items. Embedded links were popularized in the Hyperties system (Color Plate C1) (Cognetics Corp., Princeton Junction, NJ), which was used for two early commercial hypertext projects (Shneiderman, 1988; Shneiderman and Kearsley, 1989), and became the preferred method for traversing links on the World Wide Web (see Color Plates A2 to A5 and Figs. 16.5 to 16.10).

Embedded links have emerged in other applications. Air-traffic–control systems allow users to select airplanes in the spatial layout of flight paths to obtain more detailed information. Many geographic-information systems similarly allow users to select cities or other features to obtain more information. Selection of regions in a two-dimensional layout, usually called *image maps*, was built into Hyperties in 1988 and has become popular on websites. Embedded links permit items to be viewed in context and eliminate the need for a distracting and screen-wasting enumeration of items. Contextual display helps to keep the users focused on their tasks and on the objects of interest.

Iconic menus, toolbars, or palettes Menus can offer many actions that a user can select with a click and apply to a displayed object. These menus, often called *toolbars* or *palettes,* are widely used in paint and draw programs (see Fig. 6.7), in computer-assisted design packages, and in other graphics

systems. Users may be able to customize the toolbar with their choices of items, and to control the placement to be at the top or side. Users who wish to conserve screen space for their documents can eliminate the toolbar.

7.2.2 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 a sequence of menus. 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. Another familiar example is an online examination that has a sequence of multiple-choice test items, each made up as a menu. Guidance for users in making complex decisions can often be provided by a sequence of cue cards or Wizards (a Microsoft term).

Linear sequences guide the user through a complex decision-making process by presenting one decision at a time. We could improve the document-printing example by offering the user several menus on the screen at once. Putting several menus on a single dialog box simplifies the user interface, allows users to enter choices in any order, and speeds usage (Fig. 7.6).

7.2.3 Tree-structured menus

When a collection of items grows and becomes difficult to maintain under intellectual control, designers can form categories of similar items, creating a *tree structure* (Clauer, 1972; 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, for example, colors or flowers. 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.
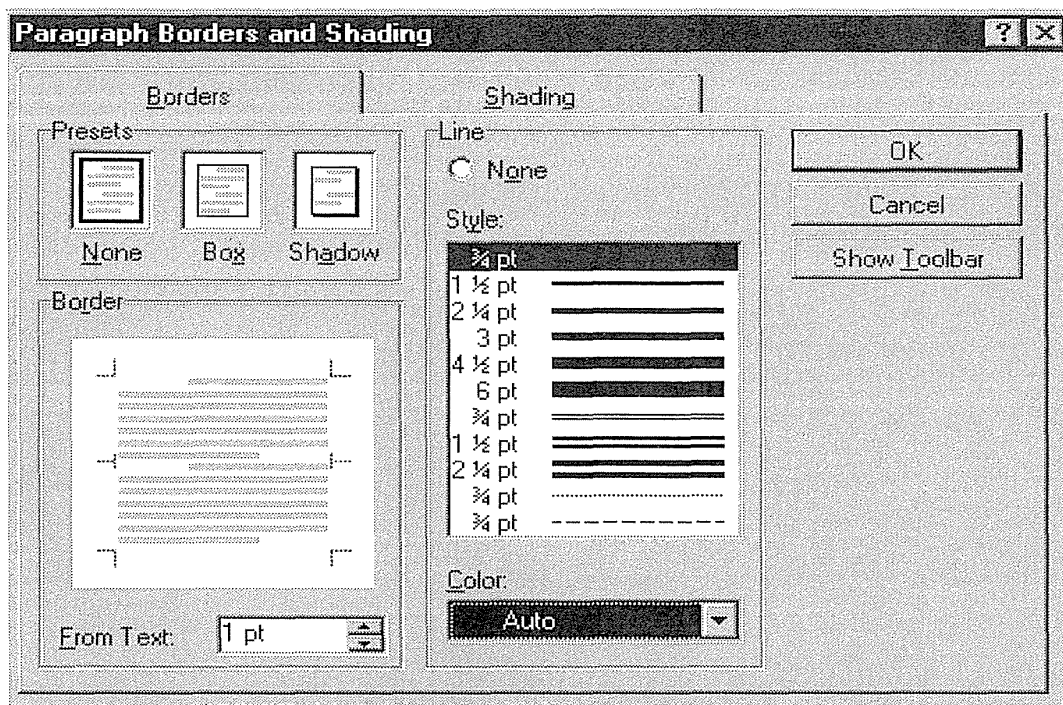
Figure 7.6

Multiple menus in a single dialog box. Users can enter choices in any order and are given a clear overview of the possibilities. (Used with permission of Microsoft Corp., Redmond, WA.)

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 30 items, then a menu tree with four levels has the capacity to lead an untrained user through a collection of 810,000 destinations. That number would be excessively large for a set of commands in a word processor, but would be realistic in a World Wide Web application such as a directory (see Fig. 16.5), a digital library (see Color Plate A5), or an online service such as America Online (see Color Plate A6).

If the groupings at each level are natural and comprehensible to users, and if users know the target, then menu traversal can be accomplished in a few seconds—it is faster than flipping through a book. On the other hand, if the groupings are unfamiliar and users have only a vague notion of the item that they seek, they may get lost for hours in the tree menus (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 urge 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 tradeoff, and the evidence is strong that breadth should be preferred over depth. In fact, there is reason to encourage designers to limit menu trees to three levels: when the depth goes to four or five, there is a good chance of users becoming lost or disoriented.

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

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

The deep narrow tree, 2×6, produced the slowest, least accurate, and least preferred version; the 8×2 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. The times for the task with words ranged from 23.4 seconds down to 12.5 seconds as the breadth increased and the number of levels decreased. Over the range studied, the authors suggest that a simple function of the number of items on the screen will predict the time, T, for a selection:

$$T = k + c^*\log b,$$

where k and c are empirically determined constants for scanning the screen to make a choice, and b is the breadth at each level. Then, the total time to traverse the menu tree depends on only the depth, D, which is

$$D = \log bN,$$

where N is the total number of items in the tree. With $N = 4096$ target items and a branching factor of $b = 16$, the depth $D = 3$, and the total time is $3*(k + c*\log 16)$.

Norman and Chin (1988) fixed the number of levels at four, with 256 target items, and varied the shape of the tree structure. They recommend greater breadth at the root and at the leaves, and add a further encouragement to minimize the total number of menu frames needed so as to increase familiarity. In an interesting variation, Wallace et al. (1987) confirmed that broader, shallower trees (4 $\times$ 3 versus 2 $\times$ 6) produced superior performance, and showed that, when users were stressed, they made 96 percent more errors and took 16 percent longer. The stressor was simply an instruction to work quickly ("It is imperative that you finish the task just as quickly as possible"); the control group received gentler verbal instruction to avoid rushing ("Take your time; there is no rush").

Even though the semantic structure of the items cannot be ignored, these studies suggest that the fewer the levels, the greater the ease of decision making. Of course, display rates, response time, and screen clutter must be considered, in addition to the semantic organization.

Task-related grouping in tree structures   Grouping menu items in a tree such that they are comprehensible to users and match the task structure is sometimes difficult. The problems are akin to putting kitchen utensils in order; steak knives go together and serving spoons go together, but where do you put butter knives or carving sets? Computer-menu problems include overlapping categories, extraneous items, conflicting classifications in the same menu, unfamiliar jargon, and generic terms. Based on this set of problems, here are several suggested rules for forming menu trees:

- *Create groups of logically similar items* For example, a comprehensible menu would list countries at level 1, states or provinces at level 2, and cities at level 3.
- *Form groups that cover all possibilities* For example, a menu with age ranges 0-9, 10-19, 20-29, and > 30 makes it easy for the user to select an item.
- *Make sure that items are nonoverlapping* Lower-level items should be naturally associated with a single higher-level item. Overlapping categories such as Entertainment and Events are a poor choice compared to Concerts and Sports.
- *Use familiar terminology, but ensure that items are distinct from one another* Generic terms such as Day and Night may be too vague, when compared to Before 6 P.M. and After 6 P.M.

Menu maps   As the depth of a menu tree grows, users find it increasingly difficult to maintain a sense of position in the tree; their sense of disorienta-
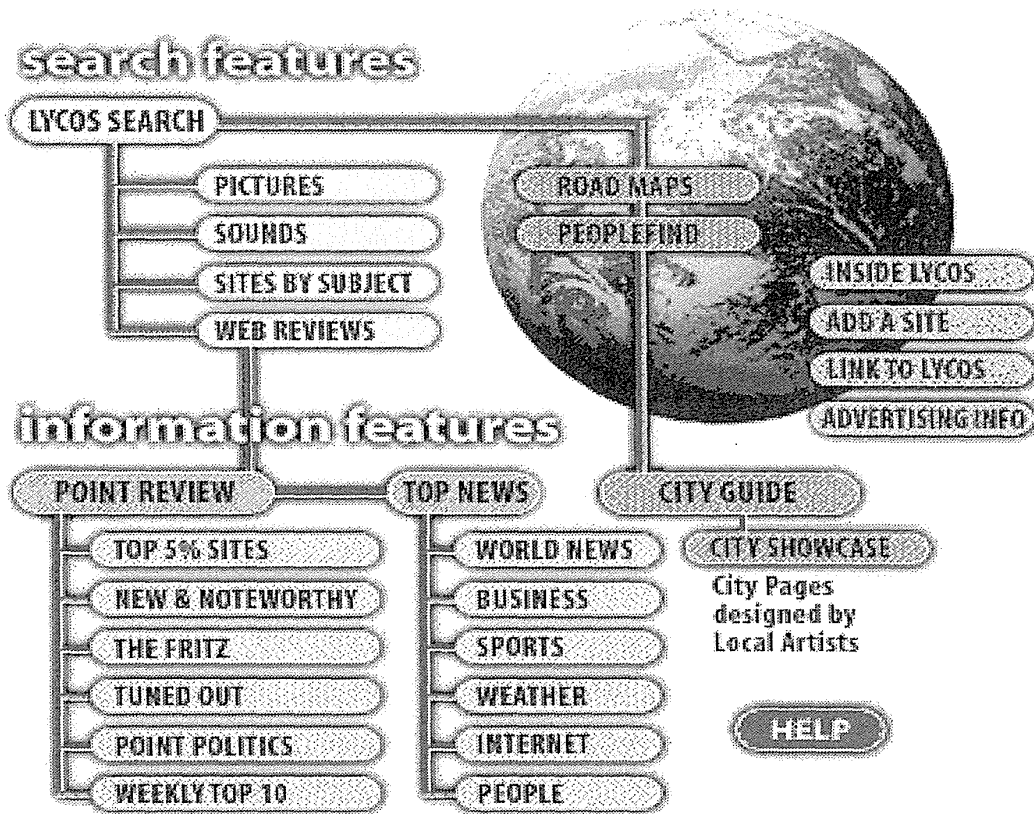
Figure 7.7

A menu map of a World Wide Web site. This example from the Lycos search service is called a sitemap.

tion, or of "getting lost," grows. Viewing one menu at a time is like seeing the world through a cardboard tube; it is hard to grasp the overall pattern and to see relationships among categories. Evidence from several early studies demonstrated the advantage of offering a spatial map to help users stay oriented. Sometimes *menu maps* are shown on web pages (Fig. 7.7); sometimes they are printed as large posters to give users a visual overview of hundreds of items at several levels. Another approach is to have the overview in the user manual as a fold-out or spread over several pages as a tree diagram or indented-text display to show levels.

Summary There is no perfect menu structure that matches every person's knowledge of the application domain. Designers must use good judgment for the initial implementation, but then must be receptive to suggested improvements and empirical data. Users will gradually gain familiarity,

even with extremely complex tree structures, and will be increasingly successful in locating required items.

7.2.4   Acyclic and cyclic menu networks

Although tree structures are appealing, sometimes network structures are more appropriate. For example, in a commercial online service, it might make sense to provide access to banking information from both the financial and consumer parts of a tree structure. A second motivation for using *menu networks* is that it may be desirable to permit paths between disparate sections of a tree, rather than requiring users to begin a new search from the main menu. Network structures in the form of *acyclic* or *cyclic graphs* arise naturally in social relationships, transportation routing, scientific-journal citations, and many other applications. As users move from trees to acyclic networks to cyclic networks, the potential for getting lost increases. Confusion and disorientation are often reported among World Wide Web users who have difficulty navigating that large cyclic network.

With a tree structure, the user can form a mental model of the structure and of the relationships among the menus. Developing this mental model may be more difficult with a network. With a tree structure, there is a single parent menu, so backward traversals toward the main menu are straightforward. In networks, a stack of visited menus must be kept to allow backward traversals. It may be helpful to preserve a notion of "level," or of distance from the main menu. Users may feel more comfortable if they have a sense of how far they are from the main menu.

7.3   Item Presentation Sequence

Once the items in a menu have been chosen, the designer is still confronted with the choice of *presentation sequence*. If the items have a natural sequence—such as days of the week, chapters in a book, or sizes of eggs—then the decision is trivial. Typical bases for sequencing items include these:

- *Time*   Chronological ordering
- *Numeric ordering*   Ascending or descending ordering
- *Physical properties*   Increasing or decreasing length, area, volume, temperature, weight, velocity, and so on

Many cases have no task-related ordering, and the designer must choose from such possibilities as these:

- *Alphabetic sequence of terms*
- *Grouping of related items* (with blank lines or other demarcation between groups)
- *Most frequently used items first*
- *Most important items first* (importance may be difficult to determine and may vary among users)

Card (1982) experimented with a single 18-item vertical permanent menu of text-editing commands such as INSERT, ITALIC, and CENTER. He presented subjects with a command, and they had to locate the command in the list, to move a mouse-controlled cursor, and to select the command by pressing a button on the mouse. The menu items were sequenced in one of three ways: alphabetically, in functional groups, and randomly. Each of four subjects made 86 trials with each sequencing strategy. The mean times were as follows

Strategy	Time per trial (seconds)
alphabetic	0.81
functional	1.28
random	3.23

Since subjects were given the target item, they did best when merely scanning to match the menu items in an alphabetic sequence. The performance with the functional groupings was remarkably good, indicating that subjects began to remember the groupings and could go directly to a group. In menu applications where the users must make a decision about the most suitable menu item, the functional arrangement might be more appealing. Users' memory for the functionally grouped items is likely to surpass their memory for the alphabetic or random sequences. The poor performance that Card observed with the random sequence confirms the importance of considering alternative presentation sequences for the items.

With a 64-item menu, the time for locating a target word was found to increase from just over 2 seconds for an alphabetic menu to more than 6 seconds for a random menu (McDonald et al., 1983). When the target word was replaced with a single-line definition, the 109 subjects could no longer scan for a simple match and had to consider each menu item carefully. The advantage of alphabetic ordering nearly vanished. User reaction time went up to about 7 seconds for the alphabetic and about 8 seconds for the random organization. Somberg and Picardi (1983) studied user reaction times in a five-item menu. Their three experiments revealed a significant and nearly linear

relationship between the user's reaction time and the serial position in the menu. Furthermore, there was a significant increase in reaction time if the target word was unfamiliar, rather than familiar.

If frequency of use is a potential guide to sequencing menu items, then it might make sense to vary the sequence adaptively to reflect the current pattern of use. Unfortunately, adaptations can be disruptive, increasing confusion and undermining the users' learning of menu structures. In addition, users might become anxious that other changes might occur at any moment. Evidence against the utility of such changes was found in a study in which a pull-down list of food items was resequenced to ensure that the most frequently selected items migrated toward the top (Mitchell and Shneiderman, 1988). Users were clearly unsettled by the changing menus, and their performance was better with static menus. In contrast, evidence in favor of adaptation was found in a study of a telephone-book menu tree that had been restructured to make frequently used telephone numbers more easily accessible (Greenberg, 1985). However, this study did not deal with the issue of potentially disorienting changes to the menu during usage. So that you avoid disruption and unpredictable behavior, it is probably a wise policy to allow users to specify when they want the menu restructured.

When some menu items are much more frequently selected than are others, there is a temptation to organize the menu in descending frequency. This organization does speed up selection of the topmost items, but the loss of a meaningful ordering for low-frequency items is disruptive. A sensible compromise is to extract three or four of the most frequently selected items and to put them on the top, while preserving the order for the remaining items. In controlled experiments and field studies with a lengthy font menu, the three popular fonts (Courier, Helvetica, and Times) were put on top, and the remaining list was left in alphabetical order. This split-menu strategy proved appealing and statistically significantly improved performance (Sears and Shneiderman, 1993). An improved theory of menu-selection performance emerged that showed that familiar items were selected in logarithmic time, whereas unfamiliar items were found in linear time with respect to their position in the menu. The software collected usage frequency, but the split-menu ordering remained stable until the system administrator decided to make a change.

7.4   Response Time and Display Rate

A critical variable that may determine the attractiveness of menu selection is the speed at which users can move through the menus. The two components of speed are system *response time*, the time it takes for the system to begin displaying information in response to a user selection, and *display rate*, the

speed at which the menus are displayed (see Chapter 10). For most modern computers, response time is so rapid that this issue is less of a concern, but delays on the World Wide Web have revived this topic.

Deep menu trees or complex traversals become annoying to the user if system response time is slow, resulting in long and multiple delays. With slow display rates, lengthy menus become annoying because of the volume of text or graphics that must be displayed. In positive terms, if the response time is long, then designers should place more items on each menu to reduce the number of menus necessary. If the display rate is slow, then designers should place fewer items on each menu to reduce the display time. If the response time is long and the display rate is low, menu selection is unappealing and command-language strategies, in spite of the greater memory demands that they place on users, become more attractive.

With short response times and rapid display rates, menu selection becomes a lively medium that can be attractive even for frequent and knowledgeable users. In almost every case studied, user performance and preference improved with broader and shallower menus. For most situations, designers are well advised to increase the size of menus, if they can reduce the number of menus.

7.5 Fast Movement Through Menus

Even with short response times and high display rates, frequent menu users may become annoyed if they must make several menu selections to complete a simple task. There is an advantage to reducing the number of menus by increasing the number of items per menu, but this strategy may not be sufficient. As response times lengthen and display rates decrease, the need for shortcuts through the menus increases.

Instead of creating a command language to accomplish the task with positional or keyword parameters, we can refine the menu approach to accommodate expert and frequent users. Three approaches have been used: allow typeahead for known menu choices, assign names to menus to allow direct access, and create menu macros that allows users to assign names to frequently used menu sequences.

7.5.1 Menus with typeahead: The BLT approach

A natural way to permit frequent menu users to speed through the menus is to allow *typeahead*. The user does not have to wait to see the menus before choosing the items, but can type a string of letters or numbers when pre-

sented with the main menu. Typeahead becomes important when the menus are familiar and response time or display rates are slow, as they are in many voice-mail systems. Most telephone-inquiry systems, electronic-mail systems, and Windows 95 applications allow the experienced user to enter a string of keypresses to select from a series of menus.

If the menu items are identified with single letters, then the concatenation of menu selections in the typeahead scheme generates a command name that acquires mnemonic value. To users of a photo-library search system that offered menus with typeahead, a color slide portrait quickly became known as a CSP, and a black-and-white print of a landscape became known as a BPL. Each mnemonic comes to be remembered and chunked as a single concept. This strategy quickly became known as the *BLT approach,* after the abbreviation for a bacon, lettuce, and tomato sandwich.

The attraction of the BLT approach is that users can move gracefully from being novice menu users to being knowledgeable command users. There are no new commands to learn; as soon as users become familiar with one branch of the tree, they can apply that knowledge to speed up their work. Learning can be incremental; users can apply one-, two-, or three-letter typeahead, and then explore the less familiar menus. If users forget part of the tree, they simply revert to menu usage.

The BLT approach requires a more elaborate parser for user input, and handling of nonexistent menu choices is a bit more problematic. It is also necessary to ensure distinct first letters for items within each menu, but ambiguity across menus presents no problem. The typeahead or BLT approach is attractive because it is powerful, is simple, and allows graceful evolution from novice to expert.

7.5.2   Menu names or bookmarks for direct access

A second approach to support frequent users is to use numbered menu items and to assign *menu names* to each menu frame. Users can follow the menus or, if they know the name of their destination, they can type it in and go there directly. The early CompuServe Information Service had a three-letter identifier for major topics, followed by a dash and a page number. Rather than working their way through three levels of menus at 30 characters per second, users knew that they could go directly to TWP-1, the start of the subtree containing today's edition of *The Washington Post.* America Online has bookmarks (Favorite Places) and keyword access.

This strategy is useful if there is only a small number of destinations that each user needs to remember. If users need to access many different portions of the menu tree, they will have difficulty keeping track of the destination names. A list of the current destination names is necessary to ensure that designers create unique names for new entries.

An empirical comparison of the learnability of the typeahead and direct-access strategies demonstrated an advantage for the latter (Laverson et al., 1987). Thirty-two undergraduates had to learn either path names (typeahead) or destination names (direct access) for a four-level menu tree. The direct-access names proved to be significantly faster to learn and were preferred. Different tree structures or menu contents may influence the outcome of similar studies.

In World Wide Web browsers, *bookmarks* provide a way for users to take shortcuts to destinations that they have visited previously. For many users, this menu of destinations can grow quickly and require hierarchical management strategies.

7.5.3 Menu macros, custom toolbars, and style sheets

A third approach to serving frequent menu users is to allow regularly used paths to be recorded by users as *menu macros* or to be placed in the *toolbar* as a user-selected icon. A user can invoke the macro or customization facility, traverse the menu structure, and then assign a name or icon. When the name or icon is invoked, the traversal is executed automatically. This mechanism allows tailoring of the system and can provide a simplified access mechanism for users who have special needs. Many word processors provide a style-sheet facility to allow users to make multiple menu selections and to record those choices as a personal style. For example, the style for chapter titles might be set to boldface, 24-point, italic, Times font, and centered text. Then, this chapter-title style can be saved and later invoked when needed as a form of macro. Users may also be allowed to rearrange the menu items to accommodate their patterns of work.

7.6 Menu Layout

Little experimental research has been done on menu layout. This section contains many subjective judgments, which are in need of empirical validation (Box 7.1).

7.6.1 Titles

Choosing the title for a book is a delicate matter for an author, editor, or publisher. A particularly descriptive or memorable title can make a big difference in reader responses. Similarly, choosing titles for menus is a complex matter that deserves serious thought.

Box 7.1

Menu Selection Guidelines

- Use task semantics to organize menus
  (single, linear sequence, tree structure, acyclic and cyclic networks)
- Prefer broad–shallow to narrow–deep
- Show position by graphics, numbers, or titles
- Use items as titles for sub trees
- Group items meaningfully
- Sequence items meaningfully
- Use brief items, begin with the keyword
- Use consistent grammar, layout, terminology
- Allow type ahead, jump ahead, or other short cuts
- Enable jumps to previous and main menu
- Consider online help; novel selection mechanisms; and optimal response
  time, display rate, screen size

For single menus, a simple descriptive title that identifies the situation is all that is necessary. With a linear sequence of menus, the titles should accurately represent the stages in the linear sequence. Consistent grammatical style can reduce confusion and brief but unambiguous noun phrases are often sufficient.

For tree-structured menus, choosing titles is more difficult. Titles such as `Main menu` or topic descriptions such as `Bank transactions` for the root of the tree clearly indicate that the user is at the beginning of a session. One potentially helpful rule is to use exactly the same words in the high-level menu items and in the titles for the next lower-level menu. It is reassuring to users to see an item such as `Business and financial services` and, after they selected it, a screen that is titled `Business and financial services`. It might be unsettling to get a screen titled `Managing your money`, even though the intent is similar. Imagine looking in the table of contents of a book and seeing a chapter title such as "The American Revolution," but, when you turn to the indicated page, finding "Our early history"—you might worry about whether you had made a mistake, and your confidence might be undermined. Similarly, when you design World Wide Web pages, you should ensure that the embedded menu item matches the title on the destination page. Using menu items as titles may encourage the menu author to choose items more

carefully so that they are descriptive in the context of the other menu items and as the title of the next menu.

A further concern is consistency in placement of titles and other features in a menu screen. Teitelbaum and Granda (1983) demonstrated that user think time nearly doubled when the position of information, such as titles or prompts, was varied on menu screens.

7.6.2 Phrasing of menu items

Just because a system has menu choices written with English words, phrases, or sentences, that is no guarantee that it is comprehensible. Individual words may not be familiar to some users (for example, "repaginate"), and often two menu items may appear to satisfy the user's needs, whereas only one does (for example, "put away" or "eject"). This enduring problem has no perfect solution. Designers can gather feedback from colleagues, users, pilot studies, acceptance tests, and user-performance monitoring. The following guidelines may seem obvious, but we state them because they are so often violated:

- *Use familiar and consistent terminology* Carefully select terminology that is familiar to the designated user community and keep a list of these terms to facilitate consistent use.

- *Ensure that items are distinct from one another* Each item should be distinguished clearly from other items. For example, Slow tours of the countryside, Journeys with visits to parks, and Leisurely voyages are less distinctive than are Bike tours, Train tours to national parks, and Cruise-ship tours.

- *Use consistent and concise phrasing* Review the collection of items to ensure consistency and conciseness. Users are likely to feel more comfortable and to be more successful with Animal, Vegetable, and Mineral than with Information about animals, Vegetable choices you can make, and Viewing mineral categories.

- *Bring the keyword to the left* Try to write menu items such that the first word aids the user in recognizing and discriminating among items— use Size of type instead of Set the type size. Users scan menu items from left to right; if the first word indicates that this item is not relevant, they can begin scanning the next item.

7.6.3 Graphic layout and design

The constraints of screen width and length, display rate, character set, and highlighting techniques strongly influence the graphic layout of menus. Presenting 50 states as menu items was natural on a large screen with rapid display

rate. On the other hand, systems with small text-only displays or slow modems must add levels of subcategories to present the same information.

Menu designers should establish guidelines for consistency of at least these menu components:

- *Titles*   Some people prefer centered titles, but left justification is an acceptable approach, especially with slow display rates.

- *Item placement*   Typically, items are left justified, with the item number or letter preceding the item description. Blank lines may be used to separate meaningful groups of items. If multiple columns are used, a consistent pattern of numbering or lettering should be used (for example, down the columns is easier to scan than across the rows).

- *Instructions*   The instructions should be identical in each menu, and should be placed in the same position. This rule includes instructions about traversals, help, or function-key usage.

- *Error messages*   If the users make an unacceptable choice, the error message should appear in a consistent position and should use consistent terminology and syntax.

- *Status reports*   Some systems indicate which portion of the menu structure is currently being searched, which page of the structure is currently being viewed, or which choices must be made to complete a task. This information should appear in a consistent position and should have a consistent structure.

Consistent formats help users to locate necessary information, focus users' attention on relevant material, and reduce users' anxiety by offering predictability.

In addition, since disorientation is a potential problem, techniques to indicate position in the menu structure can be useful. In books, different fonts and typefaces may indicate chapter, section, and subsection organization. Similarly, in menu trees, as the user goes down the tree structure, the titles can be designed to indicate the level or distance from the main menu. If graphics, fonts, typefaces, or highlighting techniques are available, they can be used beneficially. But even simple techniques with only fixed-size upper-case characters and indentation can be effective:

```
                                   * * * * * * * * * * * * * * * * * * * * * * * *
MAIN MENU                                        *  MAIN MENU  *
                                   * * * * * * * * * * * * * * * * * * * * * * * *

HOME SERVICES                      *   *   *  HOME SERVICES  *   *   *

NEWSPAPERS                         -  -  NEWSPAPERS  -  -

The New York Times                 THE  NEW  YORK  TIMES
```

This display gives a clear indication of progress down the tree. When users wait to do traversal back up the tree or to an adjoining menu at the same level, they feel confident about what action to take.

With linear sequences of menus, the users can be given a simple visual presentation of position in the sequence: *position marker*. In a computer-assisted instruction sequence with 12 menu frames, a position marker (+) just below the menu items might show progress. In the first frame, the position marker is

```
+---------
```

in the second frame, it is

```
-+--------
```

and in the final frame, it is

```
-----------+
```

The users can gauge their progress and can see how much remains to be done.

With GUIs, many possibilities exist for showing progress through successive levels of a tree-structured menu or through linear sequences. A common approach is to show a cascade of successive menu boxes set slightly lower than and slightly to the right of the previous items. For pull-down menus, *cascading or walking menus* (in which users walk through several levels at a time) are perceptually meaningful, but can present a motor challenge a user who must to move the cursor in the appropriate direction. Microsoft Windows 95 provides a convenient Start button on the lower left, but traversing the walking menu down several layers is a challenge for some users.

Another graphic innovation is to use transparent or see-through menus or tool palettes called *magic lenses* that can be dragged near to the object of interest while only partially obscuring it (Bier et al., 1994). Harrison and Vicente (1996) showed that user performance remains unchanged as the menu becomes up to 50 percent transparent, but the users make significantly more errors and their performance slows as the transparency reaches 75 percent.

With rapid high-resolution displays, more elegant visual representations are possible. Given sufficient screen space, it is possible to show a large portion of the menu map and to allow users to point at a menu item anywhere in the tree. Graphic designers or layout artists are useful partners in such design projects.

7.7   Form Fillin

Menu selection is effective for choosing an item from a list, but some tasks are cumbersome with menus. If data entry of personal names or numeric values is required, then keyboard typing becomes more attractive. When many fields of data are necessary, the appropriate interaction style is *form fillin*. For example, the user might be presented with a name and address form (Fig. 7.8). Form fillin was an important strategy in the days of 80 × 24 textual displays, and it has flourished in the world of graphical dialog boxes as well as on the World Wide Web.

The form-fillin approach is attractive because the full complement of information is visible, giving users a feeling of being in control of the dialog. Few instructions are necessary, since the display resembles familiar paper forms. On the other hand, users must be familiar with keyboards, use of the TAB key or mouse to move the cursor, error correction by backspacing, field-label meanings, permissible field contents, and use of the ENTER key.

7.7.1   Form-fillin design guidelines

There is a paucity of empirical work on form fillin, but several design guidelines have emerged from practitioners (Galitz, 1993; Brown, 1988). An experimental comparison of database update by form fillin and by a command-language strategy demonstrated a significant speed advantage for the former (Ogden and Boyle, 1982): 11 of the 12 subjects expressed a preference for the form-fill-in approach. Software tools simplify design, help to ensure consistency, ease maintenance, and speed implementation. But even with excellent tools, the designer must still make many complex decisions (Box 7.2).

The elements of form-fillin design include the following:

- *Meaningful title*   Identify the topic and avoid computer terminology.
- *Comprehensible instructions*   Describe the user's tasks in familiar terminology. Be brief; if more information is needed, make a set of help screens available to the novice user. In support of brevity, just describe the necessary action (Type the address or simply Address:) and avoid pronouns (You should type the address) or references to "the user" (The user of the form should type the address). Another useful rule is to use the word type for entering information

Name and Address

Please complete this section:

Name: | Albert Einstein

Company: | Relativity, Inc.

Address:

             | Apt #2

             | 112 Mercer Street

City: | Princeton

State/Province: | NJ

Country: | USA

ZIP/Postal Code: | 08540

Telephone Number: | 609-555-1212

Fax Number: | 609-555-2355

Your Email address: | al@ias.princeton.edu

Figure 7.8

A form-fillin design for name and address entry on a web page.

and `press` for special keys such as the TAB, ENTER, cursor movement, or programmed function (PFK, PF, or F) keys. Since "ENTER" often refers to the special key, avoid using it in the instructions (for example, do not use `Enter the address`; instead, stick to `Type the address`.) Once a grammatical style for instructions is developed, be careful to apply that style consistently.

- *Logical grouping and sequencing of fields*   Related fields should be adjacent, and should be aligned with blank space for separation between groups. The sequencing should reflect common patterns—for example, city followed by state followed by zip code.

Box 7.2

Form Fillin Design Guidelines

- Meaningful title
- Comprehensible instructions
- Logical grouping and sequencing of fields
- Visually appealing layout of the form
- Familiar field labels
- Consistent terminology and abbreviations
- Visible space and boundaries for data-entry fields
- Convenient cursor movement
- Error correction for individual characters and entire fields
- Error prevention where possible
- Error messages for unacceptable values
- Marking of optional fields
- Explanatory messages for fields
- Completion signal to support user control

- *Visually appealing layout of the form*   Using a uniform distribution of fields is preferable to crowding one part of the screen and leaving other parts blank. Alignment creates a feeling of order and comprehensibility. For example, the field labels Name, Address, and City can be right justified so that the data-entry fields are vertically aligned. This layout allows the frequent user to concentrate on the entry fields and to ignore the labels. If users are working from hardcopy, the screen should match the paper form.

- *Familiar field labels*   Common terms should be used. If Home Address were replaced by Domicile, many users would be uncertain or anxious about what to do.

- *Consistent terminology and abbreviations*   Prepare a list of terms and acceptable abbreviations and use the list diligently, making additions only after careful consideration. Instead of varying such terms as Address, Employee Address, ADDR., and Addr., stick to one term, such as Address.

- *Visible space and boundaries for data-entry fields*   Users should be able to see the size of the field and to anticipate whether abbreviations or other trimming strategies will be needed. Underscores can indicate the num-

ber of characters available on text-only displays, and an appropriate-sized box can show field length in GUIs.

- *Convenient cursor movement* Use a simple and visible mechanism for moving the cursor, such as a TAB key or cursor-movement arrows.
- *Error correction for individual characters and entire fields* Allow use of a backspace key and overtyping to enable the user to make easy repairs or changes to entire fields.
- *Error prevention* Where possible, prevent users from entering incorrect values. For example, in a field requiring a positive integer, do not allow the user to enter letters, minus signs, or decimal points.
- *Error messages for unacceptable values* If users enter an unacceptable value, the error message should appear on completion of the field. The message should indicate permissible values of the field; for example, if the zip code is entered as 28K21 or 2380, the message might be `Zip codes should have 5 digits.`
- *Optional fields clearly marked* Wherever appropriate, the word `Optional` or other indicators should be visible. Optional fields should follow required fields, whenever possible.
- *Explanatory messages for fields* If possible, explanatory information about a field or the permissible values should appear in a standard position, such as in a window on the bottom, whenever the cursor is in the field.
- *Completion signal* It should be clear to the users what they must do when they are finished filling in the fields. Generally, designers should avoid automatic completion when the final field is filled because users may wish to review or alter previous field entries.

These considerations may seem obvious, but often forms designers omit the title or an obvious way to signal completion, or include unnecessary computer file names, strange codes, unintelligible instructions, unintuitive groupings of fields, cluttered layouts, obscure field labels, inconsistent abbreviations or field formats, awkward cursor movement, confusing error-correction procedures, or hostile error messages.

Detailed design rules should reflect local terminology and abbreviations. They should specify field sequences familiar to the users; the width and height of the display device; highlighting features such as reverse video, underscoring, intensity levels, color, and fonts; the cursor-movement keys; and coding of fields.

7.7.2 List and combo boxes

In graphical environments and on the World Wide Web, designers can use scrolling list boxes to reduce the users' data-entry burdens and the resultant

Date:

| December ▶ | | 17 |

| 1997 |

Time:

| 10:00 | | AM |

Passengers:

| 2 |

Boarding City:

```
Warsaw,PL - WAW                    ⬆
Washington,DC(Any) - WAS
* Washington Dulles Intl - IAD
* Washington Natl Arpt - DCA
Waterloo,IA - ALO
Wausau,WI - CWA                    ⬇
```

Arrival City:

```
Inyokern,CA - IYK                  ⬆
Iron Mountain,MI - IMT
Ironwood,MI - IWD
Jackson,WY - JAC
Jacksonville,FL - JAX
Jamestown,ND - JMS                 ⬇
```

Airport Code: |

Airport Code: |

Figure 7.9

A web page that allows users to choose a flight-booking date, time, and number of
passengers from pop-up lists. The user then selects boarding and arrival cities by
scrollling lists or filling in airport codes.

errors (Color Plate C2). Scrolling lists can be thousands of items long, as they
are in many CD-ROM encyclopedias. Rapid selection from a long list can be
facilitated by a *combo box,* in which users can type in leading characters and
force scrolling through the list. Typical lists are alphabetically ordered to
support user typing of leading characters, but categorical lists may be useful.
The principles of menu-list sequencing apply (Section 7.3). A combination of
pop-up menus, scrolling, and form fillin can support rapid selection, even
for a multistep task such as airline scheduling (Fig. 7.9).

7.7.3   Coded fields

Columns of information require special treatment for data entry and for dis-
play. Alphabetic fields are customarily left justified on entry and on display.
Numeric fields may be left justified on entry, but then become right justified on
display. When possible, avoid entry and display of leftmost zeros in numeric
fields. Numeric fields with decimal points should line up on the decimal points.
   Pay special attention to such common fields as these:

• *Telephone numbers*   Offer a form to indicate the subfields:

    Telephone: (_ _ _) _ _ _-_ _ _ _

   Be alert to special cases, such as addition of extensions or the need for
   nonstandard formats for international numbers.

- *Social-security numbers*   The pattern for U.S. social-security numbers should appear on the screen as

```
Social-security number: _ _ _ - _ _ - _ _ _ _
```

When the user has typed the first three digits, the cursor should jump to the leftmost position of the two-digit field.

- *Times*   Even though the 24-hour clock is convenient, many people find it confusing and prefer A.M. or P.M. designations. The form might appear as

```
_ _:_ _ _ _ (9:45 AM or PM)
```

Seconds may or may not be included, adding to the variety of necessary formats.

- *Dates*   How to specify dates is one of the nastiest problems; no good solution exists. Different formats for dates are appropriate for different tasks, and European rules differ from American rules. An acceptable standard may never emerge.

  When the display presents coded fields, the instructions might show an example of correct entry; for example,

```
Date: _ _/_ _/_ _ (04/22/98 indicates April 22, 1998)
```

For many people, examples are more comprehensible than is an abstract description, such as

```
MM/DD/YY
```

- *Dollar amounts (or other currency)* The dollar sign should appear on the screen, so users then type only the amount. If a large number of whole-dollar amounts is to be entered, users might be presented with a field such as

```
Deposit amount: $_ _ _ _ _.00
```

with the cursor to the left of the decimal point. As the user types numbers, they shift left, calculator style. To enter an occasional cents amount, the user must type the decimal point to reach the 00 field for overtyping.

Other considerations in form-fillin design include multiscreen forms, mixed menus and forms, use of graphics, relationship to paper forms, use of

pointing devices, use of color, handling of special cases, and integration of a word processor to allow remarks.

7.8 Dialog Boxes

In modern GUIs, users can make some choices from pull-down or pop-up menus, but many tasks require multiple selections as well as data entry of numeric values or alphanumeric strings. The most common solution to complex tasks is to provide a dialog box for users. Familiar examples include the Open, Save, Find, Replace, and Spell Check dialog boxes (Fig. 7.10). Dialog boxes can also contain task-specific functions, such as entering customer name and address for a car rental; specifying clothing color, size, and fabric for an order-entry system; or selecting colors and textures for a geographic-information system.

Dialog-box design combines menu-selection and form-fillin issues with additional concerns about consistency across hundreds of dialog boxes and relationship with other items on the screen (Galitz, 1994). A guidelines document for dialog boxes can help to ensure appropriate consistency (Box 7.3). Dialog boxes should have meaningful titles to identify them, and should have consistent visual properties—for example, centered, mixed uppercase and lowercase, 12-point, black, Helvetica font. Dialog boxes are often shaped and sized to fit each situation, but distinctive sizes or aspect ratios may be used to signal errors, confirmations, or components of the application. Within a dia-
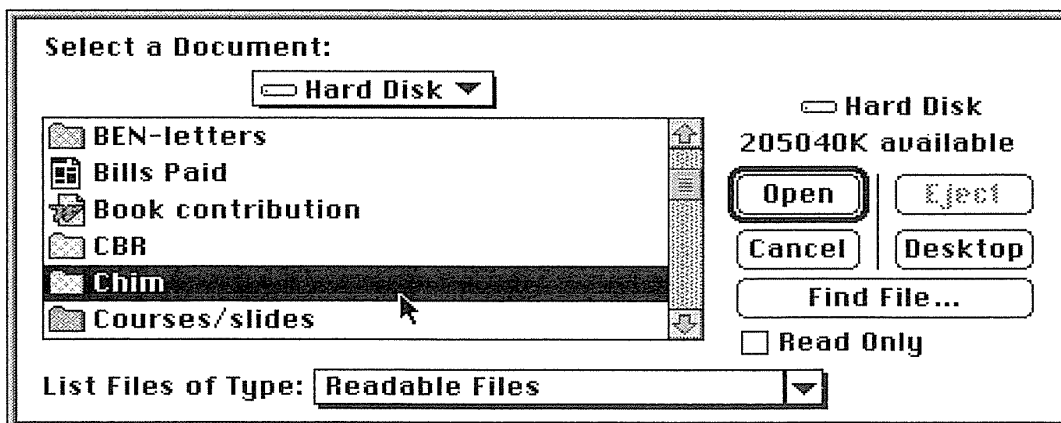

Figure 7.10

Open dialog box from Microsoft Word for the Macintosh.

Box 7.3

Dialog Box Guidelines

Internal layout: like that of menus and forms

- Meaningful title, consistent style
- Top-left to bottom-right sequencing
- Clustering and emphasis
- Consistent layouts (margins, grid, whitespace, lines, boxes)
- Consistent terminology, fonts, capitalization, justification
- Standard buttons (OK, Cancel)
- Error prevention by direct manipulation

External relationship

- Smooth appearance and disappearance
- Distinguishable but small boundary
- Size small enough to reduce overlap problems
- Display close to appropriate items
- No overlap of required items
- Easy to make disappear
- Clear how to complete/cancel

---

log box, there should be standard margins and visual organization, typically from top-left to bottom-right for languages that read left to right. A grid structure helps to organize the contents, and symmetry can be used to provide order when appropriate. Clustering of related items within a box or separation by horizontal and vertical rules gives users help in understanding the contents. Emphasis can be added by color, font size, or style of type.

The elements of a dialog box will depend on the toolkit or design tool (see Chapter 5), but they usually include buttons, check boxes, fill in fields, list boxes, combo boxes, and sliders. Standard buttons—with consistent labels, colors, and fonts—help users to navigate correctly and quickly. Where possible, users should be able to undo each step, and should be prevented from making errors.

Dialog-box design also involves the relationship with the current contents of the screen. Since dialog boxes usually pop up on top of some portion of the screen, there is a danger that they will obscure relevant information. Therefore, dialog boxes should be as small as is reasonable to minimize the overlap

and visual disruption. Dialog boxes should appear near, but not on top of, the related screen items. When a user clicks on a city on a map, the dialog box about the city should appear just next to the click point. The most common annoyance is to have the Find or Spell Check box obscure a relevant part of the text.

Dialog boxes should be distinct enough that users can easily distinguish them from the background, but should not be so harsh as to be visually disruptive. Finally, dialog boxes should disappear easily with as little visual disruption as possible (see Section 13.4 and 13.5).

When tasks are complex, multiple dialog boxes may be needed, leading some designers to chose a tabbed dialog box, in which two to 20 protruding tabs indicate the presence of multiple dialog boxes. This technique can be effective, but carries with it the potential problem of too much fragmentation; users may have a hard time finding what they want underneath the tabs. A smaller number of larger dialog boxes may be advantageous, since users usually prefer doing visual search to having to remember where to find a desired control.

7.9   Practitioner's Summary

Concentrate on organizing the structure and sequence of menus to match the users' tasks, ensure that each menu is a meaningful task-related unit, and create items that are distinctive and comprehensible. If some users make frequent use of the system, then typeahead, shortcut, or macro strategies should be allowed. Permit simple traversals to the previously displayed menu and to the main menu. Be sure to conduct human-factors tests and to involve human-factors specialists in the design process. When the system is implemented, collect usage data, error statistics, and subjective reactions to guide refinement.

Whenever possible, use software tools to produce and display a menu, form fillin, or dialog box. Commercial systems reduce implementation time, ensure consistent layout and instructions, and simplify maintenance.

7.10   Researcher's Agenda

Experimental research could help to refine the design guidelines concerning organization and sequencing in single and linear sequences of menus. How can differing communities of users be satisfied with a common organization when their information needs are markedly different? Should

users be allowed to tailor the structure of the menus, or is there greater advantage in compelling everyone to use the same structure and terminology? Should a tree structure be preserved even if some redundancy is introduced?

Research opportunities abound. Depth-versus-breadth tradeoffs under differing conditions need to be studied to provide guidance for designers. Layout strategies, wording of instructions, phrasing of menu items, graphic design, and response time are all excellent candidates for experimentation. Exciting possibilities are becoming available with larger screens and novel selection devices.

Implementers would benefit from advanced software tools to automate creation, management, usage-statistics gathering, and evolutionary refinement. Portability could be enhanced to facilitate transfer across systems, and internationalization could be facilitated by tools to support redesign for multiple national languages.


World Wide Web Resources   WWW

Information on menu, form fillin and dialog box design including empirical studies and examples of systems. The most interesting experience is scanning the World Wide Web to see how designers have laid out menu trees or aligned form-fillin boxes.

http://www.aw.com/DTUI

References

Ahlberg, C. and Shneiderman, B., AlphaSlider: A compact and rapid selector, *Proc. CHI '94 Human Factors in Computer Systems*, ACM, New York (April 1994), 365–371.

Bier, Eric, Stone, Maureen, Fishkin, Ken, Buxton, William, and Baudel, T., A taxonomy of see-through tools, *Proc. CHI '94 Human Factors in Computing Systems*, ACM, New York (1994), 358–364.

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

Callahan, Jack, Hopkins, Don, Weiser, Mark, and Shneiderman, Ben, An empirical comparison of pie versus linear menus, *Proc. CHI '88 Human Factors in Computer Systems*, ACM, New York (1988), 95–100.

Card, Stuart K., User perceptual mechanisms in the search of computer command menus, *Proc. Human Factors in Computer Systems*, Washington, D.C., Chapter of ACM (March 1982), 190–196.

Clauer, Calvin Kingsley, An experimental evaluation of hierarchical decision-making for information retrieval, IBM Research Report RJ 1093, San Jose, CA (September 15, 1972).

Galitz, Wilbert O., *It's Time to Clean Your Windows: Designing GUIs that Work*, John Wiley and Sons, New York (1994).

Greenberg, Saul and Witten, Ian H., Adaptive personalized interfaces: A question of viability, *Behaviour and Information Technology*, 4, 1 (1985), 31–45.

Harrison, Beverly L. and Vicente, Kim J., An experimental evaluation of transparent menu usage, *Proc. CHI '96, Human Factors in Computing Systems*, ACM, New York (1996), 391–398.

Kiger, John I., The depth/breadth trade-off in the design of menu-driven user interfaces, *International Journal of Man–Machine Studies*, 20, (1984), 201–213.

Koved, Lawrence, and Shneiderman, Ben, Embedded menus: Menu selection in context, *Communications of the ACM*, 29, (1986), 312–318.

Landauer, T. K., and Nachbar, D. W., Selection from alphabetic and numeric menu trees using a touch screen: Breadth, depth, and width, *Proc. CHI '85, Human Factors in Computing Systems*, ACM, New York (April 1985), 73–78.

Laverson, Alan, Norman, Kent, and Shneiderman, Ben, An evaluation of jump-ahead techniques for frequent menu users, *Behaviour and Information Technology*, 6, (1987), 97–108.

Liebelt, Linda S., McDonald, James E., Stone, Jim D., and Karat, John, The effect of organization on learning menu access, *Proc. Human Factors Society, Twenty-Sixth Annual Meeting*, Santa Monica, CA (1982), 546–550.

McDonald, James E., Stone, Jim D., and Liebelt, Linda S., Searching for items in menus: The effects of organization and type of target, *Proc. Human Factors Society, Twenty-Seventh Annual Meeting*, Santa Monica, CA (1983), 834–837.

Mitchell, Jeffrey and Shneiderman, Ben, Dynamic versus static menus: An experimental comparison, *ACM SIGCHI Bulletin*, 20, 4 (1989), 33–36.

Norman, Kent, *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*, Ablex, Norwood, NJ (1991).

Norman, Kent L. and Chin, John P., The effect of tree structure on search in a hierarchical menu selection system, *Behaviour and Information Technology*, 7, (1988), 51–65.

Norman, Kent L. and Chin, John P., The menu metaphor: Food for thought, *Behaviour and Information Technology*, 8, 2 (1989), 125–134.

Ogden, William C. and Boyle, James M., Evaluating human–computer dialog styles: Command versus form/fill-in for report modification, *Proc. Human Factors Society, Twenty-Sixth Annual Meeting*, Santa Monica, CA (1982), 542–545.

Robertson, G., McCracken, D., and Newell, A., The ZOG approach to man–machine communication, *International Journal of Man–Machine Studies*, 14, (1981), 461–488.

Sears, Andrew and Shneiderman, Ben, Split menus: Effectively using selection frequency to organize menus, *ACM Transactions on Computer-Human Interaction*, 1, 1 (1994), 27–51.

Shneiderman, Ben (Editor), *Hypertext on Hypertext*, Hyperties disk with 1 Mbyte data and graphics incorporating July 1988 CACM, ACM Press, New York (July 1988).

Shneiderman, Ben and Kearsley, Greg, *Hypertext Hands-On! An Introduction to a New Way of Organizing and Accessing Information*, Addison-Wesley, Reading, MA; book and hypertext disk using Hyperties (May 1989).

Somberg, Benjamin, and Picardi, Maria C., Locus of information familiarity effect in the search of computer menus, *Proc. Human Factors Society, Twenty-Seventh Annual Meeting*, Santa Monica, CA (1983), 826–830.

Tapia, Mark A., and Kurtenbach, Gordon, Some design refinements and principles on the appearance and behavior of marking menus, *Proc. User Interface Software and Technology '95*, ACM, New York (1995), 189–195.

Teitelbaum, Richard C., and Granda, Richard, The effects of positional constancy on searching menus for information, *Proc. CHI '83, Human Factors in Computing Systems*, ACM, New York (1983), 150–153.

Wallace, Daniel F., Anderson, Nancy S., and Shneiderman, Ben, Time stress effects on two menu selection systems, *Proc. Human Factors Society, Thirty-First Annual Meeting*, Santa Monica, CA (1987), 727–731.
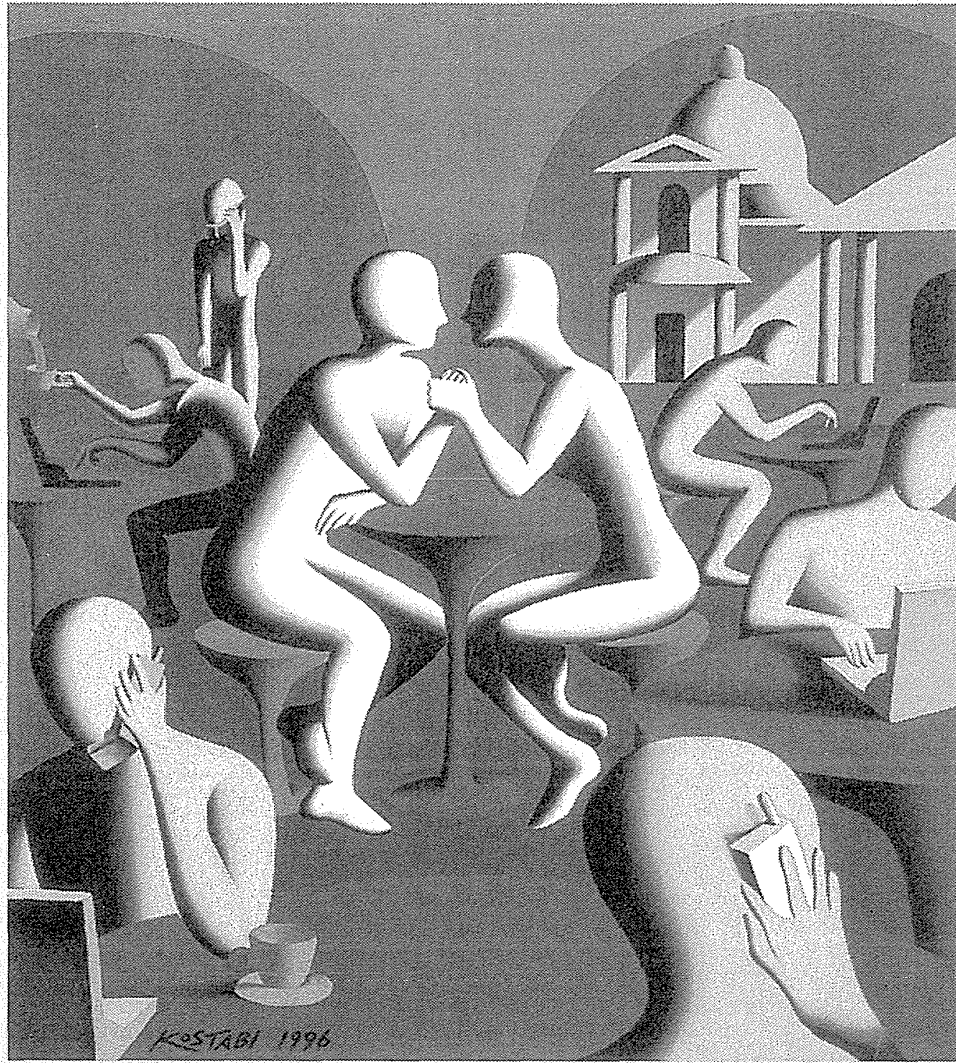
Mark Kostabi, *Oasis (Yellow Meditation)*, 1996

8

Command and Natural Languages

I soon felt that the forms of ordinary language were far too diffuse.... I was not long in deciding that the most favorable path to pursue was to have recourse to the language of signs. It then became necessary to contrive a notation which ought, if possible, to be at once simple and expressive, easily understood at the commencement, and capable of being readily retained in the memory.

Charles Babbage, "On a method of expressing
by signs the action of machinery," 1826

8.1 Introduction

The history of written language is rich and varied. Early tally marks and pictographs on cave walls existed for millennia before precise notations for numbers or other concepts appeared. The Egyptian hieroglyphs of 5000 years ago were a tremendous advance because standard notations facilitated communication across space and time. Eventually languages with a small alphabet and rules of word and sentence formation dominated because of the relative ease of learning, writing, and reading. In addition to these natural languages, special languages for mathematics, music, and chemistry emerged because they facilitated communication and problem solving. In the twentieth century, novel notations were created for such diverse domains as dance, knitting, higher forms of mathematics, logic, and DNA molecules.

The basic goals of language design are

- Precision
- Compactness
- Ease in writing and reading
- Completeness
- Speed in learning
- Simplicity to reduce errors
- Ease of retention over time

Higher-level goals include

- Close correspondence between reality and the notation
- Convenience in carrying out manipulations relevant to users' tasks
- Compatibility with existing notations
- Flexibility to accommodate novice and expert users
- Expressiveness to encourage creativity
- Visual appeal

Constraints on a language include

- The capacity for human beings to record the notation
- The match between the recording and the display media (for example, clay tablets, paper, printing presses)
- The convenience in speaking (vocalizing)

Successful languages evolve to serve the goals within the constraints.

The printing press was a remarkable stimulus to language development because it made widespread dissemination of written work possible. The computer is another remarkable stimulus to language development, not only because widespread dissemination through networks is possible, but also because computers are a tool to manipulate languages and because languages are a tool for manipulating computers.

The computer has had only a modest influence on spoken natural languages, compared to its enormous impact as a stimulus to the development of numerous new formal written languages. Early computers were built to perform mathematical computations, so the first programming languages had a strong mathematical flavor. But computers were quickly found to be effective manipulators of logical expressions, business data, graphics, sound, and text. Increasingly, computers are used to operate on the real world: directing robots, issuing dollar bills at bank machines, controlling manufacturing, and guiding spacecraft. These newer applications encourage language designers

to find convenient notations to direct the computer while preserving the needs of people to use the language for communication and problem solving.

Therefore, effective computer languages must not only represent the users' tasks and satisfy the human needs for communication, but also be in harmony with mechanisms for recording, manipulating, and displaying these languages in a computer.

Computer programming languages such as FORTRAN, COBOL, ALGOL, PL/I, and Pascal, that were developed in the 1960s and early 1970s were designed for use in a noninteractive computer environment. Programmers would compose hundreds or thousands of lines of code, carefully check that code, and then *compile* or interpret it by computer to produce a desired result. Incremental programming was one of the design considerations in BASIC and in advanced languages such as LISP, APL, and PROLOG. Programmers in these languages were expected to build small pieces online and to test the pieces interactively. Still, the common goal was to create a large program that was preserved, studied, extended, and modified. The attraction of rapid compilation and execution led to the widespread success of the compact, but sometimes obscure, notation used in C. The pressures for team programming, organizational standards for sharing, and the increased demands for reusability promoted encapsulation and the development of object-oriented programming concepts in languages such as ADA and C++. The demands of network environments and the pursuit of cross-platform tools led to the emergence of Java.

Scripting languages emphasizing screen presentation and mouse control became popular in the late 1980s, with the appearance of HyperCard, Super-Card, ToolBook, and so on. These languages included novel operators, such as ON MOUSEDOWN, BLINK, or IF FIRST CHARACTER OF THE MESSAGE BOX IS 'A'. Java expanded the possibilities for web-oriented screen management, secure network operations, and portability.

World Wide Web addresses can be seen as a form of command language. Users come to memorize the structure and to memorize favorite sites, even though the typical usage is to click to select from a web page or a bookmark list. Web addresses begin with a protocol name (http, ftp, gopher, and so on), followed by a colon and two forward slashes. Then, the server address (which also can include country codes or domain names, such as gov, edu, mil, org), directory path, and file name; for example,

```
http://www.whitehouse.gov/WH/glimpse/top.html
```

Database-query languages for relational databases were developed in the middle to late 1970s and led to the widely used SQL. It emphasized short segments of code (three to 20 lines) that could be written at a terminal and

executed immediately. The goal of the user was to create a result, rather than a program. A key part of database-query languages and information-retrieval languages was the specification of Boolean operations: AND, OR, and NOT.

Command languages, which originated with operating-systems commands, are distinguished by their immediacy and by their impact on devices or information. Users issue a command and watch what happens. If the result is correct, the next command is issued; if not, some other strategy is adopted. The commands are brief and their existence is transitory. Command histories are sometimes kept and macros are created in some command languages, but the essence of command languages is that they have an ephemeral nature and that they produce an immediate result on some object of interest.

Command languages are distinguished from menu-selection systems in that their users must recall notation and initiate action. Menu-selection users view or hear the limited set of items; they respond more than initiate. Command-language users are often called on to accomplish remarkable feats of memorization and typing. For example, this Unix command, used to delete blank lines from a file, is not obvious:

```
grep -v ^$ filea > fileb
```

Similarly, to get printout on unlined paper on a high-volume laser printer, a user at one installation was instructed to type

```
CP TAG DEV E VTSO LOCAL 2 OPTCD=J F=3871 X=GB12
```

The puzzled user was greeted with a shrug of the shoulders and the equally cryptic comment that "Sometimes, logic doesn't come into play; it's just getting the job done." This style of work may have been acceptable in the past, but user communities and their expectations are changing. While there are still millions of users of command languages, the development of new command languages has slowed dramatically due to the emergence of direct-manipulation and menu-selection interfaces.

Command languages may consist of single commands or may have complex syntax (Section 8.2). The language may have only a few operations or may have thousands. Commands may have a hierarchical structure or may permit concatenation to form variations (Section 8.3). A typical form is a verb followed by a noun object with qualifiers or arguments for the verb or noun, for example, PRINT MYFILE 3 COPIES. Abbreviations may be permitted (Section 8.5). Feedback may be generated for acceptable commands, and error messages (Section 11.2) may result from unacceptable forms or typos. Command-language systems may offer the user brief prompts or may be close to

menu-selection systems (Section 8.6). Finally, natural-language interaction can be considered as a complex form of command language (Section 8.7).

8.2   Functionality to Support Users' Tasks

People use computers and command-language systems to accomplish a wide range of work, such as text editing, operating-system control, bibliographic retrieval, database manipulation, electronic mail, financial management, airline or hotel reservations, inventory, manufacturing process control, and adventure games.

People will use a computer system if it gives them powers not otherwise available. If the power is attractive enough, people will use a system despite a poor user interface. Therefore, the first step for the designer is to determine the functionality of the system by studying the users' task domain. The outcome is a list of task actions and objects, which is then abstracted into a set of interface actions and objects. These items, in turn, are represented with the low-level interface syntax.

A common design error is to provide an excessive numbers of objects and actions, which can overwhelm the user. Excessive objects and actions take more code to maintain; potentially cause more bugs; possibly incur slower execution; and require more help screens, error messages, and user manuals (see Chapters 11 and 12). For the user, excess functionality slows learning, increases the chance of error, and adds the confusion of longer manuals, more help screens, and less-specific error messages. On the other hand, insufficient objects or actions leaves the user frustrated because a desired function is not supported. For instance, users might have to copy a list with a pen and paper because there is no simple print command or to reorder a list by hand because there is no sort command.

Careful task analysis might result in a table of user communities and tasks, with each entry indicating expected frequency of use. The high-volume tasks should be made easy to carry out. The designer must decide which communities of users are the prime audience for the system. Users may differ in their position in an organization, their knowledge of computers, or their frequency of system use.

At an early stage, the destructive actions—such as deleting objects or changing formats—should be evaluated carefully to ensure that they are reversible, or at least are protected from accidental invocation. Designers should also identify error conditions and prepare error messages. A transition diagram showing how each command takes the user to another state is a highly benefi-
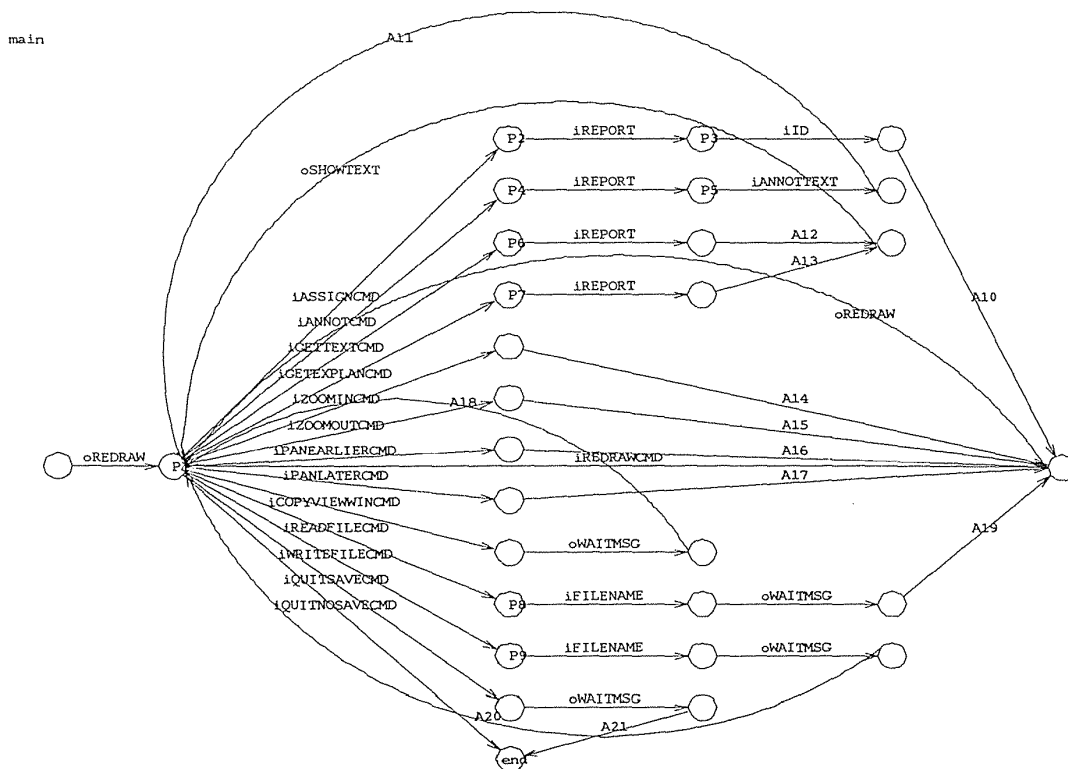
Figure 8.1

Transition diagram indicating user inputs with an "i" and computer outputs with an "o." This relatively simple diagram shows only a portion of the system. Complete transition diagrams may comprise many pages. (Courtesy of Robert J. K. Jacob, Naval Research Laboratory, Washington, D.C.)

cial aid to design, as well as for eventual training of users (Fig. 8.1). A diagram that grows too complicated may signal the need for system redesign.

Major considerations for expert users are the possibilities of tailoring the language to suit personal work styles and of creating named macros to permit several operations to be carried out with a single command. Macro facilities allow extensions that the designers could not foresee or that are beneficial to only a small fragment of the user community. A macro facility can be a full programming language that might include specification of arguments, conditionals, iteration, integers, strings, and screen-manipulation primitives, plus library and editing tools. Well-developed macro facilities are one of the strong attractions of command languages.

8.3   Command-Organization Strategies

Several strategies for command organization have emerged. A unifying interface concept or metaphor aids learning, problem solving, and retention. Electronic-mail enthusiasts conduct lively discussions about the metaphoric merits of such task-related objects as file drawers, folders, documents, memos, notes, letters, or messages. They debate the appropriate interface actions (CREATE, EDIT, COPY, MOVE, DELETE) and the choice of action pairs: LOAD/SAVE (too much in the computer domain), READ/WRITE (acceptable for letters, but awkward for file drawers), or OPEN/CLOSE (acceptable for folders, but awkward for notes).

Designers often err by choosing a metaphor closer to the computer domain than to the user's task domain. Of course, metaphors can mislead the user, but careful design can reap the benefits while reducing the detriments. Having adopted an interface concept or metaphor for actions and objects, the designer must then choose a strategy for the command syntax. Mixed strategies are possible, but learning, problem solving, and retention may be aided by limitation of complexity.

8.3.1   Simple command set

In a *simple command set*, each command is chosen to carry out a single task, and the number of commands matches the number of tasks. When there is only a small number of tasks, this approach can produce a system that is simple to learn and use. Some MUD commands are simple, such as look, go, who, rooms, and quit. When there is a large number of commands, however, there is danger of confusion. The vi editor on Unix systems offers many commands while attempting to keep the number of keystrokes low. The result is complex strategies that employ single letters, shifted single letters, and the CTRL key plus single letters (Fig. 8.2). Furthermore, some commands stand alone, whereas others must be combined, often in irregular patterns.

8.3.2   Command plus arguments

The second option is to follow each command (COPY, DELETE, PRINT) by one or more *arguments* (FILEA, FILEB, FILEC) that indicate objects to be manipulated:

```
COPY FILEA,FILEB
DELETE FILEA
PRINT FILEA,FILEB,FILEC
```

The commands may be separated from the arguments by a blank or other delimiter, and the arguments may have blanks or delimiters between them

VI COMMANDS TO MOVE THE CURSOR

Moving within a window

H	go to home position (upper left)
L	go to last line
M	go to middle line
(CR)	next line (carriage return)
+	next line
-	previous line
CTRL-P	previous line in same column
CTRL-N	next line in same column
(LF)	next line in same column (line feed)

Moving within a line

0	go to start of line
$	go to end of line
(space)	go right one space
CRTL-H	go left one space
h	go left one space
w	forward one word
b	backward one word
e	end (rightmost) character of a word
)	forward one sentence
(	backward one sentence
}	forward one paragraph
{	backward one paragraph
W	blank out a delimited word
B	backwards blank out a delimited word
E	go to the end of a delimited word

Finding a character

fx	find the character x going forward
Fx	find the character x going backward
tx	go up to x going forward
Tx	go up to x going backward

Scrolling the window

CTRL-F	go forward one screen
CTRL-B	go backward one screen
CTRL-D	go forward one half screen
CTRL-U	go backward one half screen
G	go to line
/pat	go to line with pattern forward
pat	go to line with pattern backward

Figure 8.2

The profusion of cursor-movement commands in vi enable expert users to get tasks done with just a few actions, but they can overwhelm novice and intermittent users.

(Schneider et al., 1984). Keyword labels for arguments may be helpful to some users; for example,

```
COPY FROM=FILEA TO=FILEB
```

The labels require extra typing and thus increase chances of a typo, but readability is improved and order dependence is eliminated.

8.3.3   Command plus options and arguments

Commands may have *options* (3, HQ, and so on) to indicate special cases. For example,

```
PRINT/3,HQ FILEA
PRINT (3,HQ) FILEA
PRINT FILEA -3,HQ
```

may produce three copies of FILEA at the printer in the headquarters building. As the number of options grows, the complexity can become overwhelming and the error messages must be less specific. The arguments also may have options, such as version numbers, privacy keys, or disk addresses.

The number of arguments, of options, and of permissible syntactic forms can grow rapidly. One airline-reservations system uses the following command to check the seat availability on a flight on August 21, from Washington's National Airport (DCA) to New York's LaGuardia Airport (LGA) at about 3:00 P.M.:

```
A0821DCALGA0300P
```

Even with substantial training, error rates can be high with this approach, but frequent users seem to manage and even to appreciate the compact form of this type of command.

The Unix command-language system is widely used, in spite of the complexity of its command formats, which have been criticized severely (Norman, 1981). Here again, users will master complexity to benefit from the rich functionality in a system. Observed error rates with actual use of Unix commands have ranged from 3 to 53 percent (Hanson et al., 1984). Even common commands have generated high syntactic error rates: mv (18 percent) or cp (30 percent). Still, the complexity has a certain attraction for a portion of the potential user community. Users gain satisfaction in overcoming the difficulties and becoming one of the inner circle (gurus or wizards) who are knowledgeable about system features—command-language macho.

8.3.4   Hierarchical command structure

In a *hierarchical command structure,* the full set of commands is organized into a tree structure, like a menu tree. The first level might be the command action, the second might be an object argument, and the third might be a destination argument:

Action	Object	Destination
CREATE	File	File
DISPLAY	Process	Local printer
REMOVE	Directory	Screen
COPY		Remote printer
MOVE		

If a hierarchical structure can be found for a set of tasks, it offers a meaningful structure to a large number of commands. In this case, $5 \times 3 \times 4 = 60$ tasks can be carried out with only five command names and one rule of formation. Another advantage is that a command-menu approach can be developed to aid the novice or intermittent user, as was done in VisiCalc and later in Lotus 1–2–3 and Excel.

8.4   The Benefits of Structure

Human learning, problem solving, and memory are greatly facilitated by meaningful structure. If command languages are well designed, users can recognize the structure and can easily encode it in their semantic-knowledge storage. For example, if users can uniformly edit such objects as characters, words, sentences, paragraphs, chapters, and documents, this meaningful pattern is easy for them to learn, apply, and recall. On the other hand, if they must overtype a character, change a word, revise a sentence, replace a paragraph, substitute a chapter, and alter a document, then the challenge and potential for error grow substantially, no matter how elegant the syntax (Scapin, 1982).

Meaningful structure is beneficial for task concepts, computer concepts, and syntactic details of command languages. Yet many systems fail to provide a meaningful structure. Users of one operating system display information with the LIST, QUERY, HELP, and TYPE commands, and move objects with the PRINT, TYPE, SPOOL, SEND, COPY, or MOVE commands. Defaults are inconsistent, four different abbreviations for PRINT and LINECOUNT are required, binary choices vary between YES/NO and ON/OFF, and function-key usage is inconsistent. These flaws emerge from multiple uncoordinated

design groups and reflect insufficient attention by the managers, especially as features are added over time.

An explicit list of design conventions in a *guidelines document* can be an aid to designers and managers. Exceptions may be permitted, but only after thoughtful discussions. Users can learn systems that contain inconsistencies, but they do so slowly and with a high chance of making mistakes.

8.4.1   Consistent argument ordering

Several studies have shown that there are benefits associated with using a *consistent order for arguments* (Barnard et al., 1981).

Inconsistent order of arguments	Consistent order of arguments
`SEARCH file no,message id`	`SEARCH message id,file no`
`TRIM message id,segment size`	`TRIM message id,segment size`
`REPLACE message id,code no`	`REPLACE message id,code no`
`INVERT group size,message id`	`INVERT message id,group size`

Time to perform tasks for the 48 subjects was significantly shorter with the consistent argument ordering.

8.4.2   Symbols versus keywords

Evidence that command structure affects performance comes from a comparison of 15 commands in a commercially used symbol-oriented text editor, and revised commands that had a more keyword-oriented style (Ledgard et al., 1980). Here are three sample commands:

Symbol editor	Keyword editor
`FIND:/TOOTH/;-1`	`BACKWARD TO "TOOTH"`
`LIST;10`	`LIST 10 LINES`
`RS:/KO/,/OK/;*`	`CHANGE ALL "KO" TO "OK"`

The revised commands performed the same functions. Single-letter abbreviations (`L;10` or `L 10 L`) were permitted in both editors, so the number of keystrokes was approximately the same. The difference in the revised commands was that keywords were used in an intuitively meaningful way, but there were no standard rules of formation. Eight subjects at three levels of text-editor experience used both versions in this counterbalanced-order within-subjects design. The results (Table 8.1) clearly favored the keyword editor, indicating that command-formation rules do make a difference.

Table 8.1

Effects of revised text-editor commands on three levels of users. (Ledgard et al., 1980.)

	Percentage of Task Completed		Percentage of Erroneous Commands	
	Symbol	*Keyword*	*Symbol*	*Keyword*
Inexperienced users	28	42	19.0	11.0
Familiar users	43	62	18.0	6.4
Experienced users	74	84	9.9	5.6

8.4.3   Hierarchical structure and congruence

Carroll (1982) altered two design variables to produce four versions of a 16-command language for controlling a robot (Table 8.2). Commands could be hierarchical (verb-object–qualifier) or nonhierarchical (verb only) and congruent (for example, ADVANCE/RETREAT or RIGHT/LEFT) or noncongruent (GO/BACK or TURN/LEFT). Carroll uses *congruent* to refer to meaningful pairs of opposites (*symmetry* might be a better term). Hierarchical structure and congruence have been shown to be advantageous in psycholinguistic experiments. Thirty-two undergraduate subjects studied one of the four command sets in a written manual, gave subjective ratings, and then carried out paper-and-pencil tasks.

Subjective ratings prior to performance of tasks showed that subjects disapproved of the nonhierarchical noncongruent form and gave the highest rating for the nonhierarchical congruent form. Memory and problem-solving tasks showed that congruent forms were clearly superior and that the hierarchical forms were superior for several dependent measures. Error rates were dramatically lower for the congruent hierarchical forms.

This study assessed performance of new users of a small command language. Congruence helped subjects to remember the natural pairs of concepts and terms. The hierarchical structure enabled subjects to master 16 commands with only one rule of formation and 12 keywords. With a larger command set—say, 60 or 160 commands—the advantage of hierarchical structure should increase, assuming that a hierarchical structure could be found to accommodate the full set of commands. Another conjecture is that retention should be facilitated by the hierarchical structure and congruence.

Carroll's study was conducted during a half-day period; with one week of regular use, differences probably would be reduced substantially. However, with intermittent users or with users under stress, the hierarchical congruent form might again prove superior. An online experiment might have

Table 8.2

Command sets and partial results. (Carroll 1982.)

CONGRUENT		NONCONGRUENT	
Hierarchical	*Non-hierarchical*	*Hierarchical*	*Non-hierarchical*
MOVE ROBOT FORWARD	ADVANCE	MOVE ROBOT FORWARD	GO
MOVE ROBOT BACKWARD	RETREAT	CHANGE ROBOT BACKWARD	BACK
MOVE ROBOT RIGHT	RIGHT	CHANGE ROBOT RIGHT	TURN
MOVE ROBOT LEFT	LEFT	MOVE ROBOT LEFT	LEFT
MOVE ROBOT UP	STRAIGHTEN	CHANGE ROBOT UP	UP
MOVE ROBOT DOWN	BEND	MOVE ROBOT DOWN	BEND
MOVE ARM FORWARD	PUSH	CHANGE ARM FORWARD	POKE
MOVE ARM BACKWARD	PULL	MOVE ARM BACKWARD	PULL
MOVE ARM RIGHT	SWING OUT	CHANGE ARM RIGHT	PIVOT
MOVE ARM LEFT	SWING IN	MOVE ARM LEFT	SWEEP
MOVE ARM UP	RAISE	MOVE ARM UP	REACH
MOVE ARM DOWN	LOWER	CHANGE ARM DOWN	DOWN
CHANGE ARM OPEN	RELEASE	CHANGE ARM OPEN	UNHOOK
CHANGE ARM CLOSE	TAKE	MOVE ARM CLOSE	GRAB
CHANGE ARM RIGHT	SCREW	MOVE ARM RIGHT	SCREW
CHANGE ARM LEFT	UNSCREW	CHANGE ARM LEFT	TWIST

Subjective Ratings (1 = Best, 5 = Worst)

	1.86	1.63	1.81	2.73
Test Scores	14.88	14.63	7.25	11.00
Errors	0.50	2.13	4.25	1.63
Omissions	2.00	2.50	4.75	4.15

been more realistic and would have brought out differences in command length that would have been a disadvantage to the hierarchical forms because of the greater number of keystrokes required. However, the hierarchical forms could all be replaced with three-letter abbreviations (for example, MAL for MOVE ARM LEFT), thereby providing an advantage even in keystroke counts.

In summary, sources of structure that have proved advantageous include these:

- Positional consistency
- Grammatical consistency
- Congruent pairing
- Hierarchical form

In addition, as discussed in Section 8.5, a mixture of meaningfulness, mnemonicity, and distinctiveness is helpful.

8.5 Naming and Abbreviations

In discussing command-language names, Schneider (1984) takes a delightful quote from Shakespeare's *Romeo and Juliet*: "A rose by any other name would smell as sweet." As Schneider points out, the lively debates in design circles suggest that this concept does not apply to command-language names. Indeed, the command names are the most visible part of a system and are likely to provoke complaints from disgruntled users.

Critics (Norman, 1981, for example) focus on the strange names in Unix, such as mkdir (make directory), cd (change directory), ls (list directory), rm (remove file), and pwd (print working directory); or in IBM's CMS, such as SO (temporarily suspend recording of trace information), LKED (link edit), NUCXMAP (identify nucleus extensions), and GENDIRT (generate directory). Part of the concern is the inconsistent abbreviation strategies that may take the first few letters, first few consonants, first and final letter, or first letter of each word in a phrase. Worse still are abbreviations with no perceivable pattern.

8.5.1 Specificity versus generality

Names are important for learning, problem solving, and retention over time. When it contains only a few names, a command set is relatively easy to master; but when it contains hundreds of names, the choice of meaningful, organized sets of names becomes more important. Similar results were found for programming tasks, where variable name choices were less important in small modules with from 10 to 20 names than in longer modules with dozens or hundreds of names.

With larger command sets, the names do make a difference, especially if they support congruence or some other meaningful structure. One naming-rule debate revolves around the question of *specificity versus generality* (Rosenberg, 1982). Specific terms can be more descriptive than general ones are, and if they are more distinctive, they may be more memorable. General terms may be more familiar and therefore easier to accept. Two weeks after a training session with 12 commands, subjects were more likely to recall and recognize the meaning of specific commands than those of general commands (Barnard et al., 1981).

In a paper-and-pencil test, 84 subjects studied one of seven sets of eight commands (Black and Moran, 1982). Two of the eight commands—the

commands for inserting and deleting text—are shown here in all seven versions:

Infrequent, discriminating words	insert	delete
Frequent, discriminating words	add	remove
Infrequent, nondiscriminating words	amble	perceive
Frequent, nondiscriminating words	walk	view
General words (frequent, nondiscriminating)	alter	correct
Nondiscriminating nonwords (nonsense)	GAC	MIK
Discriminating nonwords (icons)	abc-adbc	abc-ac

The "infrequent, discriminating" command set resulted in faster learning and superior recall than did other command sets. The general words were correlated with the lowest performance on all three measures. The nonsense words did surprisingly well, supporting the possibility that, with small command sets, distinctive names are helpful even if they are not meaningful.

8.5.2   Abbreviation strategies

Even though command names should be meaningful for human learning, problem solving, and retention, they must satisfy another important criterion: They must be in harmony with the mechanism for expressing the commands to the computer. The traditional and widely used command-entry mechanism is the keyboard, which indicates that commands should use brief and kines-thetically easy codes. Commands requiring shifted keys or CTRL keys, special characters, or difficult-to-type sequences are likely to cause higher error rates. For text editing, when many commands are applied and speed is appreciated, single-letter approaches are attractive. Overall, brevity is a worthy goal, since it can speed entry and reduce error rates. Early word-processor designers pursued this approach, even when mnemonicity was sacrificed, thereby making use more difficult for novice and intermittent users.

In less demanding applications, designers have used longer command abbreviations, hoping that the gains in recognizability would be appreciated over the reduction in key strokes. Novice users may actually prefer typing the full name of a command because they have a greater confidence in its success (Landauer et al., 1983). Novices who were required to use full command names before being taught two-letter abbreviations made fewer errors with the abbreviations than did those who were taught the abbreviations from the start and than did those who could create their own abbreviations (Grudin and Barnard, 1985).

The phenomenon of preferring the full name at first appeared in our study of bibliographic retrieval with the Library of Congress's SCORPIO system. Novices preferred typing the full name, such as BROWSE or SELECT, rather

than the traditional four-letter abbreviations BRWS or SLCT, or the single-letter abbreviations B or S. After five to seven uses of the command, their confidence increased and they attempted the single-letter abbreviations. A designer of a text adventure game recognized this principle and instructed novice users to type EAST, WEST, NORTH, or SOUTH; after five full-length commands, the system tells the user about the single-letter abbreviations.

With experience and frequent use, abbreviations become attractive for, and even necessary to satisfy, the "power" user. Efforts have been made to find optimal abbreviation strategies. Several studies support the notion that abbreviation should be accomplished by a consistent strategy (Ehrenreich and Porcu, 1982; Benbasat and Wand, 1984; Schneider, 1984). Here are six potential strategies:

1. *Simple truncation*   Use the first, second, third, and so on letters of each command. This strategy requires that each command be distinguishable by the leading string of characters. Abbreviations can be all of the same length or of different lengths.

2. *Vowel drop with simple truncation*   Eliminate vowels and use some of what remains. If the first letter is a vowel, it may or may not be retained. H, Y, and W may or may not be considered as vowels for this purpose.

3. *First and final letter*   Since the first and final letters are highly visible, use them; for example, use ST for SORT.

4. *First letter of each word in a phrase*   Use the popular acronym technique, for example, with a hierarchical design plan.

5. *Standard abbreviations from other contexts*   Use familiar abbreviations such as QTY for QUANTITY, XTALK for CROSSTALK (a software package), PRT for PRINT, or BAK for BACKUP.

6. *Phonics*   Focus attention on the sound; for example, use XQT for execute.

Truncation appears to be the most effective mechanism overall, but it has its problems. Conflicting abbreviations appear often, and decoding an unfamiliar abbreviation may not be easy as when vowel dropping is used (Schneider, 1984).

8.5.3   Guidelines for using abbreviations

Ehrenreich and Porcu (1982) offer this compromise set of guidelines:

1. A *simple,* primary rule should be used to generate abbreviations for most items; a *simple* secondary rule should be used for those items where there is a conflict.

2. Abbreviations generated by the secondary rule should have a marker (for example, an asterisk) incorporated in them.

3. The number of words abbreviated by the secondary rule should be kept to a minimum.

4. Users should be familiar with the rules used to generate abbreviations.

5. Truncation should be used because it is an easy rule for users to comprehend and remember. When it produces a large number of identical abbreviations for different words, adjustments must be found.

6. Fixed-length abbreviations should be used in preference to variable-length ones.

7. Abbreviations should not be designed to incorporate endings (e.g., ING, ED, S).

8. Unless there is a critical space problem, abbreviations should not be used in messages generated by the computer and read by the user.

Abbreviations are an important part of system design and they are appreciated by experienced users. Users are more likely to use abbreviations if they are confident in their knowledge of the abbreviations and if the benefit is a savings of more than one to two characters (Benbasat and Wand, 1984). The dominance of GUIs has reduced the importance of abbreviations strategies, but when there are appropriate situations for command abbreviations, empirical tests with users should be applied.

8.6   Command Menus

To relieve the burden of memorization of commands, some designers offer users brief prompts of available commands, in a format called a *command menu*. For example, a text-only web browser called lynx displays this prompt:

```
H)elp O)ptions P)rint G)o M)ain screen Q)uit
       /=search [delete]=history list
```

Experienced users come to know the commands and do not need to read the prompt or the help screens. Intermittent users know the concepts and refer to the prompt to jog their memory and to get help in retaining the syntax for future uses. Novice users do not benefit as much from the prompt and must take a training course or consult the online help.

The prompting approach emphasizes syntax and serves frequent users. It is closer to but more compact than a numbered menu, and preserves screen space for task-related information. The early WORDSTAR editor offered the novice and intermittent user help menus containing commands with one- or two-word descriptions (Fig. 8.3). Frequent users could turn off the display of help menus, thereby gaining screen space for additional text.

```
A:GETTYS  PAGE 1 LINE 9 COL 62              INSERT ON
               < < <      M A I N   M E N U    > > >
    --Cursor Movement--      ! -Delete- !  -Miscellaneous-  !  -Other  Menus-
^S char left ^D char right !^G  char  ! ^I Tab   ^B Reform ! (from Main only)
^A word left ^F word right !DEL chr lf! ^V INSERT ON/OFF   !^J Help  ^K Block
^E line  up  ^X line down  !^T word rt!^L Find/Replce again!^Q Quick ^P Print
      --Scrolling--        !^Y  line  !RETURN End paragraph!^O Onscreen
^Z line down ^W line up    !          ! ^N Insert a RETURN !
^C screen up ^R screen down!          ! ^U Stop a command  !
L----!----!----!----!----!----!----!----!----!----!----!--------R
      Fourscore  and  seven  years  ago  our  fathers  brought  forth  on
this  continent  a  new  nation  conceived  in  liberty  and  dedicated  to
the  proposition  that  all  men  are  created  equal.   Now  we  are
engaged  in  a  great  civil  war  testing  whether  that  nation,  or  any
nation so conceived and so dedicated, can long endure.

      We  are  met  on  a  great  battlefield  of  that  war.   We  have  come
to  dedicate  a  portion  of  that  field  as  a  final  resting-place   for
those  who  here  gave  their  lives  that  that  nation  might  live.
```

Figure 8.3

The early WORDSTAR help menus, which offered the novice and intermittent users commands with one- or two-word descriptions.

In many Command menus, users can use the mouse or arrow keys to highlight their choices, or can type single-letter choices, but frequent users do not even look at the menus as they type sequences of two, three, four, or more single letters that come to be thought of as a command. Windows 95 shows the single-letter command by underscoring a letter in the menu, allowing users to perform all operations with keyboard commands (see Color Plate A1). With a fast display, command menus blur the boundaries between commands and menus.

8.7 Natural Language in Computing

Even before there were computers, people dreamed about creating machines that would accept *natural language*. It is a wonderful fantasy, and the success of word-manipulation devices such as word processors, tape recorders, and telephones may give encouragement to some people. Although there has been some progress in machine translation from one natural language to another (for example, Japanese to English), most effective systems require constrained or preprocessed input, or postprocessing of output. Undoubtedly, improvements will continue and constraints will be reduced, but high-quality reliable translations of complete documents without human intervention seem difficult to attain. Structured texts such as weather reports are translatable; technical papers are marginally translatable; novels or poems are not translatable. Language is subtle; there are many special cases,

contexts are complex and emotional relationships have a powerful and pervasive effect in human–human communication.

Although full comprehension and generation of language seems inaccessible, there are still many ways that computers can be used in dealing with natural language, such as for interaction, queries, database searching, text generation, and adventure games (Allen, 1995). So much research has been invested in natural-language systems that undoubtedly some successes will emerge, but widespread use may not develop because the alternatives may be more appealing. More rapid progress can be made if carefully designed experimental tests are used to discover the users, tasks, and interface designs for which natural-language applications are most beneficial (Oviatt, 1994; King, 1996).

8.7.1   Natural-language interaction

Researchers hope to fulfill the Star Trek scenario in which computers will respond to commands users issue by speaking (or typing) in natural language. *Natural-language interaction* (NLI) might be defined as the operation of computers by people using a familiar natural language (such as English) to give instructions and receive responses. Users do not have to learn a command syntax or to select from menus. Early attempts at generalized "automatic programming" from natural-language statements have faded, but there are continuing efforts to provide domain-specific assistance.

The problems with NLI lie in not only implementation on the computer, but also desirability for large numbers of users for a wide variety of tasks. People are different from computers, and human–human interaction is not necessarily an appropriate model for human operation of computers. Since computers can display information 1000 times faster than people can enter commands, it seems advantageous to use the computer to display large amounts of information, and to allow novice and intermittent users simply to choose among the items. Selection helps to guide the user by making clear what functions are available. For knowledgeable and frequent users, who are thoroughly aware of the available functions, a precise, concise command language is usually preferred.

In fact, the metaphors of artificial intelligence (smart machines, intelligent agents, and expert systems) may prove to be mind-limiting distractions that inhibit designers from creating the powerful tools that become commercially successful. Spreadsheets, WYSIWYG word processors, and direct-manipulation graphics tools emerged from a recognition of what users were using effectively, rather than from the misleading notions of intelligent machines. Similarly, the next generation of groupware to support collaboration, visualization, simulation, tele-operated devices, and hypermedia stem from user-centered scenarios, rather than from the machine-centered artificial-intelligence scenarios.

The OAI model may help us to sort out the issues. Most designs for NLI do not provide information about task actions and objects; users are usually

presented with a simple prompt that invites a natural-language statement. Users who are knowledgeable about the task—for example, stock-market objects and permissible actions—could make buy or sell orders by voice or typing in natural language, but compact command languages are reliable and have been preferred by these users. NLI designs also do not usually convey information about the interface—for example, tree-structuring of information, implications of a deletion, Boolean operations, or query strategies. NLI designs should relieve users from learning new syntactic rules, since they presumably will accept familiar English language requests. Therefore, NLI can be effective for users who are knowledgeable about specific tasks and interface concepts but who are intermittent users who cannot retain the syntactic details of the interface.

By this analysis, NLI might apply to checkbook maintenance (Shneiderman, 1980), where the users recognize that there is an ascending sequence of integer-numbered checks, and that each check has a single payee field, single amount, single date, and one or more signatures. Checks can be issued, voided, searched, and printed. Following this suggestion, Ford (1981) created and tested a textual NLI system for this purpose. Subjects were paid to maintain their checkbook registers by computer using an APL-based program that was refined incrementally to account for unanticipated entries. The final system successfully handled 91 percent of users' requests, such as these:

```
Pay to Safeway on 3/24/86 $29.75.
June 10 $33.00 to Madonna.
Show me all the checks paid to George Bush.
Which checks were written on October 29?
```

Users reported satisfaction with the system and were eager to use the system after completing the several months of experimentation. This study can be seen as a success for NLI, but direct-manipulation alternatives (for example, Quicken from Intuit) have proved more attractive in the marketplace. Showing a full screen of checkbook entries with a blank line for new entries might allow users to accomplish most tasks without any commands and with minimal typing. Users could search by entering partial information (for example, Bill Clinton in the payee field) and then pressing a query key.

There have been numerous informal tests of NLI systems, but only a few have been controlled experimental comparisons against some other design. Researchers seeking to demonstrate the advantage of NLI over command-language and menu approaches for creating business graphics were surprised to find no significant differences for task time, user errors, or user attitudes (Hauptmann and Green, 1983).

A more positive result was found with users of HAL, the restricted natural-language addition to Lotus 1–2–3 (Napier et al., 1989). HAL users could avoid the command-menu /WEY (for Worksheet Erase Yes), and could type requests such as \erase worksheet, \insert row, or \total all

`columns`, starting with any of the 180 permissible verbs. In an empirical study, after 1.5 days of training, 19 HAL users and 22 Lotus 1–2–3 users worked on three substantial problem sets for another 1.5 days. Performance and preference strongly favored the restricted natural-language version, but the experimenters had difficulty identifying the features that made a difference: "It is not clear whether Lotus HAL was better because it is more like English or because it takes advantage of context, but we suspect the latter is more important." By context, the authors meant features such as the cursor position or meaningful variable names that indicate cell ranges. HAL is no longer marketed.

Some NLI work has turned to automatic speech recognition and speech generation to reduce the barriers to acceptance (Oviatt, 1994). There is some advantage to use of these technologies, but the results are still meager. A promising application is the selection of painting tools by discrete-word recognition (see Section 9.4.1), thus eliminating the frustration and delay of moving the cursor from the object to the tool menu on the border and back again (Pausch, 1991). Selections are voiced, but feedback is visual. Users of the mouse plus the voice commands performed their tasks 21-percent faster than did the users who had only the mouse. Alternatives to voice, such as keyboard or touchscreen, were not tested.

There is some portion of the user spectrum that can benefit from NLI, but it may not be as large as promoters believe. Computer users usually seek predictable responses and are discouraged if they must engage in clarification dialogs frequently. Since NLI has such varied forms, the users must constantly be aware of the computer's response, to verify that their actions were recognized. Finally, visually oriented interactions, embracing the notions of direct manipulation (see Chapter 6), make more effective use of the computer's capacity for rapid display. In short, pointing and selecting in context is often more attractive than is typing or even speaking an English sentence.

8.7.2 Natural-language queries

Since general interaction is difficult to support, some designers have pursued a more limited goal of *natural-language queries* (NLQ) against relational databases. The *relational schema* contains attribute names and the database contains attribute values, both of which are helpful in disambiguating queries. A simulated query system was used to compare a subset of the structured SQL database facility to a natural-language system (Small and Weldon, 1983). The SQL simulation resulted in faster performance on a benchmark set of tasks. Similarly, a field trial with a real system, users, and queries pointed to the advantages of SQL over the natural-language alternative (Jarke et al., 1985). Believers in NLQ may claim that more research and system development is needed before that approach can be excluded, but
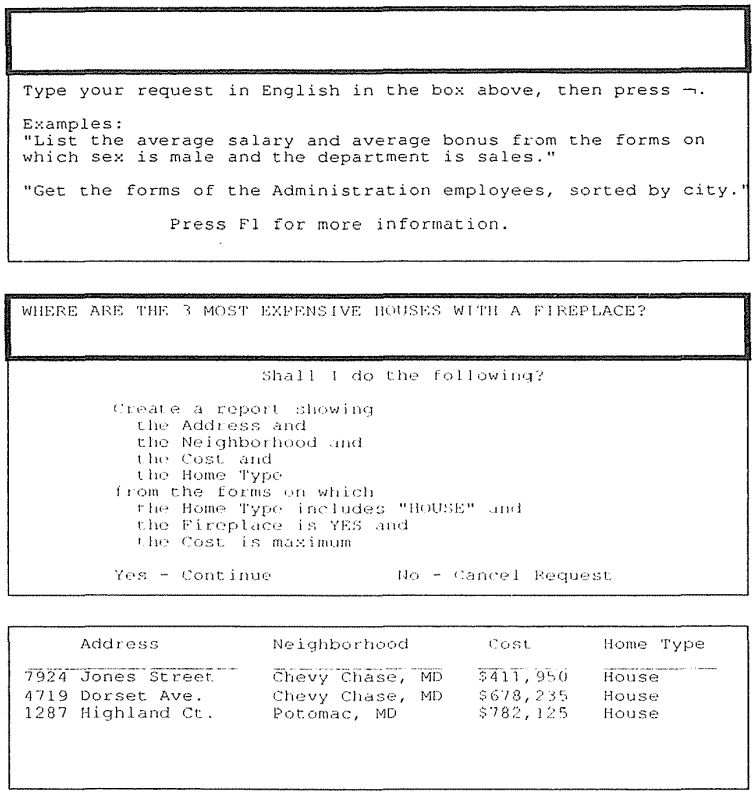
improvements in menus, command languages, and direct manipulation seem equally likely.

Supporters of NLQ can point with some pride at the modest success of the INTELLECT system, which had approximately 400 installations on large mainframe computers during the 1980s. The system's appeal has faded in recent years as users have turned to other approaches. Business executives, sales representatives, and other people use INTELLECT to search databases on a regular basis. Several innovative implementation ideas helped to make INTELLECT appealing. First, the parser used the contents of the database to parse queries; for example, the parser could determine that a query containing Cleveland referred to city locations, because Cleveland is an instance in the database. Second, the system administrator could conveniently include guidance for handling domain-specific requests, by indicating fields related to who, what, where, when, why, and how queries. Third, INTELLECT rephrased the user's query and displayed a response, such as PRINT THE CHECK NUMBERS WITH PAYEE = GEORGE BUSH. This structured response served as an educational aid, and users gravitated toward expressions that mimicked the style. Eventually, as users became more knowledgeable, they often used concise, commandlike expressions that they believed would be parsed successfully. Even the promoters of INTELLECT recognized that novice users who were unfamiliar with the task domain would have a difficult time, and that the ideal user might be a knowledgeable intermittent user.

A more successful product was Q&A from Symantec, which provided rapid, effective query interpretation and execution on IBM PCs (Fig. 8.4). The package made a positive impression, but few data have been collected about actual usage. The designers cited many instances of happy users of NLQ, and found practical applications in the users' daily work, but the popularity of the package seems to have been more closely tied to its word processor, database, and form-fillin facilities (Church and Rau, 1995). Q&A and most other NLQ packages are no longer sold. The dream of NLQ remains alive in some quarters, such as in Microsoft's Research Laboratories, where a talking parrot named Peedy is under development (Fig. 8.5).

8.7.3   Text-database searching

Text-database searching is a growing application for natural-language enthusiasts who have developed filters and parsers for queries expressed in natural language (Lewis and Jones, 1996). At one end of the spectrum is the full understanding of the meaning of a query and fulfillment of the users information needs. For example, in a legal application (Find cases of tenants who have sued landlords unsuccessfully for lack of heat), the system parses the text grammatically, provides synonyms from a thesaurus (renters for tenants), deals with singulars versus plurals, and handles other

```
┌────────────────────────────────────────────────────────────┐
│                                                            │
│                                                            │
└────────────────────────────────────────────────────────────┘

Type your request in English in the box above, then press ¬.

Examples:
"List the average salary and average bonus from the forms on
which sex is male and the department is sales."

"Get the forms of the Administration employees, sorted by city."

              Press F1 for more information.
```

```
┌────────────────────────────────────────────────────────────┐
│ WHERE ARE THE 3 MOST EXPENSIVE HOUSES WITH A FIREPLACE?     │
└────────────────────────────────────────────────────────────┘

                     Shall I do the following?

        Create a report showing
           the Address and
           the Neighborhood and
           the Cost and
           the Home Type
        from the forms on which
           the Home Type includes "HOUSE" and
           the Fireplace is YES and
           the Cost is maximum

        Yes - Continue              No - Cancel Request
```

```
      Address          Neighborhood     Cost       Home Type

7924 Jones Street    Chevy Chase, MD   $411,950    House
4719 Dorset Ave.     Chevy Chase, MD   $678,235    House
1287 Highland Ct.    Potomac, MD       $782,125    House
```

Figure 8.4

Q&A supported a natural-language front end for its database. Users could type a question in English and get the result of a structured database search. In this example, the user has typed a query, the system has responded with a verification, and then the system has generated the result. (Q&A was a product of Symantic Corp., Cupertino, CA.)

problems such as misspellings or foreign terms. Then, the analyzer separates the query into standard components—such as plaintiff, defendant, and cause—and finds all meaningfully related legal citations.

More realistic and typical scenarios are for parsers to eliminate noise words (for example, *the, of,* or *in*) provide stemming (plurals or alternate endings), and produce a relevance ranked list of documents based on term frequencies. These systems do not deal with negation, broader or narrow terms, and relationships (such as plaintiff sues defendants), but they can be effective with skilled users. A comparative-evaluation contest among information-retrieval programs that use natural-language strategies to select documents from a large collection continues to be extremely successful in promoting rapid progress (Harman, 1995). Many of the popular search tools on the World Wide Web (e.g., Lycos, Infoseek, AltaVista) use modest natural-

[Peedy, a parrot, is asleep on his perch]

User: Good morning, Peedy.   [Peedy rouses]

Peedy: Good morning.

User: Let's do a demo.   [Peedy stands up, smiles]

Peedy: Your wish is my command, what would you like to hear?

User: What have you got by Bonnie Raitt?

[Peedy waves in a stream of notes and grabs one]

Peedy: I have "The Bonnie Raitt Collection" from 1990.

User: Pick something from that.

Peedy: How about "Angel from Montgomery"?

User: Sounds good.   [Peedy drops note on pile]

Peedy: OK.

User: Play some rock after that.

Figure 8.5

Sample dialog from Microsoft Research exemplifying the goals for that company's natural-language system. (Used with permission of Microsoft Corp., Redmond, WA.)

language techniques, such as stemming, relevance ranking by word-frequency analyses, latent semantic indexing, and filtering of common words.

Another application with textual databases is extraction, in which a natural-language parser analyzes the stored text and creates a more structured format, such as a relational database. The advantage is that the parsing can be done once in advance to structure the entire database and to speed searches when users pose relational queries. Legal (Supreme Court decisions or state laws), medical (scientific journal articles or patient histories), and journalistic (Associated Press news stories or Wall Street Journal reports) texts have been used. This application is promising because even a modest increase in suitable retrievals is appreciated by users, and incorrect retrievals are tolerated better than are errors in natural-language interaction. Extraction is somewhat easier than is the task of writing a natural-language summary of a long document. Summaries must capture the essence of the content, and must convey it accurately in a compact manner. A variant task is to make categories of documents based on contents. For example, it would be useful to have an automated analysis of business news stories to separate out mergers, bankruptcies, and initial public offerings for companies in the electronics, pharmaceutical, or oil industries. The categorization task is appealing because a modest rate of errors would be tolerable (Church and Rau, 1995).

8.7.4   Natural-language text generation

Although the artificial-intelligence community often frowns on *natural-language text generation* (NLTG), this modest application does seem to be a worthy one (Fedder, 1990). It handles certain simple tasks, such as the preparation of structured weather reports (80-percent chance of light rain in northern suburbs by late Sunday afternoon) from complex mathematical models (Church and Rau, 1995). These reports can be sent out automatically, or even can be used to generate spoken reports available over the telephone in multiple languages.

More elaborate applications of NLTG include preparation of reports of medical laboratory or psychological tests. The computer generates not only readable reports (White-blood-cell count is 12,000), but also warnings (This value exceeds the normal range of 3000 to 8000 by 50 percent) or recommendations (Further examination for systemic infection is recommended). Still more involved scenarios for NLTG involve the creation of legal contracts, wills, or business proposals.

On the artistic side, computer generation of poems and even novels is a regular discussion point in literary circles. Although computer-generated combinations of randomly selected phrases can be provocative, it is still the creative work of the person who chose the set of possible words and decided which of the outputs to publish. This position parallels the custom of crediting the human photographer, rather than the camera or the subject matter of the photograph.

8.7.5   Adventure and educational games

Natural-language interaction techniques have enjoyed a notable and widespread success in a variety of adventure games. Users may indicate directions of movement or type commands, such as TAKE ALL OF THE KEYS, OPEN THE GATE, or DROP THE CAGE AND PICK UP THE SWORD. Part of the attraction of using natural-language interaction in this situation is that the system is unpredictable, and some exploration is necessary to discover the proper incantation. However, such games have largely disappeared from the market.

8.8   Practitioner's Summary

Command languages can be attractive when frequent use of a system is anticipated, users are knowledgeable about the task and interface concepts, screen space is at a premium, response time and display rates are slow, and numerous functions can be combined in a compact expression. Users have to learn the semantics and syntax, but they can initiate, rather than respond, and can rapidly specify actions involving several objects and options. Finally, a complex sequence of commands can be easily specified and stored for future use as a macro.

Designers should begin with a careful task analysis to determine what functions should be provided. Hierarchical strategies and congruent structures facilitate learning, problem solving, and human retention over time. Laying out the full set of commands on a single sheet of paper helps to show the structure to the designer and to the learner. Meaningful specific names aid learning and retention. Compact abbreviations constructed according to consistent rules facilitate retention and rapid performance for frequent users.

Command menus can be effective if rapid response to screen actions can be provided. Natural-language interaction and queries can be implemented partially, but their advantages are limited. Natural-language support has clearer niches in text searching, text generation, extraction, and games.

8.9 Researcher's Agenda

The benefits of structuring command languages based on hierarchy, congruence, consistency, and mnemonicity have been demonstrated in specific cases, but replication in varied situations should lead to a comprehensive cognitive model of command-language learning and use (Box 8.1). Novel input devices and high-speed, high-resolution displays offer new opportunities—such as command and pop-up menus—for breaking free from the traditional syntax of command languages.

Natural-language interaction still holds promise in certain applications, and empirical tests offer us a good chance to identify the appropriate niches and design strategies.

Box 8.1

Command Language Guidelines

- Create explicit model of objects and actions
- Choose meaningful, specific, distinctive names
- Try to achieve hierarchical structure
- Provide consistent structure
  (hierarchy, argument order, action-object)
- Support consistent abbreviation rules
  (prefer truncation to one letter)
- Offer frequent users the ability to create macros
- Consider command menus on high-speed displays
- Limit number of commands and ways of accomplishing a task

World Wide Web Resources   WWW

Some information on command languages but lots of activity on natural language translation, interaction, queries, and extraction.

http://www.aw.com/DTUI

References

Allen, James, *Natural Language Understanding* (Second Edition), Addison-Wesley, Reading, MA (1995).

Barnard, P. J., Hammond, N. V., Morton, J., Long, J. B., and Clark, I. A., Consistency and compatibility in human–computer dialogue, *International Journal of Man–Machine Studies*, 15, (1981), 87–134.

Benbasat, Izak and Wand, Yair, Command abbreviation behavior in human–computer interaction, *Communications of the ACM*, 27, 4 (April 1984), 376–383.

Black, J., and Moran, T., Learning and remembering command names, *Proc. Conference on Human Factors in Computer Systems*, ACM, Washington, D.C. (1982), 8–11.

Carroll, John M., Learning, using and designing command paradigms, *Human Learning*, 1, 1 (1982), 31–62.

Church, Kenneth W. and Rau, Lisa F., Commercial applications of natural language processing *Communications of the ACM*, 38, 11 (November 1995), 71–79.

Ehrenreich, S. L., and Porcu, Theodora, Abbreviations for automated systems: Teaching operators and rules. In Badre, Al, and Shneiderman, Ben, (Editors), *Directions in Human–Computer Interaction*, Ablex, Norwood, NJ (1982), 111–136.

Fedder, Lee., Recent approaches to natural language generation. In Diaper, D., Gilmore, D., Cockton, G., and Shackel, B. (Editors), *Human–Computer Interaction: Interact '90*, North-Holland, Amsterdam, The Netherlands (1990), 801–805.

Ford, W. Randolph, *Natural Language Processing by Computer—A New Approach*, Ph. D. Dissertation, Department of Psychology, Johns Hopkins University, Baltimore, MD (1981).

Grudin, Jonathan and Barnard, Phil, When does an abbreviation become a word and related questions, *Proc. CHI '85 Conference on Human Factors in Computer Systems*, ACM, New York (1985), 121–126.

Hanson, Stephen J., Kraut, Robert E., and Farber, James M., Interface design and multivariate analysis of Unix command use, *ACM Transactions on Office Information Systems*, 2, 1 (1984), 42–57.

Harman, Donna (Editor), *Proc. Third Text Retrieval Conference (TREC)*, Morgan Kaufmann, San Mateo, CA (1995).

Hauptmann, Alexander G. and Green, Bert F., A comparison of command, menu-selection and natural language computer programs, *Behaviour and Information Technology*, 2, 2 (1983), 163–178.

Jarke, Matthias, Turner, Jon A., Stohr, Edward A., Vassiliou, Yannis, White, Norman H., and Michielsen, Ken, A field evaluation of natural language for data retrieval, *IEEE Transactions on Software Engineering*, SE–11, 1 (January 1985), 97–113.

King, Margaret, Evaluating natural language processing systems, *Communications of the ACM*, 39, 1 (January 1996), 73–79.

Landauer, T. K., Calotti, K. M., and Hartwell, S., Natural command names and initial learning, *Communications of the ACM*, 26, 7 (July 1983), 495–503.

Ledgard, H., Whiteside, J. A., Singer, A., and Seymour, W., The natural language of interactive systems, *Communications of the ACM*, 23, (1980), 556–563.

Lewis, David and Jones, Karen Sparck, Natural language processing for information retrieval, *Communications of the ACM*, 39, 1 (January 1996), 92–101.

Napier, H. Albert, Lane, David, Batsell, Richard R., and Guadango, Norman S., Impact of a restricted natural language interface on ease of learning and productivity, *Communications of the ACM*, 32, 10 (October 1989), 1190–1198.

Norman, Donald, The trouble with Unix, *Datamation*, 27, (November 1981), 139–150.

Oviatt, Sharon, Interface techniques for minimizing disfluent input to spoken language systems, *Proc. CHI '94 Conference on Human Factors in Computing Systems*, ACM, New York (1994), 205–210.

Pausch, Randy and Leatherby, James H., An empirical study: Adding voice input to a graphical editor, *Journal of the American Voice Input/Output Society*, 9, 2 (July 1991), 55–66.

Rosenberg, Jarrett, Evaluating the suggestiveness of command names, *Behaviour and Information Technology*, 1, (1982), 371–400.

Scapin, Dominique L., Computer commands labeled by users versus imposed commands and the effect of structuring rules on recall, *Proc. Conference on Human Factors in Computer Systems*, ACM, Washington, D.C. (1982), 17–19.

Schneider, M. L., Ergonomic considerations in the design of text editors. In Vassiliou, Y. (Editor), *Human Factors and Interactive Computer Systems*, Ablex, Norwood, NJ (1984), 141–161.

Schneider, M. L., Hirsh-Pasek, K., and Nudelman, S., An experimental evaluation of delimiters in a command language syntax, *International Journal of Man–Machine Studies*, 20, 6 (June 1984), 521–536.

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

Small, Duane and Weldon, Linda, An experimental comparison of natural and structured query languages, *Human Factors*, 25, (1983), 253–263.
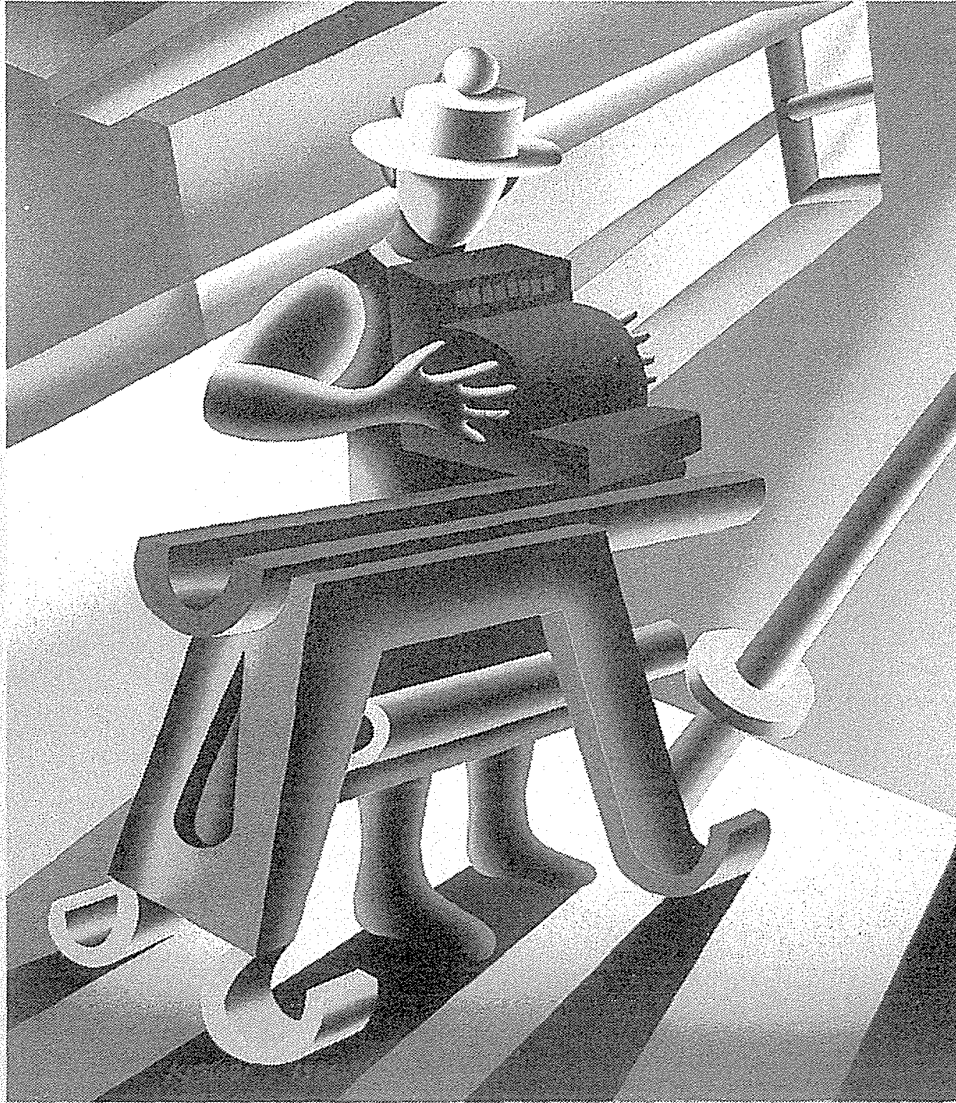
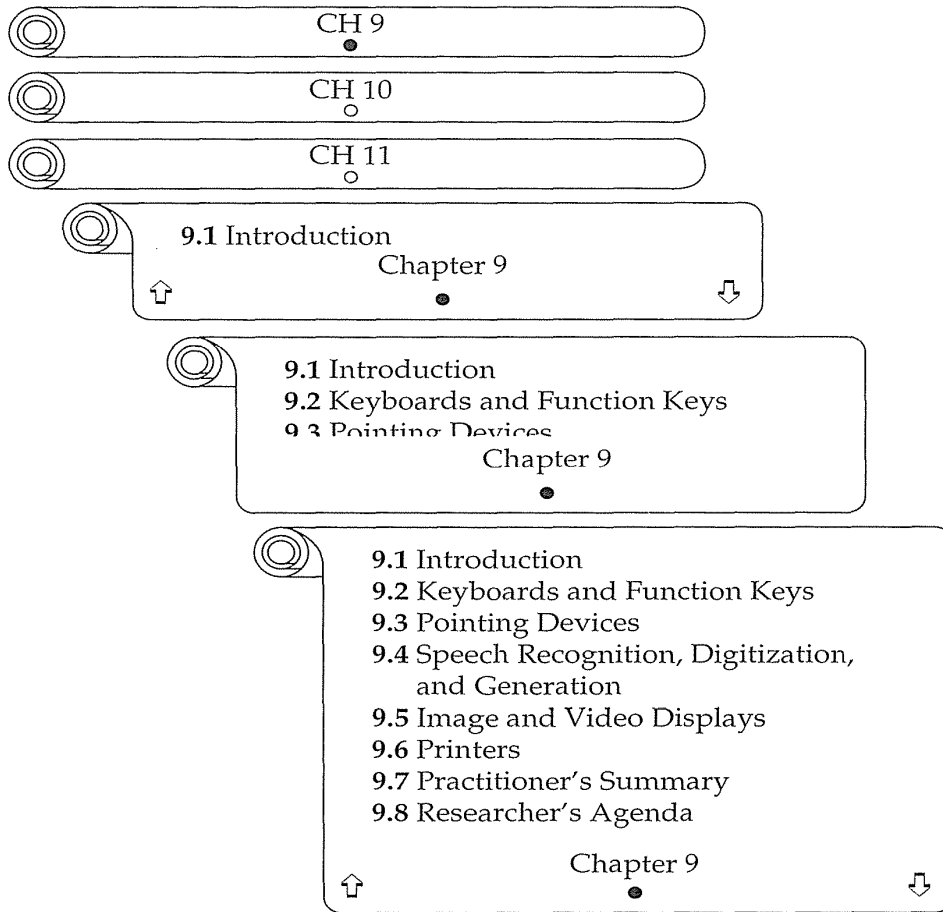Mark Kostabi, *Industrial Interior,* 1996

9

Interaction Devices

The wheel is an extension of the foot,
the book is an extension of the eye,
clothing, an extension of the skin,
electric circuitry an extension of the central nervous system.

Marshall McLuhan and Quentin Fiore,
The Medium Is the Massage, 1967

CH 9

CH 10

CH 11

9.1 Introduction
Chapter 9

9.1 Introduction
9.2 Keyboards and Function Keys
9 3 Pointing Devices
Chapter 9

9.1 Introduction
9.2 Keyboards and Function Keys
9.3 Pointing Devices
9.4 Speech Recognition, Digitization, and Generation
9.5 Image and Video Displays
9.6 Printers
9.7 Practitioner's Summary
9.8 Researcher's Agenda

Chapter 9

9.1 Introduction

The remarkable progress since 1960 in computer-processor speeds and storage capabilities is matched by improvements in many input-output devices. Ten-character-per-second Teletypes have been replaced by high-speed megapixel graphical displays for output, and the 100-year-old keyboard is giving way to rapid and high-precision pointing devices for carrying out user actions. Although the common Sholes keyboard layout is likely to remain the primary device for text input, pointing devices increasingly free users from keyboards for many tasks. The future of computing is likely to include gestural input, two-handed input, three-dimensional pointing, more voice input–output, wearable devices, and whole-body involvement for some input and output tasks.

The increased concern for human factors has led to hundreds of new devices and variants of the old devices. Novel keyboards with tilted and curved profiles to lessen repetitive-strain injuries and touchscreen or stylus replacements testify to the vital importance of textual input, even in the age of GUIs. Pointing devices such as the mouse, touchscreen, stylus, and track-ball have gone through hundreds of refinements to accommodate varied users and to squeeze out another five percent improvement in performance. The still-improving speech recognizers are joined by more mundane but widely used speech store-and-forward technologies with increased emphasis on telephone-based applications. Proponents of eyetrackers, datagloves, and force-feedback devices are trying to expand from their niches.

Color displays for desktops and most laptops are standard, but monochrome displays (especially flat-plate liquid-crystal panels) continue to proliferate in small and large formats. Compact digital cameras with instant viewing on small LCDs are creating a steadily growing success story. Low-cost printers, even with color, are widely available. Innovative input devices, sensors, and effectors, and integration of computers into the physical environment, open the door to new applications (Sherr, 1988; Greenstein and Arnaut, 1988; Foley et al., 1990; Card et al., 1991; Jacob et al., 1993).

9.2 Keyboards and Function Keys

The primary mode of textual data entry is still the keyboard. This often-criticized device is impressive in its success. Hundreds of millions of people have managed to use keyboards with speeds of up to 15 keystrokes per second (approximately 150 words per minute), although rates for beginners are less than one keystroke per second, and rates for average office workers are five keystrokes per second (approximately 50 words per minute). Contemporary keyboards generally permit only one keypress at a time, although dual keypresses (SHIFT plus a letter) are used to produce capitals and special functions (CTRL or ALT plus a letter).

More rapid data entry can be accomplished by chord keyboards that allow several keys to be pressed simultaneously to represent several characters or a word. Courtroom recorders regularly use chord keyboards serenely to enter the full text of spoken arguments, reaching rates of up to 300 words per minute. This feat requires months of training and frequent use to retain the complex pattern of chordpresses. The piano keyboard is an impressive data-entry device that allows several fingerpresses at once and is responsive to different pressures and durations. It seems that there is potential for higher rates of data entry than is possible with the current computer keyboards.

Keyboard size and packaging also influence user satisfaction and usability. Large keyboards with many keys give an impression of professionalism and complexity, but may threaten novice users. Small keyboards seem lacking in power to some users, but their compact size is an attraction to others. A thin profile (20 to 40 millimeters thick) allows users to rest the keyboards on their laps easily and permits a comfortable hand position when the keyboard is on a desk. Adjustable keyboards that tilt forward or back, or that split in the middle to reduce stressful ulnar abduction and pronation, are currently popular.

9.2.1   Keyboard layouts

The Smithsonian Institution's National Museum of American History in Washington, D.C. has a remarkable exhibit on the development of the typewriter. During the middle of the nineteenth century, hundreds of attempts were made to build typewriters, with a stunning variety of positions for the paper, mechanisms for producing a character, and layouts for the keys. By the 1870s, Christopher Latham Sholes's design was becoming successful because of a good mechanical design and a clever placement of the letters that slowed down the users enough that key jamming was infrequent. This *QWERTY layout* put frequently used letter pairs far apart, thereby increasing finger travel distances.

Sholes's success led to such widespread standardization that, a century later, almost all English-language keyboards use the QWERTY layout (Fig. 9.1). The development of electronic keyboards eliminated the mechanical problems and led many twentieth-century inventors to propose alternative layouts to reduce finger travel distances (Montgomery, 1982; Kroemer, 1993).

Figure 9.1

QWERTY keyboard from the Macintosh, with function keys, numeric keypad, separate cursor-control keys, and special functions.
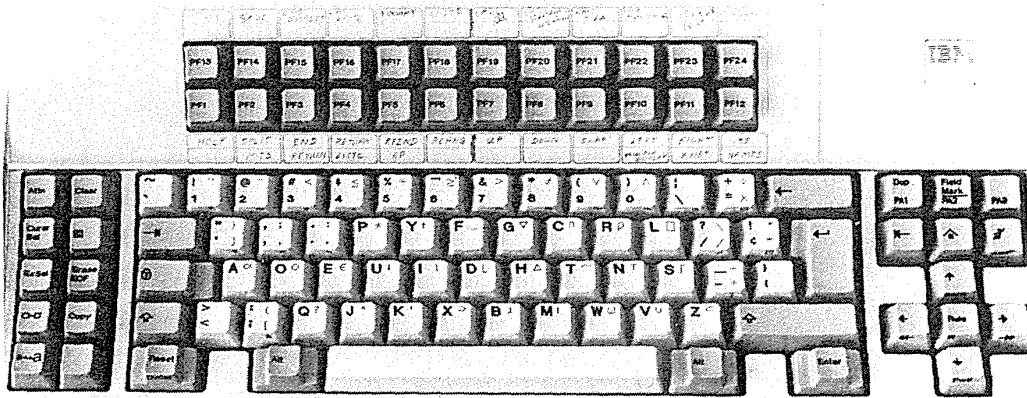
Figure 9.2

Dvorak layout on an IBM keyboard with function keys, separate cursor-control keys, and special functions. The keycaps also show the APL character set.

The *Dvorak layout* (Fig. 9.2), developed in the 1920s, supposedly reduces finger travel distances by at least one order of magnitude, thereby increasing the typing rate of expert typists from about 150 words per minute to more than 200 words per minute, while reducing errors (Potosnak, 1988).

Acceptance of the Dvorak design has been slow despite the dedicated work of devotees. Those people who have tried the keyboard report that it takes about 1 week of regular typing to make the switch, but most users have been unwilling to invest this much effort. We are confronted with an interesting example of how even documented improvements are hard to disseminate because the perceived benefit of change does not outweigh the effort.

A third keyboard layout of some interest is the *ABCDE style* that has the 26 letters of the alphabet laid out in alphabetical order. The rationale here is that nontypists will find it easier to locate the keys. A few data-entry terminals for numeric and alphabetic codes still use this style. The widespread availability of QWERTY keyboards has made typing a more common skill and has reduced the importance of the ABCDE style. Our study and those of other researchers have shown no advantage for the ABCDE style; users with little QWERTY experience are eager to acquire this expertise and often resent having to use an ABCDE style.

Beyond the letters, many debates rage about the placement of additional keys. The early IBM PC keyboard was widely criticized because of the placement of a few keys, such as a backslash key where most typists expected to find the SHIFT key, and the placement of several special characters near the ENTER
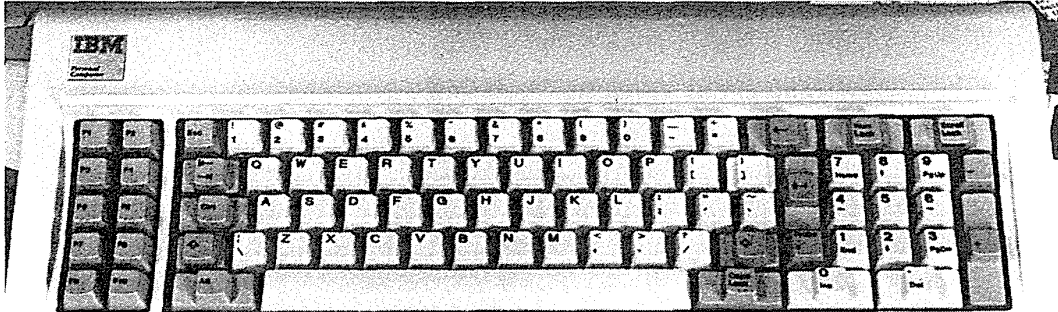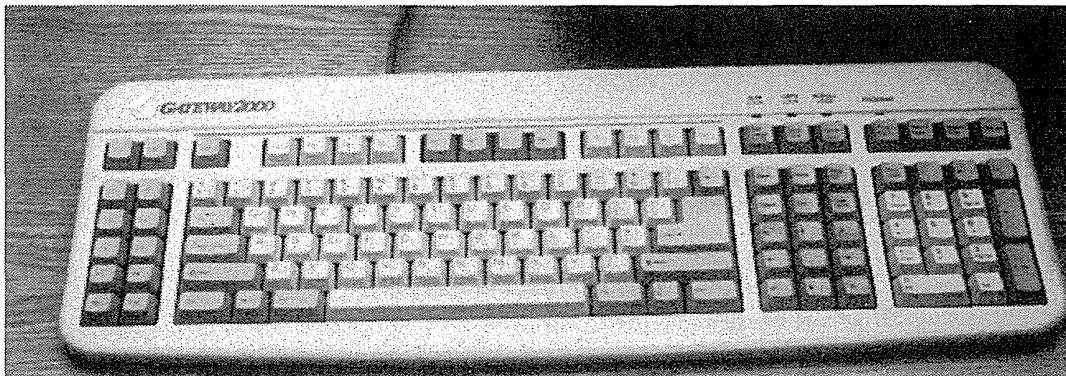
Figure 9.3

Early IBM PC keyboard with 10 function keys on the left, numeric keypad on the right, and cursor-control keys embedded in the numeric keypad.

key (Fig. 9.3). Later versions relocated the offending keys, to the acclaim of critics. Other improvements included a larger ENTER key and LEDs to signal the status of the CAPS LOCK, NUM LOCK, and SCROLL LOCK keys (Fig. 9.4). Even on laptop or notebook computers, keyboards are fullsize, but some pocket computers used a greatly reduced keyboard (Fig. 9.5).

Number pads are another source of controversy. Telephones have the 1–2–3 keys on the top row, but calculators place the 7–8–9 keys on the top row. Studies have shown a slight advantage for the telephone layout, but most computer keyboards use the calculator layout.

Some researchers have recognized that the wrist and hand placement required for standard keyboards is awkward. Redesigned keyboards that

Figure 9.4

Full-size keyboard. (Produced by Gateway 2000 Corp.)

Figure 9.5

Pocket computers have reduced-sized keyboards. Many users type with one finger on each hand.

separated the keys for the left and right hands by 9.5 centimeters, had an opening angle of 25 degrees with an inclination of 10 degrees, and offered large areas for forearm—wrist support led to lower reported tension, better posture, and higher preference scores (Nakaseko et al., 1985). However, separated keyboards have the disadvantage that visual scanning is disrupted, so various geometries have been tried with split and tilted keyboards (for example, the Microsoft Natural; Fig. 9.6), but empirical verification of benefits to typing speed, accuracy, or reduced repetitive strain injury is elusive.

9.2.2 Keys

Modern electronic keyboards use 1/2-inch-square keys (12 mm square) with about a 1/4-inch space (6 mm square) between keys. This design has been refined carefully and tested thoroughly in research laboratories and the marketplace. The keys have a slightly concave surface for good contact with fingertips, and a matte finish to reduce both reflective glare and the chance of finger slips. The keypresses require a 40- to 125-gram force and a displacement of 3 to 5 millimeters. The force and displacement have been shown to produce rapid typing with low error rates while providing suitable feedback to users. As user experience increases and the chance of a misplaced finger is reduced, the force and displacement can be lowered.

An important element in key design is the profile of force displacement. When the key has been depressed far enough to send a signal, the key gives way and emits a click. The tactile and audible feedback is extremely important in touch typing. For these reasons, membrane keyboards that use a non-moving touch-sensitive surface are unacceptable for touch typing. However,

Figure 9.6

Microsoft's Natural keyboard. The curved layout and adjustable rests underneath
help to reduce repetitive-strain injuries.

they are durable and therefore effective for public installations at museums
or amusement parks.

Certain keys, such as the space bar, ENTER key, SHIFT key, or CTRL key,
should be larger than others to allow easy, reliable access. Other keys, such as
the CAPS LOCK or NUM LOCK should have a clear indication of their state,
such as by physical locking in a lowered position or by an embedded light.
Key labels should be large enough to read, meaningful, and permanent. Dis-
crete color coding of keys helps to make a pleasing, informative layout. A
further design principle is that some of the home keys—F and J in the
QWERTY layout—may have a deeper concavity or a small raised dot to reas-
sure users that their fingers are placed properly.

9.2.3   Function keys

Many keyboards contain a set of additional *function keys* for special functions or
programmed functions. These keys are often labeled F1 ... F10 or PF1 ... PF24.
Users must remember the functions, learn about them from the screen, or con-
sult an attachable plastic template, but some keys have meaningful labels, such
as CUT, COPY, or PASTE. This strategy attempts to reduce user keystrokes by
replacing a command name with a single keystroke. Most function-key strate-
gies do not require users to press the ENTER key to invoke the function.

Function keys can reduce keystroke numbers and errors, thereby speed-
ing work for novice users who are poor typists and for expert users who

readily recall the purpose of each function key. Unfortunately, some systems confuse users with inconsistent key use. For example, the HELP key may be F1, F9, or F12, depending on the system.

The placement of function keys is important if the task requires users to go from typing to using function keys. The greater the distance of the function keys from the home position on the keyboard, the more severe the problem. Some users would rather type six or eight characters than remove their fingers from the home position. Layout of function keys also influences ease of use. A 3 by 4 layout of 12 keys is helpful, because users quickly learn functions by the keys' placement on the upper-left or lower-right. A 1 by 12 layout only has two anchors and leads to slower and more error-prone selection of middle keys. A small gap between the sixth and seventh keys could aid users by grouping keys. A 2 by 5 layout is a reasonable intermediate style.

Function keys are sometimes built in to the display-screen bezel so that they are close to displayed labels—a popular technique with bank machines. This position supports novices who need labels, but it still requires hands to stray from the home position. Lights can be built into and next to function keys to indicate availability or ON-OFF status.

If all work can be done with labeled function keys, as on some CAD systems, this solution is appealing. WordPerfect became a worldwide success partly because all actions are initiated by function keys (plus CTRL, ALT, and SHIFT) and refined by onscreen menus. Some WordPerfect devotees refuse to move to pull-down menus, since their proficiency with the keys enables them to beat many mouse users. However, most word-processor users have now adopted more graphic interfaces, including many with toolbars with icons for frequent tasks. Frequent movements between the home position on the keyboard and the mouse or distant function keys can be disruptive. An alternative strategy is to use nearby CTRL or ALT keys plus a letter to invoke a function. This approach has some mnemonic value, keeps hands on the home keys, and reduces the need for extra keys.

9.2.4   Cursor movement keys

A special category of function keys is the *cursor-movement keys*. There are usually four keys—up, down, left, and right. Some keyboards have eight keys to simplify diagonal movements. The placement of the cursor-movement keys is important in facilitating rapid and error-free use. The best layouts place the keys in their natural positions (Fig. 9.7a–d), but designers have attempted several variations (Fig. 9.7e–g). The increasingly popular inverted-T arrangement (Fig. 9.7a) allows users to place their middle three fingers in a way that reduces hand and finger movement. The cross-arrangement (Fig. 9.7b) is a better choice for novice users than the linear arrangement (Fig. 9.7e) or the box arrangement (Fig. 9.7f).
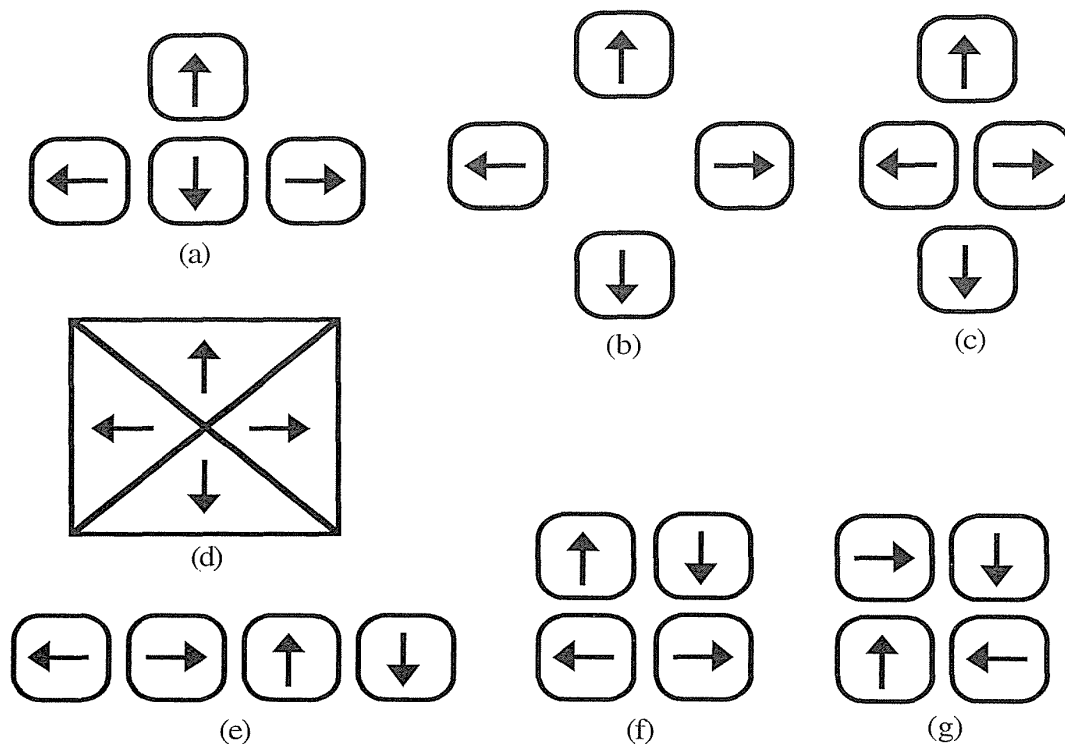
Figure 9.7

Seven styles of key layout for arrow keys. These layouts are only a subset of what is commercially available. (a–d) Key layouts that are compatible with the arrow directions. (e–g) Incompatible layouts that may result in slower performance and higher error rates.

Cursor-movement keys often have a *typamatic (auto-repeat) feature;* that is, repetition occurs automatically with continued depression. This feature is widely appreciated and may improve performance, especially if users can control the rate to accommodate their preferences (important for users who are very young, elderly, or handicapped).

Cursor-movement keys have become more important with the increased use of form-fillin and direct manipulation. Additional cursor movements might be performed by the TAB key for larger jumps, the HOME key to go to the top left, or the END key to go to the bottom right of the display. Other accelerators are popular, such as CTRL with up, down, left, or right key-presses, to jump a word or a paragraph. Cursor-movement keys can be used

to select items in a menu or on a display, but more rapid pointing at displays than can be provided by cursor-movement keys is often desired.

9.3 Pointing Devices

With complex information displays—such as those used in air-traffic control, word processing, and CAD—it is often convenient to point at and select items. This direct-manipulation approach is attractive because the users can avoid learning commands, reduce the chance of typographic errors on a keyboard, and keep their attention on the display. The results are often faster performance, fewer errors, easier learning, and higher satisfaction. The diversity of tasks and variety of devices plus strategies for using them create a rich design space (Buxton, 1985; Card et al., 1991). Physical device attributes (rotation or linear movement), dimensionality of movement (1, 2, 3, ...), and positioning (relative or absolute) are useful ways of categorizing devices; here, we discuss the tasks and degree of directness.

9.3.1 Pointing tasks

Pointing devices are applicable in six types of interaction tasks (Foley et al., 1984):

1. *Select*   The user chooses from a set of items. This technique is used for traditional menu selection, identification of a file in a directory, or marking of a part in an automobile design.
2. *Position*   The user chooses a point in a one-, two-, three-, or higher-dimensional space. Positioning may be used to create a drawing, to place a new window, or to drag a block of text in a figure.
3. *Orient*   The user chooses a direction in a two-, three-, or higher-dimensional space. The direction may simply rotate a symbol on the screen, indicate a direction of motion for a space ship, or control the operation of a robot arm.
4. *Path*   The user rapidly performs a series of position and orient operations. The path may be realized as a curving line in a drawing program, the instructions for a cloth-cutting machine, or the route on a map.
5. *Quantify*   The user specifies a numeric value. The quantify task is usually a one-dimensional selection of integer or real values to set parameters, such as the page number in a document, the velocity of a ship, or the amplitude of a sound.

6. *Text*   The user enters, moves, and edits text in a two-dimensional space. The pointing device indicates the location of an insertion, deletion, or change. Beyond the simple manipulation of the text are more elaborate tasks, such as centering, margin setting, font sizes, highlighting (bold-face or underscore), and page layout.

It is possible to perform all these tasks with a keyboard by typing: numbers or letters to select, integer coordinates to position, a number representing an angle to point, a number to quantify, and cursor-control commands to move about in text. In the past, the keyboard was used to perform all these tasks, but now users employ pointing devices to perform these tasks more rapidly and with fewer errors. Even in modern GUIs, however, some tasks are invoked so frequently that special keys may be appropriate, such as a HELP key or CTRL-C to copy a marked item.

Pointing devices can be grouped into those that offer (1) *direct control* on the screen surface, such as the lightpen, touchscreen, and stylus, and (2) *indirect control* away from the screen surface, such as mouse, trackball, joystick, graphics tablet, and touchpad. Within each category are many variations, and novel designs emerge frequently.
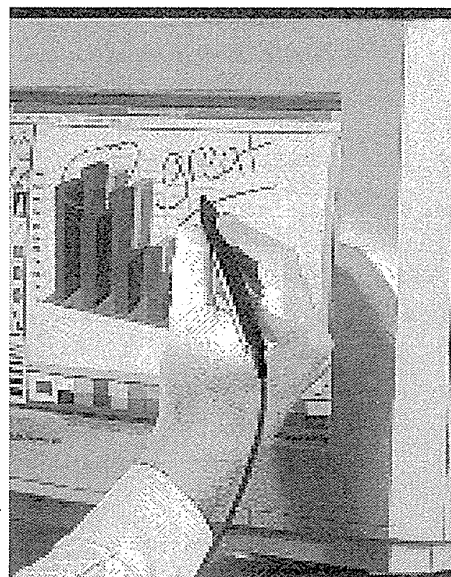
9.3.2   Direct-control pointing devices

The *lightpen* was an early device that enabled users to point to a spot on a screen and to perform a select, position, or other task (Fig. 9.8). In fact, the lightpen could be used to perform all six tasks. The lightpen was attractive
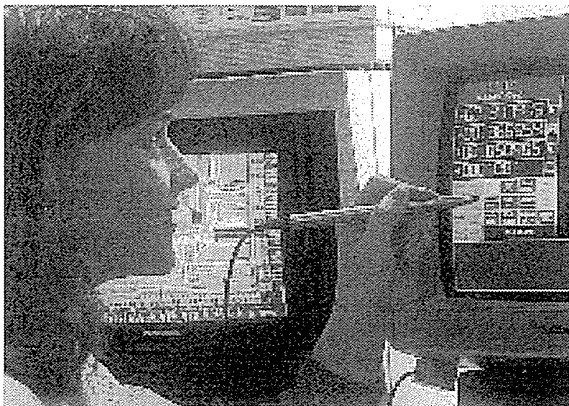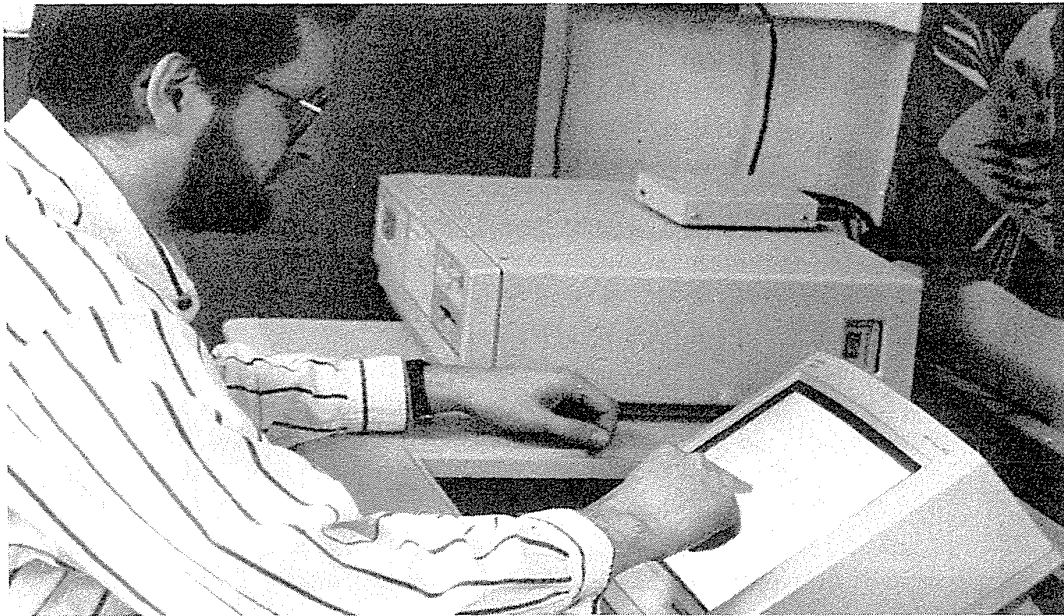

Figure 9.8

Lightpens. Users can point directly to a spot on the screen with these input devices.

because it allowed the user to gain direct control by pointing to a spot on the display, as opposed to having the indirect control provided by a graphics tablet, joystick, or mouse. Most lightpens incorporate a button for the user to press when the cursor is resting on the desired spot on the screen. Lightpens vary in thickness, length, weight, and shape (the lightgun, with a trigger, was one variation), and in position of buttons. Unfortunately, direct control on an upright screen can cause arm fatigue. The lightpen had three further disadvantages: users' hands obscured part of the screen, users had to remove their hands from the keyboard, and users had to pick up the lightpen.

Some of these disadvantages are overcome by the *touchscreen*, which does not require picking up some device, but instead allows the user to make direct-control touches on the screen with a finger (Fig. 9.9) (Shneiderman, 1991). Early touchscreen designs were rightly criticized for causing fatigue, hand obscuring the screen, hand off keyboard, imprecise pointing, and the eventual smudging of the display. Some touchscreen implementations had a further problem: The software accepted the touch immediately (*land-on strategy*), denying users the opportunity to verify the correctness of the selected spot, which they could do with lightpens. These early designs were based on physical pressure, impact, or interruption of a grid of infrared beams.

Figure 9.9

Touchscreen. The user needs only to point with a finger to make a selection. High-precision touchscreens increase the range of possible applications, especially if they are mounted in a position that is convenient for pointing and reading (30 to 45 degrees from the horizontal).

Newer designs have dramatically improved touchscreens to permit high precision (Sears and Shneiderman, 1991). The resistive, capacitive, or surface acoustic-wave hardware can provide 1024 × 1024 sensitivity, and the *lift-off strategy* enables users to point at a single pixel. The lift-off strategy has three steps. The users touch the surface, and then see a cursor that they can drag around on the display; when the users are satisfied with the position, they lift their fingers off the display to activate. The availability of high-precision touchscreens has opened the doors to many applications (Sears et al., 1992).

Refinements to touchscreens can be expected as they are integrated into applications directed at novice users, in which the keyboard can be eliminated and touch is the only interface mechanism. Touchscreens are valued by designers of public-access systems because there are no moving parts, durability in high-use environments is good (touchscreens are the only input devices that have survived at EPCOT), and the price is relatively low. Touchscreens have found a home in building-management, air-traffic–control, medical, and military systems. In these systems, space is at a premium, rugged design with no moving parts is appreciated, and users can be guided through a complex activity.

Touchscreens can produce varied displays to suit the task. Form fillin or menu selection works naturally by touchscreen, as does typing on a touchscreen keyboard. In our studies with keyboards that were 7 and 25 centimeters wide, users could, with some practice, type from 20 to 30 words per minute, respectively (Sears et al., 1993). Studies of touchscreen use for panning graphical displays show that users are more rapid and accurate when pushing the background, rather than shifting the viewpoint (Johnson, 1995).

As touchscreens are fabricated integrally with display surfaces, cost is likely to drop and parallax problems from an attached glass panel will decrease. A touchable surface on a flat-plate LCD enables construction of novel car-borne navigation displays and museum placards that contain extensive information. Every office-door nameplate, refrigerator, camera, TV, or appliance could have helpful explanations, information resources, or complete user manuals constantly available.

Palmtop computers make it natural to have pointing on the LCD surface, which is held in the hand, placed on a desk, or rested on the lap. Handwriting recognition, selection from a keyboard displayed on the surface, or selection from menus and forms permits simple and rapid data entry. The *stylus* is attractive to designers because it is familiar and comfortable for users, and because it permits high precision with good control to limit inadvertent selections. Users can guide the stylus tip to the desired location while keeping the critical sections of the display in view. These advantages over touchscreens are balanced against the need to pick up and put down a stylus.

Compact lightweight machines, such as the Apple Newton or MessagePad, Sharp Zaurus, US Robotics Pilot (see Fig. 1.6), and others are grow-

ing in popularity. The market should be large if the price can be brought down while screen resolution and readability are improved.

Alternatives to keyboard entry are gaining popularity with touchscreen or stylus entry on virtual keyboards, but these mechanisms have data-entry rates of only 20 to 30 words per minute (Sears et al., 1993). Novel data entry based on pie-menu gestures (Venolia and Neiberg, 1994) and handwriting recognition (Frankish et al., 1995) are competing with ever-faster pull-down–menu and direct-manipulation strategies.

Recognition of handwritten block or script characters has been supplemented by word-pattern recognition against a stored database of 10,000 or more words. Contextual clues and stroke speed plus direction can enhance recognition rates. The Apple Newton brought handwriting input and recognition to a wide market, but high error rates were troubling to many users. For some languages, such as Japanese or Chinese, handwriting input and recognition may dramatically increase the user population.

9.3.3 Indirect-control pointing devices

Indirect pointing devices eliminate the hand-fatigue and hand-obscuring-the-screen problems, but must overcome the problem of indirection. As they do with the lightpen, the off-keyboard hand position and pick-up problems remain. Also, indirect-control devices require more cognitive processing and hand–eye coordination to bring the onscreen cursor to the desired target.

The *mouse* is appealing because the hand rests in a comfortable position, buttons on the mouse are easily pressed, even long motions can be rapid, and positioning can be precise (Fig. 9.10). However, the user must grab the
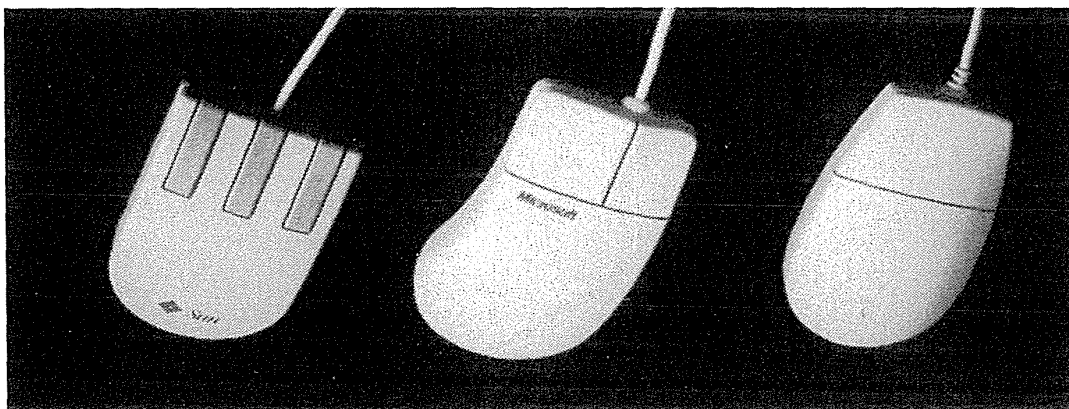
· **Figure 9.10**

Three versions of the mouse: three buttons (Sun Microsystems), two buttons (Microsoft), and one button (Apple Macintosh).

mouse to begin work, desk space is consumed, the mouse wire can be distracting, pickup and replace actions are necessary for long motions, and some practice (5 to 50 minutes) is required to develop skill. The variety of mouse technologies (physical, optical, or acoustic), number of buttons, placement of the sensor, weight, and size indicate that designers and users have yet to settle on one alternative. Personal preferences and the variety of tasks leave room for lively competition. Wireless mice that allow pointing from 5 to 30 feet away from the display without a mousepad are gaining acceptance for lecturing situations and for home entertainment in living-room environments. Infrared technologies refined from television remote controllers or more sophisticated gyroscopic designs are being promoted.

The *trackball* has sometimes been described as an upside-down mouse. It is usually implemented as a rotating ball 1 to 15 centimeters in diameter that moves a cursor on the screen as it is moved. The trackball is firmly mounted in a desk or a solid box to allow the user to hit the ball vigorously and to make it spin. The trackball has been the preferred device in the high-stress world of air-traffic control and in some video games. Small trackballs make a convenient pointing device when installed in portable laptop computers (Fig. 9.11).

The *joystick*, whose long history began in aircraft-control devices (Fig. 9.12), now has dozens of computer versions with varying stick lengths and thicknesses, displacement forces and distances, buttons or triggers, anchoring strategies for bases, and placement relative to the keyboard and screen.

Figure 9.11

Small trackball embedded in a laptop computer.

Figure 9.12

A joystick. This input device makes it easy for users to move a cursor around on the screen rapidly, but precise or drawing actions are difficult.

Joysticks are appealing for tracking purposes (i.e., to follow a moving object on a screen), partly because of the relatively small displacements needed to move a cursor and the ease of direction changes. The trackpoint is a small isometric joystick embedded in laptop keyboards that has a rubber tip to facilitate finger contact (Fig. 9.13). With modest practice, users can be quick and accurate while keeping their fingers on the keyboard home position.
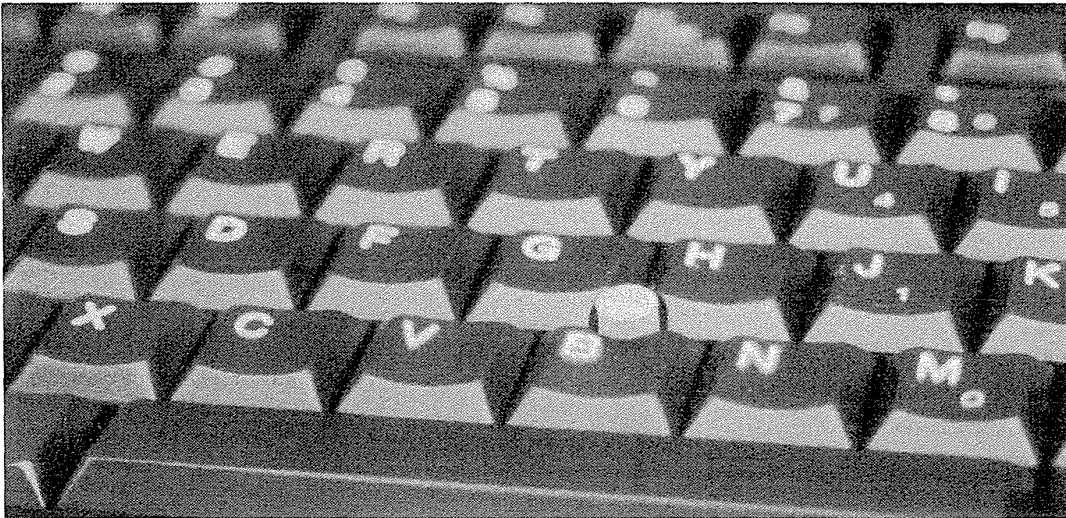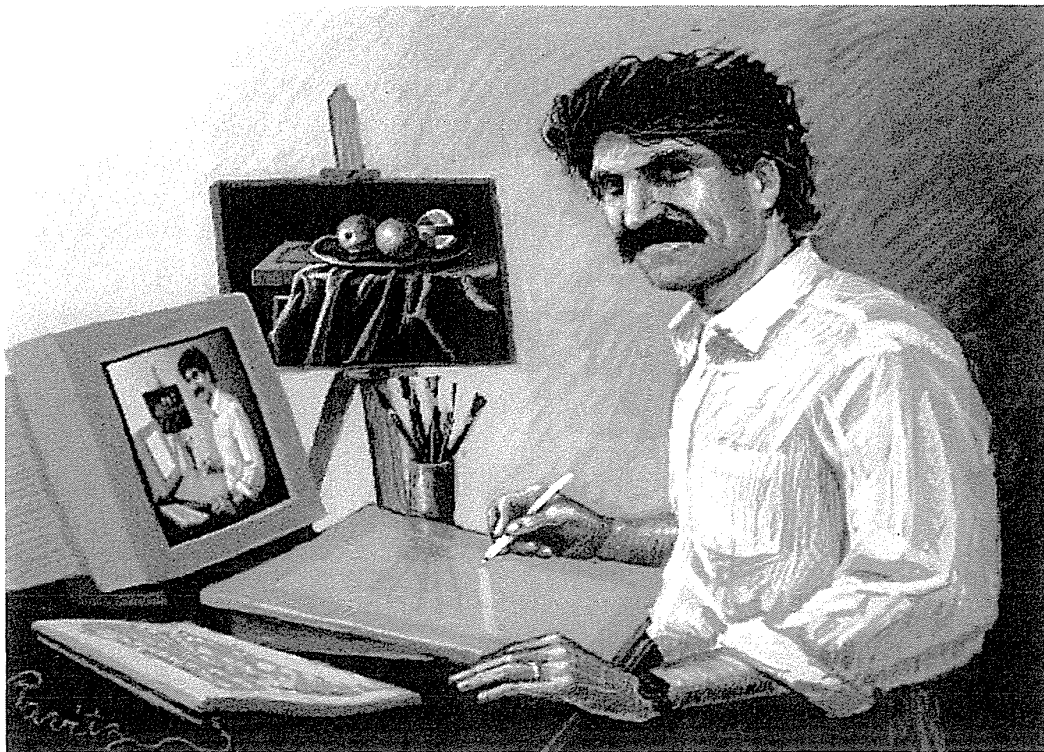

Figure 9.13

The trackpoint. This device is a small isometric joystick mounted between the G and H keys.

The *graphics tablet* is a touch-sensitive surface separate from the screen, usually laid flat on the table (Fig. 9.14) or in the user's lap. This separation allows for a comfortable hand position and keeps the users' hands off the screen. Furthermore, the graphics tablet permits a surface even larger than the screen to be covered with printing to indicate available choices, thereby providing guidance to novice users and preserving valuable screen space. Limited data entry can be done with the graphics tablet. The graphics tablet can be operated by placement of a finger, pencil, puck, or stylus, using acoustic, electronic, or contact position sensing.

A *touchpad* (5- by 8-centimeters touchable surface) built in near the keyboard offers the convenience and precision of a touchscreen while keeping the user's hand off the display surface. Users can make quick movements for long-distance traversals, and can gently rock their fingers for precise positioning, before lifting off. The lack of moving parts and the thin profile make touchpads appealing for portable devices.

Figure 9.14

An all-electronic painting created by Larry Ravitz using Adobe PhotoShop and a Wacom tablet. The Wacom stylus and graphics tablet allow the precise pointing and accurate control that artists need. (Photograph courtesy of Larry Ravitz, Takoma Park, MD.)

Among these indirect pointing devices, the mouse has been the basis for the greatest success story. Given its rapid high-precision pointing and a comfortable hand position, the modest training period is only a small impediment to its use. Most desktop computer systems offer a mouse, but the battle for the laptop rages.

9.3.4 Comparisons of pointing devices

Each pointing concept has its enthusiasts and detractors, motivated by commercial interests, by personal preference, and increasingly by empirical evidence. Human-factors variables of interest are speed of motion for short and long distances, accuracy of positioning, error rates, learning time, and user satisfaction. Other variables are cost, durability, space requirements, weight, left- versus right-hand use, likelihood to cause repetitive-strain injury, and compatibility with other systems.

In early studies, direct pointing devices such as the lightpen or touchscreen were often the fastest but the least accurate devices (Stammers and Bird, 1980; Haller et al., 1984). The speed appears to accrue from the directness of pointing, and the inaccuracy from problems with feedback, physical design, and use strategies. New strategies such as lift-off and greater precision in the devices have made it feasible to build high-precision touchscreens, graphics tablets, and pens.

Indirect pointing devices have been the cause of much controversy. The graphics tablet is appealing when the user can remain with the device for long periods without switching to a keyboard. Pens accompanying graphics tablets allow a high degree of control that is appreciated by artists using drawing programs. The Wacom tablet with its wireless pen allows freedom and control (Fig. 9.14). The mouse was found to be faster than the isometric joystick (English et al., 1967; Card et al., 1978; Rutledge and Selker, 1990) due to tremors in finger motion during fine finger movements (Mithal and Douglas, 1996).

The usual belief is that pointing devices are faster than keyboard controls, such as cursor-movement keys, but this assertion depends on the task. When a few (two to 10) targets are on the screen and the cursor can be made to jump from one target to the next, then using the cursor jump keys can be faster than using pointing devices (Fig. 9.15) (Ewing et al., 1986). For tasks that mix typing and pointing, cursor keys have also been shown to be faster than and preferred to the mouse (Karat et al., 1984). Since muscular strain is low for cursor keys (Haider et al., 1982), they should be considered for this special case. This result is supported by Card et al. (1978), who reported that, for short distances, the cursor keys were faster than the mouse (Fig. 9.16). The positioning time increases rapidly with distance for cursor keys, but only slightly for the mouse or trackball.

Figure 9.15

Path completion time for arrow-jump and mouse as a function of average target distance of the traversed path (Ewing et al., 1986). Long-distance targets were farther away from the start point than were short-distance targets. The arrow-jump strategy was faster, since a single keypress produced a jump to the target, whereas mouse users had to move the cursor across the screen.
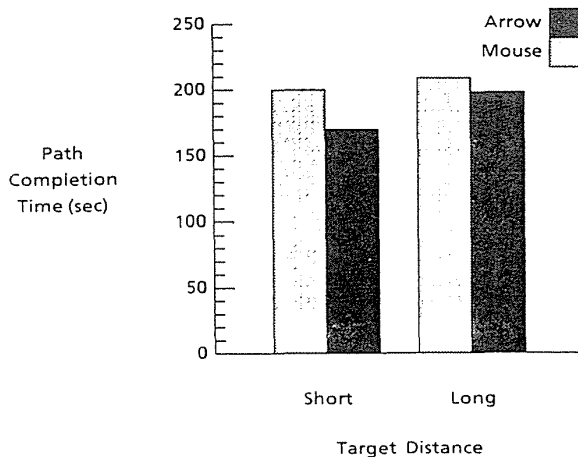
In summary, much work remains to sort out the task and individual differences with respect to pointing devices. The touchscreen and trackball are durable in public-access, shop-floor, and laboratory applications. The mouse, trackball, trackpoint, graphics tablet, and touchpad are effective for pixel-level pointing. Cursor jump keys are attractive when there is a small number of targets. Joysticks are appealing to game or aircraft-cockpit designers, apparently because of the firm grip and easy movement, but they are slow and inaccurate in guiding a cursor to a fixed destination in office automation and personal computing. Indirect-control pointing devices require more learning than do direct-control devices.

Figure 9.16

The effect of target distance on position time for arrow keys and mouse (Card et al., 1978). The positioning time for arrow keys increased dramatically with distance, because users had to make many keypresses to move the cursor to the target. The mouse time is independent of time over these distances. With very short distances and a few character positions, the arrow keys afforded a shorter mean time. (Adapted from S. K. Card, W. K. English, and B. J. Burr, Evaluation of mouse, rate-controlled isometric joystick, step keys, and task keys for text selection on a CRT, *Ergonomics*, 17, 6 [1965].)
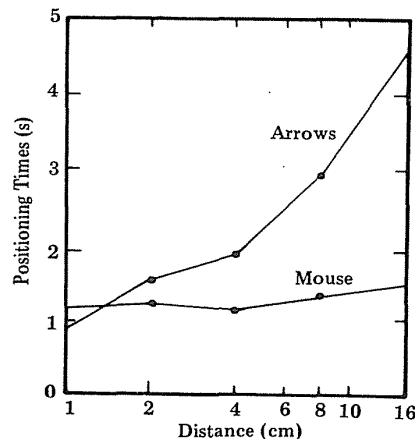
9.3.5 Fitts' Law

The major pointing devices depend on hand movement to control a cursor on the display. An effective predictive model of time to move a given distance, D, to a target of width, W, was developed by Paul Fitts (1954). He discovered that the pointing time is a function of the distance and the width; farther away and smaller targets take longer to point to. The *index of difficulty* is defined as

$$Index\ of\ difficulty = \log_2(2D/W).$$

The index of difficulty is a unitless number, but it is traditionally measured in bits. The time to perform the point action is

$$Time\ to\ point = C_1 + C_2\ (Index\ of\ difficulty),$$

where C_1 and C_2 are constants that depend on the device. Once data have been collected for a given device, C_1 and C_2 can be computed, and then predictions can be made regarding the time for other tasks. For example, for a 1 cm-wide target at a distance of 8 cm, the *Index of difficulty* is $\log_2(2*8/1) = \log_2(16) = 4$ bits. If tests show that, for a given device, $C_1 = 0.2$, $C_2 = 0.1$, then the *Time to point* $= 0.2 + 0.1\ (4) = 0.6$ seconds.

MacKenzie (1992) lucidly describes what Fitts' law is, how it has been applied, and what the many refinements are for cases such as two-dimensional pointing. In our studies of high-precision touchscreens (Sears and Shneiderman, 1991), we found that, in addition to the gross arm movement predicted by Fitts, there was also a fine-tuning motion of the fingers to move in on small targets such as a single pixel. A three-component equation was thus more suited for the high-precision pointing task:

$$Time\ for\ precision\ pointing = C_1 + C_2\ (index\ of\ difficulty) + C_3\ \log_2\ (C_4\ /\ W).$$

The third term, time for fine tuning, increases as the target width, W, decreases. This extension to Fitts' law is quite understandable and simple; it suggests that the *Time for precision pointing* at an object consists of a time for initiation of action, C_1, a time for gross movement, and time for fine adjustment. Fitts's studies focused on moderate movements, but current studies deal with a greater range of arm motion as well as precise finger positioning, even in three-dimensional space (Zhai et al., 1996). An open problem is how to design devices that produce smaller constants for the predictive equation.

9.3.6   Novel pointing devices

The popularity of pointing devices and the quest for new ways to engage diverse users for diverse tasks has led to provocative innovations. Since a user's hands might be busy on the keyboard, several designers have explored other methods for selection and pointing. Foot controls are popular with rock-music performers, organists, dentists, medical-equipment users, car drivers, and pilots, so maybe computer users could benefit from them as well. A foot mouse was tested and was found to take about twice as much time to use as a hand-operated mouse, but benefits in special applications may exist (Pearson and Weiser, 1986).

Eye-tracking, gaze-detecting controllers have been developed by several researchers and companies who make devices to assist the handicapped (Jacob, 1991). Nonintrusive and noncontact equipment using video-camera image recognition of pupil position can give 1- to 2-degree accuracy, and fixations of 200 to 600 milliseconds are used to make selections. Unfortunately, the "Midas touch problem" intrudes, since every gaze has the potential to activate an unintended command. For the moment, eye tracking remains a research tool and an aid for the handicapped.

The VPL DataGlove appeared in 1987 and attracted serious researchers, game developers, cyberspace adventurers, and virtual-reality devotees (see Section 6.8). Descendants of the original DataGlove are still often made of sleek black spandex with attached fiber-optic sensors to measure finger position. The displayed feedback can show the placement of each finger; thus, commands such as a closed fist, open hand, index-finger pointing, and thumbs-up gesture can be recognized. With a *Polhemus tracker,* complete three-dimensional placement and orientation can be recorded. Control over three-dimensional objects seems a natural application, but comparisons with other strategies reveal low precision and slow response. Devotees claim that the naturalness of gestures will enable use by many keyboard-averse or mouse-phobic users, although users require substantial training to master more than a half-dozen gestures. Gestural input with the glove can have special applications such as input of American Sign Language or musical performance.

An alternative to the goggles-and-gloves approach is the Fakespace *Binocular Omni-Orientation Monitor (BOOM)* (see Fig. 6.21), which allows users to step up to a viewer with handles and look in the binocular-like device while holding the handles to shift vantage points within the range of the mechanical boom. The display updates to create the illusion that the user is moving in three dimensions, and users have an immersive experience without the heavy and confining head-mounted goggles.

Support for virtual reality (see Chapter 6) is one motivation, but many design, medical, and other tasks may require three-dimensional input or

even six degrees of freedom to indicate a position and an orientation. Commercial devices include the Logitech 3D mouse, Ascension Bird, and Polhemus 3Ball. Glove-mounted devices or tethered balls are being refined (Zhai et al., 1996). The bat brush gives flexibility for artistic effects (Ware and Baxter, 1989), and other graspable user interfaces seem ripe for exploration (Fitzmaurice et al., 1995; Fitzmaurice, 1993). Matching of task with device and refining the input plus feedback strategies are common themes (Jacob et al., 1994).

Pointing devices with *haptic feedback* are an intriguing research direction. Several technologies have been employed to allow users to push a mouse or other device and to feel resistance (for example, as they cross a window boundary) or a hard wall (for example, as they navigate a maze). Three-dimensional versions, such as the Phantom, are still more intriguing, but compelling commercial applications have yet to emerge. Sound is a good substitute for haptic feedback in many cases, and the special-purpose applications of current haptic devices limit widespread use.

9.4   Speech Recognition, Digitization, and Generation

The dream of speaking to computers and having computers speak has lured many researchers and visionaries. Arthur C. Clarke's 1968 fantasy of the HAL 9000 computer in the book and movie *2001* has set the standard for future performance of computers in science fiction and for some advanced developers. The reality is more complex and sometimes more surprising than the dream. Hardware designers have made dramatic progress with speech and voice-manipulation devices, but current successes are sobering compared to the science-fiction fantasy (Yankelovich et al., 1995; Schmandt, 1994; Strathmeyer, 1990). Even science-fiction writers have shifted their scenarios, as shown by the reduced use of voice interaction in favor of larger visual displays in Star Trek's *Next Generation* and *Voyager*.

The vision of a computer that has a leisurely chat with the user seems more of a fantasy than a desired or believable reality. Instead, practical applications for specific tasks with specific devices are more effective in serving the user's need to work rapidly with low error rates. Designers are reluctantly recognizing that, even as technical problems are solved and the recognition algorithms improve, voice commanding is more demanding of users' working memory than is hand–eye coordination, which is processed elsewhere in the brain. Unfortunately, background noise and variations in user speech performance

make the challenge still greater. By contrast, *speech store and forward* and *speech generation* are satisfyingly predictable and appealingly available because of the telephone's ubiquity, but they will always be slower and more difficult to traverse than are textual and graphical displays. Speech store and forward is a success because the emotional content and prosody in human speech is compelling in voice messaging, museum tours, and instructional contexts.

The benefits to people with certain physical handicaps are rewarding to see, but the general users of office or personal computing are not rushing toward speech input and output. However, speech store-and-forward systems and telephone-based information services are growing in popularity. Speech is the bicycle of user-interface design: It is great fun to use and has an important role, but it can carry only a light load. Sober advocates know that it will be tough to replace the automobile: graphic user interfaces.

Speech enthusiasts can claim huge successes in *telephony,* where digital circuitry has increased the capacity of networks and have improved voice quality. Cellular telephones have been a huge success in developed countries and often bring telephone service rapidly to less developed countries. Internet telephony is rising rapidly, giving many users low-cost long-distance service even though sound quality is lower. The immediacy and emotional impact of a telephone conversation is a compelling component of human–human communication.

For designers of human–computer interaction, speech technology has four variations: discrete-word recognition, continuous-speech recognition, speech store and forward, and speech generation. A related topic is the use of audio tones, audiolization, and music. These components can be combined in creative ways: from simple systems that merely play back or generate a message, to complex interactions that accept speech commands, generate speech feedback, provide audiolizations of scientific data, and allow annotation plus editing of stored speech (Blattner et al., 1989).

A deeper understanding of neurological processing of sounds would be helpful. Why does listening to Mozart symphonies encourage creative work, whereas listening to radio news reports suspends it? Is the linguistic processing needed to absorb a radio news report disruptive, whereas background Mozart is somehow invigorating? Of course, listening to Mozart with the serious intention of a musicologist would be completely absorbing of mental resources. Are there uses of sound or speech and ways of shifting attention, that might be less disruptive or even supportive of symbolic processing, analytic reasoning, or graphic designing? Could sound be a more useful component of drawing software than of word processors?

9.4.1   Discrete-word recognition

Discrete-word–recognition devices recognize individual words spoken by a specific person; they can work with 90- to 98-percent reliability for 20- to 200-

word vocabularies. *Speaker-dependent training,* in which the user repeats the full vocabulary once or twice, is a part of most systems. *Speaker-independent* systems are beginning to be reliable enough for certain commercial applications. Quiet environments, head-mounted microphones, and careful choice of vocabularies improve recognition rates.

Applications for the physically handicapped have been successful in enabling bedridden, paralyzed, or otherwise disabled people to broaden the horizons of their life. They can control wheelchairs, operate equipment, or use personal computers for a variety of tasks.

Other applications have been successful when at least one of these conditions exist:

- Speaker's hands are busy.
- Mobility is required.
- Speaker's eyes are occupied.
- Harsh (underwater or battlefield) or cramped (airplane-cockpit) conditions preclude use of a keyboard.

Example applications include those for aircraft-engine inspectors who wear a wireless microphone as they walk around the engine opening cover-plates or adjusting components. They can issue orders, read serial numbers, or retrieve previous maintenance records by using a 35-word vocabulary. Baggage handlers for a major airline speak the destination city names as they place bags on a moving conveyor belt, thereby routing the bag to the proper airplane loading gate. For this application, the speaker-dependent training produced higher recognition rates when done in the noisy but more realistic environment of the conveyor belt rather than in the quiet conditions of a recording studio. Implementers should consider conducting speaker-dependent training in the task environment.

Consumer products include a wireless VCR controller with voice recognition for play, rewind, stop, and record (plus date and time commands); on this device, a sliding panel reveals the usual buttons if recognition fails because of a sore throat or a noisy room. Telephone companies are offering voice-dialing services to allow users simply to say "Call Mom" and be connected. Difficulties with training for multiple users in a household and reliable recognition are apparently slowing acceptance.

Many advanced development efforts have tested speech recognition in military aircraft, medical operating rooms, training laboratories, and offices. The results reveal problems with recognition rates even for speaker-dependent training systems, when background sounds change, when the user is ill or under stress, and when words in the vocabulary are similar (dime–time or Houston–Austin).

For common computing applications when a screen is used, speech input has not been beneficial. Studies of users controlling cursor movement by

voice and keyboard (Murray et al., 1983) found that cursor-movement keys were twice as fast and were preferred by users. In a study with four one-hour sessions, 10 typists and 10 nontypists used typed and spoken commands to correct online documents using the UNIX ed editor (Morrison et al., 1984). For both typed and spoken commands, the user still had to type parameter strings. Typists preferred to use the keyboard. Nontypists began with a preference for spoken commands, but switched to favor using the keyboard by the end of the four sessions. No significant differences were found for task-completion time or error rates.

In a study of 24 knowledgeable programmers, a voice editor led to a lower task-completion rate than did a keyboard editor. However, the keyboard entry produced a higher error rate (Leggett and Williams, 1984) (see Table 9.1). The authors suggest that further experience with voice systems, beyond the 90 minutes of this study, might lead to better performance. A speed advantage for voice entry over a menu-selection strategy was found in a study of two beginners and three advanced users of a CAD system (Shutoh et al., 1984).

A study of eight MacDraw users drawing eight diagrams each showed that allowing users to select one of 19 commands by voice instead of selection from a palette improved performance times by an average of 21 percent (Pausch and Leatherby, 1991). The advantage seems to have been gained through avoidance of the time-consuming and distracting effort of moving the cursor repeatedly from the diagram to the tool palette and back. A replication confirmed this result with word-processing tasks using 18 spoken commands such as "boldface," "down," "italic," "paste," and

Table 9.1

Average percentage scores for input and editing tasks with subjects using keyboard and voice editors. (Data from Leggett, John, and Williams, Glen, An empirical investigation of voice as an input modality for computer programming, *International Journal of Man–Machine Studies*, 21, (1984), 493–520.)

	Key editor	**Voice editor**
Input task		
Input task completed	70.6	50.7
Erroneous input	11.0	3.8
Edit Task		
Edit task completed	70.3	55.3
Erroneous commands	2.4	1.5
Erroneous input	14.3	1.2

"undo" (Karl et al., 1993). Although overall voice recognition was 19 percent faster than mouse pointing, mainly due to mouse acquisition time, error rates were higher for voice users in tasks that required high short-term–memory load. This unexpected result was explained by psychologists, who pointed out that short-term memory is sometimes referred to as "acoustic memory." Speaking commands is more demanding of working memory than is performing the hand–eye coordination needed for mouse pointing, which apparently is handled in parallel by other parts of the brain.

This result may explain the slower acceptance of speech interfaces as compared with GUIs; speaking commands or listening disrupts problem solving more than does selecting actions from a menu with a mouse. This phenomenon was noted by product evaluators for an IBM dictation package. They wrote that "thought, for many people is very closely linked to language. In keyboarding, users can continue to hone their words while their fingers output an earlier version. In dictation, users may experience more interference between outputting their initial thought and elaborating on it." (Danis et al., 1994)

Current research projects are devoted to improving recognition rates in difficult conditions, eliminating the need for speaker-dependent training, and increasing the vocabularies handled to 10,000 and even 20,000 words. IBM's speech-dictation system is trained by users reading a passage from Mark Twain for about 90 minutes. Users report satisfactory recognition rates when speaking with brief pauses between words. A newer version called Voice Type 3.0 promises simpler training, higher recognition rates, and an improved interface. With specialized vocabularies such as in radiology, even greater success has been demonstrated.

Whether continuous speech will gain widespread acceptance is still an open question. Speech recognition for discrete words works well for special-purpose applications, but it does not serve as a general interaction medium. Keyboards, function keys, and pointing devices with direct manipulation are often more rapid, and the actions or commands can be made visible for easy editing. Error handling and appropriate feedback with voice input are difficult and slow. Combinations of voice and direct manipulation may be useful, as indicated by Pausch and Leatherby's study.

9.4.2   Continuous-speech recognition

HAL's ability to understand the astronauts' spoken words and even to read their lips was an appealing fantasy, but the reality is more sobering. Although many research projects have pursued *continuous-speech recognition*, commercially successful products are still restricted to specialty niches

such as radiologists (Lai and Vergo, 1997). The difficulty lies in recognizing the boundaries between spoken words. Normal speech patterns blur the boundaries.

The hope is that, with a continuous-speech–recognition system, users could dictate letters, compose reports verbally for automatic transcription, and enable computers to scan long audio soundtracks, radio programs, or telephone calls for specific words or topics. Simply segmenting a movie by speakers would be a useful contribution. Using voice for identification purposes could be a benefit for security systems. Users would be asked to speak a novel phrase, and the system would ascertain which of the registered users was speaking.

Continuous-speech–recognition products are offered by manufacturers such as Verbex, which claims greater than 99.5-percent accuracy with speaker-dependent training with vocabularies of up to 10,000 words, and Speech Systems, which claims 95-percent accuracy for speaker-independent systems with 40,000 word vocabularies. IBM has tested a continuous-speech recognizer that uses subgrammars to increase recognition rates. Target tasks include operating-system control, police requests for information on car licenses, and stock-broker orders, but troubling error rates have delayed commercial distribution. Microsoft's Peedy, a graphically attractive conversational parrot, is cute at first, but I wonder whether users will become annoyed with it by their second and third encounter. A telephone message service called Wildfire is also appealing, but moving from a good demo to a working product may prove to be difficult.

Although progress has been made by the many companies and research groups, the following evaluation is still valid: "Comfortable and natural communication in a general setting (no constraints on what you can say and how you say it) is beyond us for now, posing a problem too difficult to solve" (Peacocke and Graf, 1990).

9.4.3   Speech store and forward

Less exciting but more immediately useful are the systems that enable storing and forwarding of spoken messages. Stored messages are commonly used for weather, airline, and financial information, but personal messaging through the telephone network is growing more popular. After registering with the service, users can touch commands on a 12-key telephone to store spoken messages and can have the messages sent to one or more people who are also registered with the service. Users can receive messages, replay messages, reply to the caller, forward messages to other users, delete messages, or archive messages. Automatic elimination of silences and speedup with frequency shifting can cut listening time in half.

Voice-mail technology works reliably, is fairly low cost, and is generally liked by users. Problems arise mainly because of the awkwardness of using

the 12-key telephone pad for commands, the need to dial in to check whether messages have been left, and the potential for too many "junk" telephone messages because of the ease of broadcasting a message to many people.

Telephone-based information systems have also been a great success, although many users are frustrated by the lengthy and deep menu structures, or by long informational speeches in which it seems that the needed fact is always at the end or is omitted. Well-designed systems (Resnick and Virzi, 1995) can provide reasonable customer service and timely information at relatively low cost.

Personal tape recorders are moving toward digital approaches, with small handheld voice note takers carving out a successful consumer market. Credit-card–sized devices that cost about $40 can store and randomly access several minutes of voice-quality notes. More ambitious handheld devices enable users to manage audio databases and to retrieve selected music segments or recorded lecture segments (Schmandt, 1994).

Audio tours in museums have been successful because they allow user control of the pace, while conveying the curator's enthusiasm. Educational psychologists conjecture that, if several senses (sight, touch, hearing) are engaged, then learning can be facilitated. Adding a spoken voice to an instructional system or an online help system may improve the learning process. However, there is evidence that users of instructional systems prefer textual displays to voice presentations (Resnick and Lammers, 1985). Adding voice annotation to a document may make it easier for teachers to comment on student papers, or for business executives to leave detailed responses or instructions. Editing the voice annotation is possible, but is still difficult.

9.4.4   Speech generation

Speech generation is an example of a successful technology that is used, but whose applicability was overestimated by some developers. Inexpensive, compact, reliable speech-generation (also called synthesis) devices have been used in cameras ("too dark—use flash"), soft-drink vending machines ("insert correct change and make your selection," "thank you"), automobiles ("your door is ajar"), children's games, and utility-control rooms ("danger").

In some cases, the novelty wears off leading to removal of the speech generation. Talking supermarket checkout machines that read products and prices were found to violate shoppers' sense of privacy about purchases and to be too noisy. Automobile speech-generation devices are now less widely used; a few tones and red-light indicators were found to be more acceptable. Spoken warnings in cockpits or control rooms were sometimes missed, or were in competition with human–human communication.

Applications for the blind are an important success story (Songco et al., 1980). The Kurzweil Reader is used in hundreds of libraries. Patrons can place a book on a copierlike device that scans the text and does an acceptable job of reading the text one word at a time.

The quality of the sound can be good when the words and pronunciation for digitized human speech can be stored in a dictionary. When algorithms are used to generate the sound, the quality is sometimes degraded. Digitized human speech for phrases or sentences is often a useful strategy, since human intonation provides more authentic sound. For some applications, a computerlike sound may be preferred. Apparently, the robotlike sounds used in the Atlanta airport subway drew more attention than did a tape recording of a human giving directions.

Michaelis and Wiggins (1982) suggest that speech generation is frequently preferable when the

1. message is simple.
2. message is short.
3. message will not be referred to later.
4. message deals with events in time.
5. message requires an immediate response.
6. visual channels of communication are overloaded.
7. environment is too brightly lit, too poorly lit, subject to severe vibration, or otherwise unsuitable for transmission of visual information.
8. user must be free to move around.
9. user is subjected to high G forces or anoxia (lack of oxygen, typically occuring at high altitudes). The magnitude of G forces or anoxia at which eyesight begins to be impaired is well below that needed to affect hearing.

These criteria apply to digitized human speech and to simple playbacks of tape recordings.

Digitized speech segments can be concatenated to form more complex phrases and sentences. Telephone-based voice information systems for banking (Fidelity Automated Service Telephone (FAST)), credit-card information (Citibank Customer Service), airline schedules (American Airlines Dial-AA-Flight), and so on have touchtone keying of codes and voice output of information.

In summary, speech generation is technologically feasible. Now, clever designers must find the situations in which it is superior to competing technologies. Novel applications may be by way of the telephone as a supplement to the CRT or through embedding in small consumer products.

9.4.5 Audio tones, audiolization, and music

In addition to speech, auditory machine outputs include individual *audio tones;* more complex information presentation by combinations of sound or *audiolization;* and *music.* Early Teletypes did include a bell tone to alert users that a message was coming or that paper had run out. Later computer systems added a range of tones to indicate warnings or simply to acknowledge the completion of an action. Even keyboards were built with the intent to preserve sound feedback. As digital-signal-processing chips to perform digital-to-analog and analog-to-digital conversions have become more powerful and cheaper, innovations have begun to appear. Gaver's SonicFinder (1989) added sound to the Macintosh interface by offering a dragging sound when a file was being dragged, a click when a window boundary was passed, and a thunk when the file was dropped into the trashcan for deletion. The effect for most users is a satisfying confirmation of actions; for visually impaired users, the sounds are vital. On the other hand, after a few hours the sound can become a distraction rather than a contribution, especially in a room with several machines and users. An auditory-enhanced scroll bar that provided feedback about user actions produced a 20- to 25-percent speedup in search and navigation tasks when tested with 12 experienced users (Brewster et al., 1994).

Auditory browsers for blind users (see Section 1.5.5) or telephonic usage have been proposed. Each file might have a sound whose frequency is related to its size, and might be assigned an instrument (violin, flute, trumpet). Then, when the directory is opened, each file might play its sound simultaneously or sequentially (in alphabetical order?). Alternatively, files might have sounds associated with their file types so that users could hear whether there were only spreadsheet, graphic, or text files (Blattner et al., 1989).

More ambitious audiolizations have been proposed (Smith et al., 1990; Blattner et al., 1991) in which scientific data are presented as a series of stereophonic sounds rather than as images. Other explorations have included audio tones for mass-spectrograph output to allow operators to hear the differences between a standard and a test sample, and appealing musical output to debug the execution of a computer with 16 parallel processors.

Adding traditional music to user interfaces seems to be an appropriate idea to heighten drama, to relax users, to draw attention, or to set a mood (patriotic marches, romantic sonatas, or gentle waltzes). These approaches have been used in video games and educational packages; they might also be suitable for public access, home control, sales kiosks, bank machines, and other applications.

The potential for novel musical instruments seems especially attractive. With a touchscreen, it should be possible to offer appropriate feedback to

give musicians an experience similar to a piano keyboard, a drum, a wood-wind, or a stringed instrument. There is a possibility of inventing new instruments whose frequency, amplitude, and effect are governed by the placement of the touch, as well as by its direction, speed, and even acceleration. Music composition using computers expanded as powerful musical-instrument digital-interface (MIDI) hardware and software has become widely available at reasonable prices, and user interfaces effectively combine piano and computer keyboards (Baggi, 1991).

9.5   Image and Video Displays

The *visual display unit (VDU)* has become the primary source of feedback to the user from the computer (Cakir et al., 1980; Grandjean, 1987; Helander, 1987). The VDU has many important features, including these:

- *Rapid operation*   Thousands of characters per second or a full image in a few milliseconds.
- *Reasonable size*   Early displays had 24 lines of 80 characters, but current devices show graphics and often more than 66 lines of 166 characters.
- *Reasonable resolution*   Typical resolution is 768 × 1024 pixels, but 1280 × 1024 is common.
- *Quiet operation*
- *No paper waste*
- *Relatively low cost*   Displays can cost as little as $100.
- *Reliability*
- *Highlighting*   Overwriting, windowing, and blinking are examples.
- *Graphics and animation*

Health concerns—such as visual fatigue, stress, and radiation exposure—are being addressed by manufacturers and government agencies, but remain active.

9.5.1   Display devices

For certain applications, *monochrome displays* are adequate and are attractive because of their lower cost. Color displays can make video games, educational simulations, CAD, and many other applications more attractive and effective for users. There are, however, real dangers in misusing color and difficulties in ensuring color constancy across devices.

Display technologies include:

- *Raster-scan cathode-ray tube (CRT)*   This popular device is similar to a television monitor, with an electron beam sweeping out lines of dots to form letters and graphics. The refresh rates (the reciprocal of the time required to produce a full screen image) vary from 30 to 70 per second. Higher rates are preferred, because they reduce *flicker*. Early CRT displays were often green because the P39 green phosphor has a long decay time, permitting relatively stable images. Another important property of a phosphor is the low *bloom level,* allowing sharp images because the small granules of the phosphor do not spread the glow to nearby points. The maximum resolution of a CRT is about 100 lines per inch but higher resolutions are being developed. CRT sizes (measured diagonally) range from less than 2 inches to more than 30 inches; popular models are in the range of 11 to 17 inches.

- *Liquid-crystal displays (LCDs)*   Voltage changes influence the polarization of tiny capsules of liquid crystals, turning some spots darker when viewed by reflected light. LCDs are flickerfree, but the size of the capsules limits the resolution. Portable computers usually have LCD displays because of the latter's thin form and light weight. Resolutions have moved up from 640 × 480 to 768 × 1024, with improved viewing from oblique angles, brighter images with better contrast, and more rapid adaptation to movement. Watches and calculators often use LCDs because of small size, light weight, and low power consumption.

- *Plasma panel*   Rows of horizontal wires are slightly separated from vertical wires by small glass-enclosed capsules of neon-based gases. When the horizontal and vertical wires on either side of the capsule receive a high voltage, the gas glows. Plasma displays are usually orange and are flickerfree, but the size of the capsules limits the resolution. Plasma computer displays have been built to display up to 62 lines of 166 characters, and bright multicolor plasma displays are being built.

- *Light-emitting diodes (LEDs)*   Certain diodes emit light when a voltage is applied. Arrays of these small diodes can be assembled to display characters. Here again, the resolution is limited by manufacturing techniques and costs are still high.

The technology employed affects these display attributes:

- Size
- Refresh rate
- Capacity to show animation

- Resolution
- Surface flatness
- Surface glare from reflected light
- Contrast between characters and background
- Brightness
- Line sharpness
- Character formation
- Tolerance for vibration

Each display technology has advantages and disadvantages with respect to these attributes. Users should expect these features:

- User control of contrast and brightness
- Software highlighting of characters by brightness
- Underscoring, reverse video, blinking (possibly at several rates)
- Extensive character set (alphabetic, numeric, special and international characters)
- Multiple type styles (for example, italic, bold) and fonts
- User control of cursor shape, blinking, and brightness
- Scrolling mechanism (smooth scrolling is preferred)
- User control of number of lines or characters per line displayed
- Support of negative and positive polarity (light on dark or dark on light)

Some frequent users place contrast-enhancement filters or masks in front of displays. Filters reduce reflected glare by using polarizers or a thin film of antireflective coating. Masks may be made of nylon mesh or simple matte surfaces. These devices are helpful to some users, but they can reduce resolution and are subject to smudging from fingerprints.

Dramatic progress in computer graphics has led to increasing use in motion pictures and television. Startling images have been created by George Lucas's Industrial Light and Magic and by Walt Disney Studios for movies, such as the *Star Wars* series, *Terminator 2, Jurassic Park,* and *Toy Story.* Many television commercials, station-identification segments, and news-related graphics have been constructed by computer animation. Finally, video games are another source of impressive computer-graphics images. The ACM's SIGGRAPH (Special Interest Group on Graphics) has an exciting annual conference with exhibitions of novel graphics devices and applica-

tions. The conference proceedings and videotape digest are rich sources of information.

9.5.2   Digital photography and scanners

Digital photography has become widespread in the news media and photographic agencies, where rapid electronic editing and dissemination is paramount. Many suppliers offer specialized cameras or add-ons for existing cameras (such as Kodak's attachment for Nikon cameras) that can take 100 images on a portable battery-driven hard-disk drive. Professional and amateur photographers are warming up to the digital cameras offered by Canon, Casio, Kodak, and SONY. Downloading to personal computers is simple, and display on the World Wide Web is a popular pursuit. The community of interested photographers has been enlarged with Kodak's PhotoCDs, which are produced from standard 35-millimeter negatives at the same time as prints are made. The PhotoCD's high resolution and good integration with personal computers are attractive to professional and amateur photographers who are seeking images for everything from professional documents to electronic family albums.

The increasing use of images has stimulated the need to scan photos, maps, documents, or handwritten notes. Page-sized scanners with 300-points-per-inch resolution are commonly available for a few hundred dollars, and larger scanners with higher resolution can be had for higher prices. Scanning packages often include character-recognition software that can convert text in printed documents into electronic forms with good reliability, but verification is necessary for demanding applications.

9.5.3   Digital video

The first generation of video applications was based on videodisk sources provided by producers who have access to interesting visual resources. Producers such as National Geographic (GEO), the Library of Congress (American Memory), ABC News (Election of 1988, Middle East history), Voyager (National Gallery of Art), and many others generated videodisks with tens of thousands of still images or hundreds of motion video segments. Each package had its own access software, and the thrill was to view the treasured images on command. Success often depended more on content than on design.

The 12-inch videodisks can store up to 54,000 still images, or 30 minutes of motion video, per side. Access time has been reduced steadily; on new players, it is under 1 second. Videodisk databases are a major application in museums (paintings, photos, and so on), travel (previews of hotels, tourist

attractions, and so on), education (microbiology slides, environmental aware-ness, current events, and so on), industrial training (truck drivers, financial sales, power-plant control, and so on), and sales (shoes, sports equipment, real estate, and so on). User-interface issues revolve around access to indexes, searching methods, action sets (start, pause, replay, stop, fast forward or backward), branching capability to allow individual exploration, capacity to extract and export, annotation, and synchronization with other activities.

Abbe Don, a multimedia artist who created "We Make Memories," used a HyperCard stack and videodisk to display family history as told by herself, her sister, her mother, and her grandmother. In this electronic version of a fam-ily photo album, events take on universal themes, and the emotional engage-ment foretells future applications that deal more with the heart than the head.

Videodisks are still suited for full-length motion pictures, but cheap *CD-ROMs* can provide up to 600 megabytes of textual or numeric data, or approximately 6000 graphic images, 1 hour of music, or 6 to 72 minutes (depending on effectiveness of compression and the resolution of the images) of motion video. CD-ROMs are relatively cheap, are small, and have reading devices smaller than those of videodisks. CD-ROMs are restructur-ing libraries and offices as the latter acquire more electronic reference sources and the computers to search that material. The next generation of CD-ROMs, *digital video disks (DVDs)*, will have an order-of-magnitude greater storage space, to allow storage of 2 hours of medium-resolution video.

Second-generation digital video capabilities—which allow users to create and store their own images and videos and to send this material to other people—have already begun to spice the pot of computing applications. Applications include video electronic mail, video conferences, personal image databases, video tutorials, video online help, and remote control with video feedback. Medical image-processing applications include X-ray images, sonograms, nuclear magnetic-resonance images (MRI), and com-puted-axial-tomography (CAT) scans.

Since video storage can consume many megabytes, efficient and rapid compression and decompression techniques become vital. The *Motion Picture Experts Group (MPEG)* approach has made digital-video servers a workable reality, even for full-length motion pictures. MPEG algorithms can compress 1 second of full-motion video into approximately 150 kilobytes—approxi-mately 5 kilobytes per frame. MPEG algorithms attempt to store only differ-ences across frames, so that stable images are compressed more than active or panning sequences. The elimination of special videodisk players and the capacity for recording video with standard magnetic media are attractive.

User-interface issues for these video environments are just beginning to be explored. For retrieval-oriented applications, the key question is how to find the desired videos in a library or a segment within a two-hour video.

Computer-based video-conferencing systems allow users to send an image over normal telephone lines in compressed data formats in a fraction of a second for low-resolution images, and in from 5 to 30 seconds for higher-resolution images. Increasingly available higher-speed lines—such as ISDN, leased lines, cable TV, and direct broadcast satellites—are enabling good-quality images and video to be used in a wide range of applications.

9.5.4   Projectors, heads-up displays, helmet-mounted displays

The desire to show and see computer-generated images has inspired several novel products. *Projector* television systems have been adapted to show the higher-resolution images from computers. These devices can generate 2 by 3 meter displays with good saturation, and larger displays with some loss of fidelity. An important variation is to use an LCD plate in connection with a common overhead projector to show color computer displays for meetings of 10 to 1000 people. These devices are rapidly declining in price and increasing in quality.

Personal display technology involves small portable monitors, often made with LCD in monochrome or color. A *heads-up display* projects information on a partially silvered windscreen of an airplane or car, so that the pilots or drivers can keep their attention focused on the surroundings while receiving computer-generated information. An alternative, the *helmet-mounted display (HMD)*, consists of a small partially silvered glass mounted on a helmet or hat that lets users see information even while turning their heads. In fact, the information that they see may be varied as a function of the direction in which they are looking.

The Private Eye technology uses a line of 200 LEDs and a moving mirror to produce 720- by 200-pixel resolution images in a lightweight and small display that can be mounted on a pair of glasses. This early example of wearable computers has focused attention on small portable devices that people can use while moving or accomplishing other tasks, such as jet engine repair or inventory control.

Attempts to produce three-dimensional displays include vibrating surfaces, holograms, polarized glasses, red–blue glasses, and synchronized shutter glasses. The CrystalEyes glasses from Stereographics shift from left- to right-eye vision at 120 hertz, and with a synchronized display, give users a strong sense of three-dimensional vision.

Still more innovative approaches come from performance artists such as Vincent Vincent, whose Mandala system is a three-dimensional environment for theatrical exploration. Performers or amateur users touch images of harps, bells, drums, or cymbals, and the instruments respond. Myron Krueger's artificial realities contain friendly video-projected cartoonlike

creatures who playfully crawl on your arm or approach your outstretched hand. In both of these environments, input is from video cameras or body sensors that do not require the user–performers to wear special equipment. Such environments invite participation, and the serious research aspects fade as joyful exploration takes over and you step inside the computer's world.

9.6 Printers

Even when they have good-quality and high-speed displays, people still have a great desire for hardcopy printouts. Paper documents can be easily copied, mailed, marked, and stored. The following are the important criteria for printers:

- Speed
- Print quality
- Cost
- Compactness
- Quietness of operation
- Type of paper (fanfolded or single sheet)
- Character set
- Variety of typefaces, fonts, and sizes
- Highlighting techniques (boldface, underscore, and so on)
- Support for special forms (printed forms, different lengths, and so on)
- Reliability

Early computer printers worked at 10 characters per second and did not support graphics. Personal-computer *dot-matrix printers* print more than 200 characters per second, have multiple fonts, can print boldface, use variable width and size, and have graphics capabilities. *Inkjet printers* offer quiet operation and high-quality output on plain paper. *Thermal printers* (often used in *fax machines*) offer quiet, compact, and inexpensive output on specially coated or plain paper.

Printing systems on mainframe computers with *impact line printers* that operate at 1200 lines per minute have all but vanished in favor of *laser printers* that operate at 30,000 lines per minute. The laser printers, now widely available for microcomputers, support graphics and produce high-quality images. Speeds vary from 4 to 40 pages per minute; resolution ranges from 200 to 1200 points per inch. Software to permit publication-quality typesetting has opened the door to *desktop-publishing* ventures.

Compact laser printers offer users the satisfaction of producing elegant business documents, scientific reports, novels, or personal correspondence. Users should consider output quality, speed, choice of faces and fonts, graphics capabilities, and special paper requirements.

Color printers allow users to produce hardcopy output of color graphics, usually by an inkjet approach with three colored inks and a black ink. The printed image is often of lower quality than the screen image and may not be faithful to the screen colors. Color laser printers or dye-transfer methods bring the satisfaction of bright and sharp color images, but at a higher price.

Plotters enable output of graphs, bar charts, line drawings, and maps on rolls of paper or sheets up to 100 by 150 cm. Plotters may have single pens, multiple pens, or inkjets for color output. Other design factors are the precision of small movements, the accuracy in placement of the pens, the speed of pen motion, the repeatability of drawings, and the software support.

Photographic printers allow the creation of 35-millimeter or larger slides (transparencies) and photographic prints. These printers are often designed as add-on devices in front of a display, but high-quality printing systems are independent devices. *Newspaper-* or *magazine-layout systems* allow electronic editing of images and text before generation of production-quality output for printing.

9.7   Practitioner's Summary

Choosing hardware always involves making a compromise between the ideal and the practical. The designer's vision of what an input or output device should be must be tempered by the realities of what is commercially available within the project budget. Devices should be tested in the application domain to verify the manufacturer's claims, and testimonials or suggestions from users should be obtained.

Designers should pay attention to current trends for specific devices, such as the mouse, touchscreen, stylus, or voice recognizer. Since new devices and refinements to old devices appear regularly, device-independent architecture and software will permit easy integration of novel devices. Avoid being locked into one device; the hardware is often the softest part of the system. Also, remember that a successful software idea can become even more successful if reimplementation on other devices is easy to achieve.

Keyboard entry is here to stay for a long time, but consider other forms of input when text entry is limited. Selecting rather than typing has many benefits for both novice and frequent users. Direct-pointing devices are faster

and more convenient for novices than are indirect-pointing devices, and accurate pointing is now possible. Beware of the hand-off-the-keyboard problem for all pointing devices, and strive to reduce the number of shifts between the keyboard and the pointing device.

Speech input and output are commercially viable and should be applied where appropriate, but take care to ensure that performance is genuinely improved over other interaction strategies. Display technology is moving rapidly and user expectations are increasing. Higher-resolution, color, and larger displays will be sought by users. Even when they have sharp, rapid, and accurate color displays, users still need high-quality hardcopy output.

9.8   Researcher's Agenda

Novel text-entry keyboards to speed input and to reduce error rates will have to provide significant benefits to displace the well-entrenched QWERTY design. For numerous applications not requiring extensive text entry, opportunities exist to create special-purpose devices or to redesign the task to permit direct-manipulation selection instead. Increasingly, input can be accomplished via conversion or extraction of data from online sources. Another input source is optical character recognition of printed text or of bar codes printed in magazines, on bank statements, in books, or on record albums.

Pointing devices will certainly play an increasing role. A clearer understanding of pointing tasks and the refinement of pointing devices to suit each task seem inevitable. Improvements can be made not only to the devices, but also to the software with which the devices are used. The same mouse hardware can be used in many ways to speed up movement, to provide more accurate feedback to users, and to reduce errors.

Research on speech systems can also be directed at improving the device and at redesigning the application to make more effective use of the speech input and output technology. Complete and accurate continuous-speech recognition does not seem attainable, but if users will modify their speaking style in specific applications, then more progress is possible. Another worthy direction is to increase rates of continuous-speech recognition for such tasks as finding a given phrase in a large body of recorded speech.

Larger, higher-resolution displays seem attainable. Thin, lightweight, durable, and inexpensive displays will spawn many applications, not only in portable computers, but also for embedding in briefcases, appliances, telephones, and automobiles. A battery-powered book-sized computer could contain the information from thousands of books or support Internet access.

A low-cost *webtop* computer for reading only could do away with keyboards, hard-disk drives, and floppy-disk drives.

Among the most exciting developments will be the increased facility for manipulating video and images. Many possibilities will open with improved graphics editors; faster image-processing hardware and algorithms; and cheaper image input, storage, and output devices. How will people search for images, integrate images with text, or modify images? What level of increased visual literacy will schools expect? How can animation be used as a more common part of computer applications? Can the hardware or software evoke more emotional responses and broaden the spectrum of computer devotees?

World Wide Web Resources | WWW |

Rich resources are available on commercial input devices, especially pointing devices and handwriting input. Promoters of speech recognition offer commercial packages, software tools, and demonstrations. MIDI tools and virtual reality devices enable serious hobbyists and researchers to create novel experiences for users.

http://www.aw.com/DTUI

References

Baggi, Dennis L., Computer-generated music, *IEEE Computer*, 24, 7 (July 1991), 6–9.

Blattner, Meera M., Greenberg, R. M., and Kamegai, M., Listening to turbulence: An example of scientific audiolization. In Blattner, M. and Dannenberg, R. B. (Editors), *Interactive Multimedia Computing*, ACM Press, New York (1991).

Blattner, Meera M., Sumikawa, Denise A., and Greenberg, R. M., Earcons and icons: Their structure and common design principles, *Human–Computer Interaction*, 4, (1989), 11–44.

Brewster, Stephen A., Wright, Peter C., and Edwards, Alistair D. N., The design and evaluation of an auditory-enhanced scrollbar, *Proc. CHI '94: Human Factors in Computing Systems*, ACM, New York (1994), 173–179.

Buxton, William, There's more to interaction than meets the eye: Some issues in manual input. In Norman, D. A., and Draper, S. W. (Editors), *User Centered System Design: New Perspectives on Human–Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ (1985) 319–337.

Cakir, A., Hart, D. J., and Stewart, T. F. M., *The VDT Manual*, John Wiley and Sons, New York (1980).

Card, Stuart K., Mackinlay, Jock D., and Robertson, George G., A morphological analysis of the design space of input devices, *ACM Transactions on Information Systems*, 9, 2 (1991), 99–122.

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

Danis, Catalina, Comerford, Liam, Janke, Eric, Davies, Ken, DeVries, Jackie, and Bertran, Alex, StoryWriter: A speech oriented editor, *Proc. CHI '94: Human Factors in Computing Systems: Conference Companion*, ACM, New York (1994), 277–278.

Dunsmore, H. E., Data entry. In Kantowitz, Barry H., and Sorkin, Robert D., *Human Factors: Understanding People–Systems Relationships*, John Wiley and Sons, New York (1983), 335–366.

Emmons, W. H., A comparison of cursor-key arrangements (box versus cross) for VDUs. In Grandjean, Etienne (Editor), *Ergonomics and Health in Modern Offices*, Taylor and Francis, London and Philadelphia (1984), 214–219.

English, William K., Engelbart, Douglas C., and Berman, Melvyn L., Display-selection techniques for text manipulation, *IEEE Transactions on Human Factors in Electronics*, HFE–8, 1 (March 1967), 5–15.

Ewing, John, Mehrabanzad, Simin, Sheck, Scott, Ostroff, Dan, and Shneiderman, Ben, An experimental comparison of a mouse and arrow-jump keys for an interactive encyclopedia, *International Journal of Man–Machine Studies*, 23, 1 (January 1986), 29–45.

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

Fitzmaurice, George W., Situated information spaces and spatially aware palmtop computers, *Communications of the ACM*, 36, 7 (1993), 38–49.

Fitzmaurice, George, W, Ishii, Hiroshi, and Buxton, William, Bricks: Laying the foundation for graspable user interfaces, *CHI '95: Human Factors in Computing Systems*, ACM, New York (1995), 442–449.

Foley, James D., Van Dam, Andries, Feiner, Steven K., and Hughes, John F., *Computer Graphics: Principles and Practice* (Second Edition), Addison-Wesley, Reading, MA (1990).

Foley, James D., Wallace, Victor L., and Chan, Peggy, The human factors of computer graphics interaction techniques, *IEEE Computer Graphics and Applications*, 4, 11 (November 1984), 13–48.

Frankish, Clive, Hull, Richard, and Morgan, Pam, Recognition accuracy and user acceptance of pen interfaces, *Proc. CHI '95 Conference: Human Factors in Computing Systems*, ACM, New York (1995), 503–510.

Gaver, William W., The SonicFinder: An interface that uses auditory icons, *Human–Computer Interaction*, 4, 1 (1989), 67–94.

Grandjean, E., Design of VDT workstations. In Salvendy, Gavriel (Editor), *Handbook of Human Factors*, John Wiley and Sons, New York (1987), 1359–1397.

Greenstein, Joel and Arnaut, Lynn, Input devices. In Helander, Martin, *Handbook of Human–Computer Interaction*, North-Holland, Amsterdam, The Netherlands (1988), 495–516.

Haider, E., Luczak, H., and Rohmert, W., Ergonomics investigations of workplaces in a police command-control centre equipped with TV displays, *Applied Ergonomics*, 13, 3 (1982), 163–170.

Haller, R., Mutschler, H., and Voss, M., Comparison of input devices for correction of typing errors in office systems, *INTERACT 84* (1984), 218–223.

Helander, Martin G., Design of visual displays. In Salvendy, Gavriel (Editor), *Handbook of Human Factors*, John Wiley and Sons, New York (1987), 507–548.

Jacob, Robert J. K., The use of eye movements in human–computer interaction techniques: What you look at is what you get, *ACM Trans. on Information Systems*, 9, 3 (1991), 152–169.

Jacob, Robert J. K., Leggett, John, Myers, Brad A., and Pausch, Randy, Interaction styles and input/output devices, *Behaviour & Information Technology*, 12, 2 (1993), 69–79.

Jacob, Robert J. K., Sibert, Linda E., McFarlane, Daniel C., and Mullen, Jr., M. Preston, Integrality and separability of input devices, *ACM Trans. on Computer–Human-Interaction*, 1, 1 (March 1994), 3–26.

Johnson, Jeff A., A comparison of user interfaces for panning on a touch-controlled display, *Proc. ACM CHI '95: Human Factors in Computing Systems*, ACM, New York (1995), 218–225.

Karat, John, McDonald, James, and Anderson, Matt, A comparison of selection techniques: Touch panel, mouse and keyboard, *INTERACT 84* (September 1984), 149–153.

Karl, Lewis, Pettey, Michael, and Shneiderman, Ben, Speech versus mouse commands for word processing applications: An empirical evaluation, *International Journal for Man–Machine Studies*, 39, 4 (1993), 667–687.

Kroemer, K. H. E., Operation of ternary chorded keys, *International Journal of Human–Computer Interaction*, 5, 3 (1993), 267–288.

Lai, Jennifer and Vergo, John, MedSpeak: Report creation with continuous speech recognition, Proc. *ACM CHI '97: Human Factors in Computing Systems*, ACM, New York (1997), 431–438.

Leggett, John, and Williams, Glen, An empirical investigation of voice as an input modality for computer programming, *International Journal of Man–Machine Studies*, 21, (1984), 493–520.

MacKenzie, I. Scott, Movement time prediction in human–computer interfaces, *Graphics Interface '92*, Morgan Kaufmann, San Francisco (1992), 140–150.

Michaelis, Paul Roller, and Wiggins, Richard H., A human factors engineer's introduction to speech synthesizers. In Badre, A. and Shneiderman, B. (Editors), *Directions in Human–Computer Interaction*, Ablex, Norwood, NJ (1982), 149–178.

Mithal, Anant Kartik and Douglas, Sarah A., Differences in movement microstructure of the mouse and the finger-controlled isometric joystick, *Proc. ACM CHI '96: Human Factors in Computing Systems*, ACM, New York (1996), 300–307.

Montgomery, Edward B., Bringing manual input into the twentieth century, *IEEE Computer*, 15, 3 (March 1982), 11–18.

Morrison, D. L., Green, T. R. G., Shaw, A. C., and Payne, S. J., Speech-controlled text-editing: effects of input modality and of command structure, *International Journal of Man–Machine Studies*, 21, 1 (1984), 49–63.

Murray, J. Thomas, Van Praag, John, and Gilfoil, David, Voice versus keyboard control of cursor motion, *Proc. Human Factors Society—Twenty-Seventh Annual Meeting,* Human Factors Society, Santa Monica, CA (1983), 103.

Nakaseko, M., Grandjean, E., Hunting, W., and Gierer, R., Studies of ergonomically designed alphanumeric keyboards, *Human Factors,* 27, 2 (1985), 175–187.

Pausch, Randy and Leatherby, James H., An empirical study: Adding voice input to a graphical editor, *Journal of the American Voice Input/Output Society,* 9, 2 (July 1991), 55–66.

Peacocke, Richard D. and Graf, Daryl H., An introduction to speech and speaker recognition, *IEEE Computer,* 23, 8 (August 1990), 26–33.

Pearson, Glenn and Weiser, Mark, Of moles and men: The design of foot controls for workstations, *Proc. ACM CHI '86: Human Factors in Computing Systems,* ACM, New York (1986), 333–339.

Potosnak, Kathleen M., Input devices. In Helander, Martin (Editor), *Handbook of Human–Computer Interaction,* North-Holland, Amsterdam, The Netherlands (1988), 475–494.

Resnick, Paul and Virzi, Robert A., Relief from the audio interface blues: Expanding the spectrum of menu, list, and form styles, *ACM Trans. on Computer-Human-Interaction,* 2, 2 (June 1995), 145–176.

Resnick, P. V. and Lammers, H. B., The influence of self esteem on cognitive responses to-machine-like versus human-like computer feedback, *Journal of Social Psychology,* 125, 6 (1985), 761–769.

Rutledge, J. D. and Selker, T., Force-to-motion functions for pointing in human–computer interaction, *Proc. INTERACT '90,* North-Holland, Amsterdam, The Netherlands (1990), 701–706.

Schmandt, Christopher, *Voice Communication with Computers,* Van Nostrand Reinhold, New York (1994).

Sears, Andrew, Plaisant, Catherine, and Shneiderman, Ben, A new era for touch-screen applications: High-precision, dragging, and direct manipulation metaphors. In Hartson, R. H. and Hix, D. (Editors), *Advances in Human–Computer Interaction,* Volume 3, Ablex, Norwood, NJ (1992), 1–33.

Sears, Andrew, Revis, Doreen, Swatski, Jean, Crittenden, Robert, and Shneiderman, Ben, Investigating touchscreen typing: The effect of keyboard size on typing speed, *Behaviour & Information Technology,* 12, 1 (Jan–Feb 1993), 17–22.

Sears, Andrew and Shneiderman, Ben, High precision touchscreens: Design strategies and comparison with a mouse, *International Journal of Man–Machine Studies,* 34, 4 (April 1991), 593–613.

Sherr, Sol (Editor), *Input Devices,* Academic Press, San Diego, CA (1988).

Shneiderman, B., Touch screens now offer compelling uses, *IEEE Software,* 8, 2 (March 1991), 93–94, 107.

Shutoh, Tomoki, Tsuruta, Shichiro, Kawai, Ryuichi, and Shutoh, Masamichi, Voice operation in CAD system. In Hendrick, H. W., and Brown, O., Jr., (Editors), *Human Factors in Organizational Design and Management,* Elsevier Science Publishers B.V. (North-Holland), Amsterdam, The Netherlands (1984), 205–209.

Smith, Stuart, Bergeron, R. Daniel, and Grinstein, Georges, G., Stereophonic and surface sound generation for exploratory data analysis, *Proc. CHI '90: Conference: Human Factors in Computing Systems,* ACM, New York (1990), 125–132.

Songco, D. C., Allen, S. I., Plexico, P. S., and Morford, R. A., How computers talk to the blind, *IEEE Spectrum,* [VOLUME, ISSUE] (May 1980), 34–38.

Stammers, R. B. and Bird, J. M., Controller evaluation of a touch input air traffic data system: An indelicate experiment, *Human Factors,* 22, 5 (1980), 581–589.

Strathmeyer, Carl R., Voice in computing: An overview of available technologies, *IEEE Computer,* 23, 8 (August 1990), 10–16.

Venolia, Dan and Neiberg, Forrest, T-Cube: A fast self-disclosing pen-based alphabet, *Proc. CHI '94 Conference: Human Factors in Computing Systems,* ACM, New York (1994), 265–270.

Ware, Colin and Baxter, Curtis, Bat Brushes: On the uses of six position and orientation parameters in a paint program, *Proc. CHI '89 Conference: Human Factors in Computing Systems,* ACM, New York (1989), 155–160.

Yankelovich, Nicole, Levow, Gina-Anne, and Marx, Matt, Designing SpeechActs: Issues in speech user interfaces, *Proc. CHI '95 Conference: Human Factors in Computing Systems,* ACM, New York (1995), 369–376.

Zhai, Shuman, Milgram, Paul and Buxton, William, The influence of muscle groups on performance of multiple degree-of-freedom input, *Proc. CHI '96 Conference: Human Factors in Computing Systems,* ACM, New York (1996), 308–315.
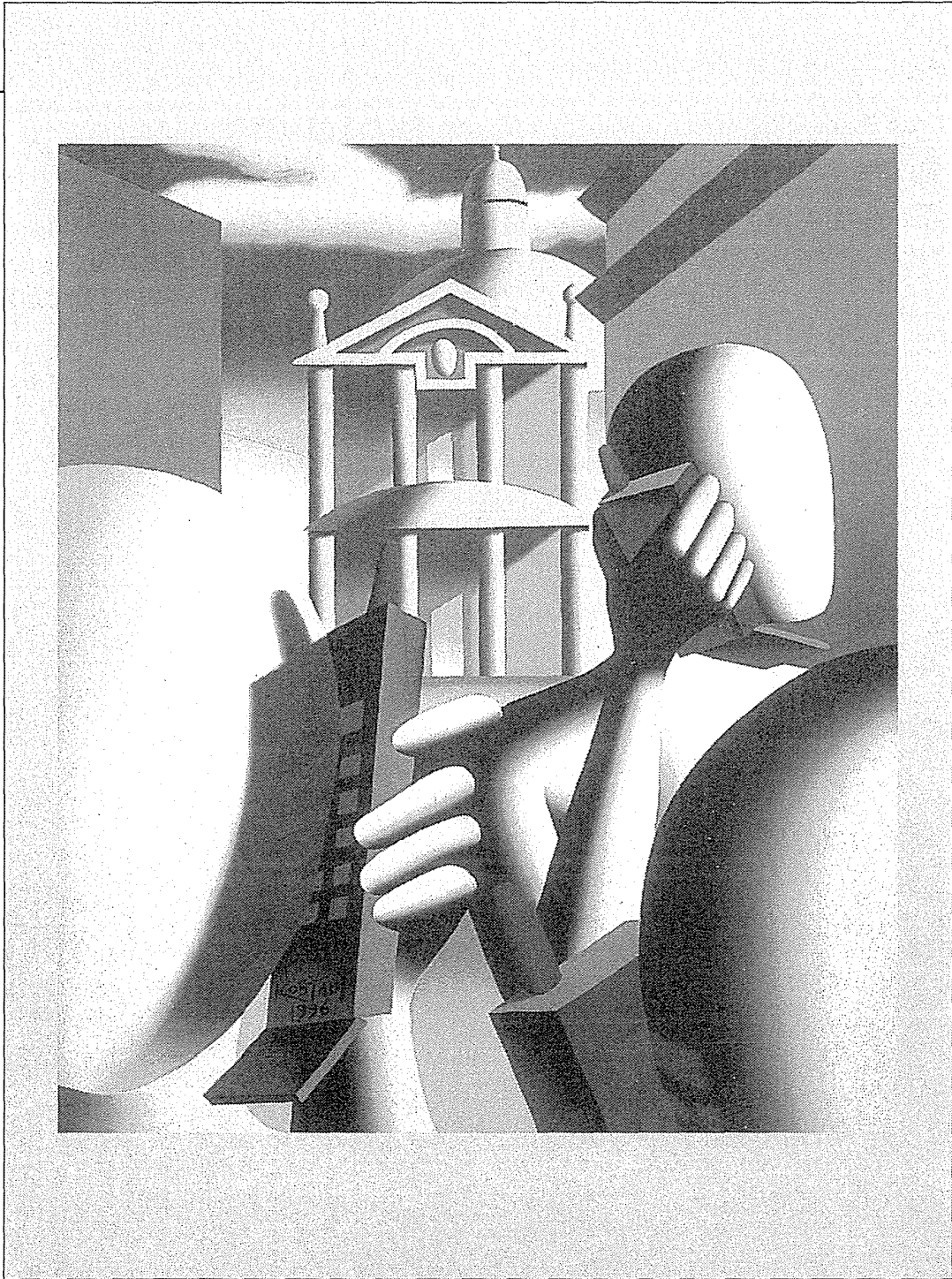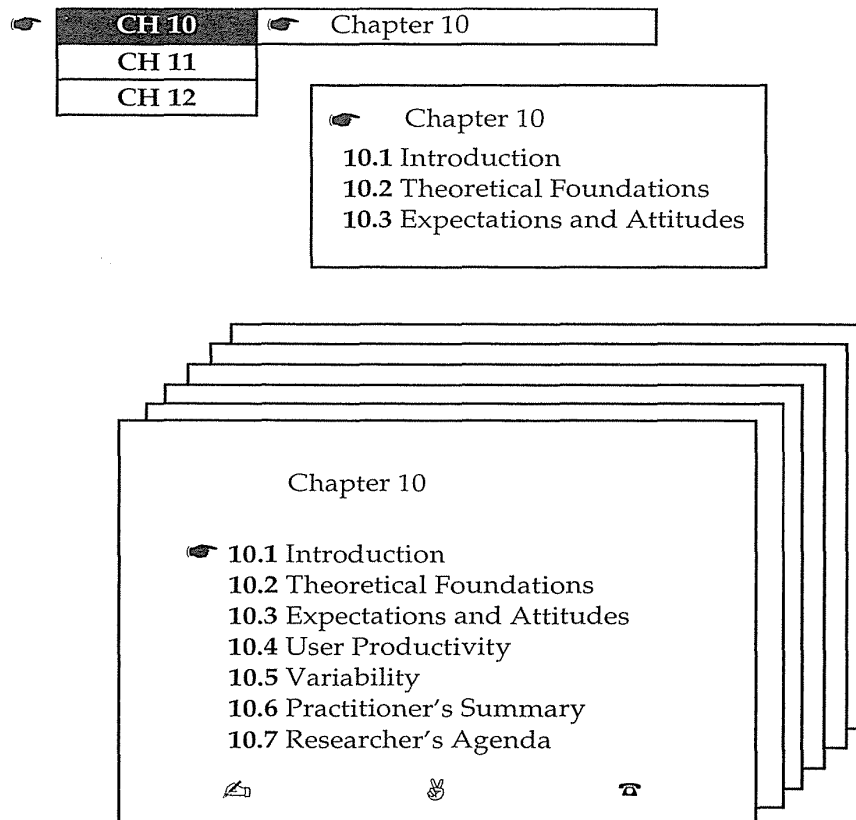
Mark Kostabi, *The Listeners (Post Modern Rome)*, 1996

Response Time and Display Rate

Stimulation is the indispensable requisite for pleasure in an experience,
and the feeling of bare time is the least stimulating experience we can have.

William James, *Principles of Psychology,* **Volume I, 1890**

Nothing can be more useful to a man than a determination not to be hurried.

Henry David Thoreau, *Journal*

☞ Chapter 10

10.1 Introduction
10.2 Theoretical Foundations
10.3 Expectations and Attitudes

Chapter 10

☞ **10.1** Introduction
  10.2 Theoretical Foundations
  10.3 Expectations and Attitudes
  10.4 User Productivity
  10.5 Variability
 **10.6** Practitioner's Summary
  10.7 Researcher's Agenda

10.1 Introduction

Time is precious. When externally imposed delays impede progress on a task, many people become frustrated, annoyed, and eventually angry. Lengthy or unexpected system response times and slow display rates produce these reactions in computer users, leading to frequent errors and low satisfaction. Some users accept the situation with a shrug of their shoulders, but most users prefer to work more quickly than the computer allows.

There is a danger in working too quickly. As users pick up the pace of a rapid interaction sequence, they may learn less, read with lower comprehension, make more ill-considered decisions, and commit more data-entry errors. Stress can build in this situation if it is hard to recover from errors, or if the errors destroy data, damage equipment, or imperil human life (for example, in air-traffic control or medical systems) (Emurian, 1989; Kuhmann, 1989).

The computer system's *response time* is the number of seconds it takes from the moment users initiate an activity (usually by pressing an ENTER
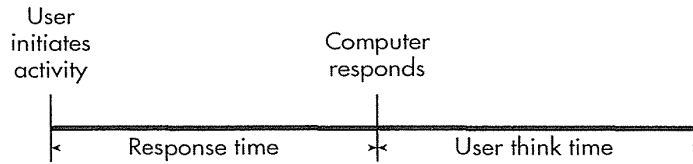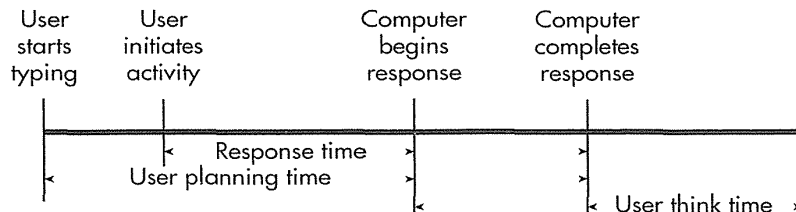
Figure 10.1

Simple model of system response time and user think time.

key or mouse button) until the computer begins to present results on the display or printer (Fig. 10.1). When the response is completely displayed, users begin formulating the next action. The *user think time* is the number of seconds during which users think before entering the next action. In this simple model, users initiate, wait for the computer to respond, watch while the results appear, think for a while, and initiate again.

In a more realistic model (Fig. 10.2), users plan while reading results, while typing, and while the computer is generating results or retrieving information across the network. Most people will use whatever time they have to plan ahead; thus, precise measurements of user think time are difficult to obtain. The computer's response is usually more precisely defined and measurable, but there are problems here as well. Some systems respond with distracting messages, informative feedback, or a simple prompt immediately after a command is initiated, but actual results may not appear for a few seconds.

Designers who specify response times and display rates in human–computer interactions have to consider the complex interaction of technical feasibility, costs, task complexity, user expectations, speed of task performance, error rates, and error-handling procedures. Decisions about these variables are further complicated by the influence of users' personality differences, fatigue, familiarity with computers, experience with the task, and motivation (Carbonell et al., 1968; Shneiderman, 1980).

Figure 10.2

Model of system response time, user planning time, and user think time. This model is more realistic than the one in Fig. 10.1.

Although some people are content with a slower system for some tasks, the overwhelming majority prefer rapid interactions. Overall productivity depends not only on the speed of the system, but also on the rate of human error and the ease of recovery from those errors. Lengthy (longer than 15-second) response times are generally detrimental to productivity, increasing error rates and decreasing satisfaction. More rapid (less than 1-second) interactions are generally preferred and can increase productivity, but may increase error rates for complex tasks. The high cost of providing rapid response times or display rates and the loss from increased errors must be evaluated in the choice of an optimum pace.

For alphanumeric displays, the *display rate* is the speed, in *characters per second (cps)*, at which characters appear for the user to read. The rate may be limited by inexpensive modems to 30 to 120 cps or the display may fill instantaneously (typical for many personal computers and workstations). In World Wide Web applications, the display rate of a page may be limited by network transmission speed or server performance. Portions of images or fragments of a page may appear with interspersed delays of several seconds. Display rates for graphics are measured in bytes per second; a typical home user has a 28.8Kbps modem, which is capable of receiving approximately 3600 bytes per second but more commonly receives 500 to 2000 bytes per second. At that rate, a 100-Kbyte image takes more than one minute to load—a long delay. Faster communications lines by more advanced network connections (such as Asynchronous Transfer Mode (ATM)), satellites, or cable modems could reduce times to a few seconds.

Reading textual information from a screen is a challenging cognitive and perceptual task—it is more difficult than reading from a book. If the display rate can be made so fast that the screen appears to fill instantly (beyond the speed at which someone might feel compelled to keep up), subjects seem to relax, to pace themselves, and to work productively. Since users often scan a web page looking for highlights or links, rather than reading the full text, it is useful to display text first, leaving space for the graphical elements that are slower to display.

This chapter begins in Section 10.2 by discussing a model of short-term human memory and identifying the sources of human error. Section 10.3 focuses on the role of users' expectations and attitudes in shaping users' subjective reactions to the computer-system response time. Section 10.4 concentrates on productivity as a function of response time. Section 10.5 reviews the research on the influence of variable response times.

10.2   Theoretical Foundations

A cognitive model of human performance that accounts for the experimental results in response time and display rates would be useful in making predictions, designing systems, and formulating management policies. A complete,

predictive model that accounts for all the variables may never be realized, but even fragments of such a model are useful to designers.

Robert B. Miller's review (1968) presented a lucid analysis of response-time issues and a list of 17 situations in which preferred response times might differ. Much has changed since his paper was written, but the principles of closure, short-term–memory limitations, and chunking still apply.

10.2.1 Limitations of short-term and working memory

Any cognitive model must emerge from an understanding of human problem-solving abilities and information-processing capabilities. A central issue is the limitation of *short-term memory* capacity.

George Miller's classic 1956 paper, "The magical number seven—plus or minus two," identified the limited capacities people have for absorbing information (Miller, 1956). People can rapidly recognize approximately seven (this value was contested by later researchers, but serves as a good estimate) *chunks* of information at a time and can hold those chunks in short-term memory for 15 to 30 seconds. The size of a chunk of information depends on the person's familiarity with the material.

For example, most people could look at seven binary digits for a few seconds and then recall the digits correctly from memory within 15 seconds. However, performing a distracting task during those 15 seconds, such as reciting a poem, would erase the binary digits. Of course, if people concentrate on remembering the binary digits and succeed in transferring them to long-term memory, then they can retain the binary digits for much longer periods. Most Americans could also probably remember seven decimal digits, seven alphabetic characters, seven English words, or even seven familiar advertising slogans. Although these items have increasing complexity, they are still treated as single chunks. However, Americans might not succeed in remembering seven Russian letters, Chinese pictograms, or Polish sayings. Knowledge and experience govern the size of a chunk for each individual.

People use short-term memory in conjunction with *working memory* for processing information and for problem solving. Short-term memory processes perceptual input, whereas working memory is used to generate and implement solutions. If many facts and decisions are necessary to solve a problem, then short-term and working memory may become overloaded. People learn to cope with complex problems by developing higher-level concepts that bring together several lower-level concepts into a single chunk. Novices at any task tend to work with smaller chunks until they can cluster concepts into larger chunks. Novices will break a complex task into a sequence of smaller tasks that they are confident about accomplishing.

This chunking phenomenon was demonstrated by Neal (1977), who required 15 experienced keypunch operators to type data records organized

into numeric, alphanumeric, and English word fields. The median interkey-stroke time was 0.2 seconds, but it rose to more than 0.3 seconds at field boundaries and 0.9 seconds at record boundaries.

Short-term and working memory are highly volatile; disruptions cause loss of information, and delays can require that the memory be refreshed. Visual distractions or noisy environments also interfere with cognitive processing. Furthermore, anxiety apparently reduces the size of the available memory, since the person's attention is partially absorbed in concerns that are beyond the problem-solving task.

10.2.2   Sources of errors

If people are able to construct a solution to a problem in spite of interference, they must still record or implement the solution. If they can implement the solution immediately, they can proceed quickly through their work. On the other hand, if they must record the solution in long-term memory, on paper, or on a complex device, the chances for error increase and the pace of work slows.

Multiplying two four-digit numbers in your head is difficult because the intermediate results cannot be maintained in working memory and must be transferred to long-term memory. Controlling a nuclear reactor or air traffic is a challenge in part because these tasks often require integration of information (in short-term and working memory) from several sources, as well as maintenance of awareness of the complete situation. In attending to newly arriving information, operators may be distracted and may lose the contents of their short-term or working memory.

When using an interactive computer system, users may formulate plans and then have to wait while they execute each step in the plan. If a step produces an unexpected result or if the delays are long, then the user may forget part of the plan or be forced to review the plan continually.

Long (1976) studied delays of approximately 0.1 to 0.5 seconds in the time for a keystroke to produce a character on an impact printer. He found that unskilled and skilled typists worked more slowly and made more errors with longer response times. Even these brief delays were distracting in the rapid process of typing. If users try to work too quickly, they may not allow sufficient time to formulate a solution plan correctly, and error rates may increase. As familiarity with the task increases, users' capacity to work more quickly and with fewer errors should increase.

This model leads to the conjecture that, for a given user and task, there is a preferred response time. Long response times lead to wasted effort and more errors because a solution plan is reviewed repeatedly. Shorter response times may generate a faster pace in which solution plans are prepared hastily and incompletely. More data from a variety of situations and users would clarify these conjectures.

10.2.3 Conditions for optimum problem solving

As response times grow longer, users may become more anxious because the penalty for an error increases. As the difficulty in handling an error increases, their anxiety levels increase, further slowing performance and increasing errors. As response times grow shorter and display rates increase, users pick up the pace of the system and may fail to comprehend the presented material, may generate incorrect solution plans, and may make more execution errors. Wickelgren (1977) reviews speed–accuracy tradeoffs.

Car driving may offer a useful analogy. Although higher speed limits are attractive to many drivers and do produce faster completion of trips, they also lead to higher accident rates. Since automobile accidents have dreadful consequences, we accept speed limits. When incorrect use of computer systems can lead to damage to life, property, or data, should not speed limits be provided?

Another lesson from driving is the importance of progress indicators. Drivers want to know how far it is to their destination and what progress they are making by seeing the declining number of miles on road signs. Similarly, computer users want to know how long it will take for a web page to load or a database search to complete. Users given graphical dynamic progress indicators rather than a static ("Please wait"), blinking, or numeric (number of seconds left) message had higher satisfaction and shorter perceived elapsed times to completion (Meyer et al., 1995; 1996).

Users may achieve rapid task performance, low error rates, and high satisfaction if the following criteria are met:

- Users have adequate knowledge of the objects and actions necessary for the problem-solving task.
- The solution plan can be carried out without delays.
- Distractions are eliminated.
- User anxiety is low.
- There is feedback about progress toward solution.
- Errors can be avoided or, if they occur, can be handled easily.

These conditions for optimum problem solving, with acceptable cost and technical feasibility, are the basic constraints on design. However, other conjectures may play a role in choosing the optimum interaction speed:

- Novices may exhibit better performance with somewhat slower response time.
- Novices prefer to work at speeds slower than those chosen by knowledgeable frequent users.
- When there is little penalty for an error, users prefer to work more quickly.

- When the task is familiar and easily comprehended, users prefer more rapid action.

- If users have experienced rapid performance previously, they will expect and demand it in future situations.

These informal conjectures need to be qualified and verified. Then a more rigorous cognitive model needs to be developed to accommodate the great diversity in human work styles and in computer-use situations. Practitioners can conduct field tests to measure productivity, error rates, and satisfaction as a function of response times in their application areas.

The experiments described in the following sections are tiles in the mosaic of human performance with computers, but many more tiles are necessary before the fragments form a complete image. Some guidelines have emerged for designers and information-system managers, but local testing and continuous monitoring of performance and satisfaction are still necessary. The remarkable adaptability of computer users means that researchers and practitioners will have to be alert to novel conditions that require revisions of these guidelines.

10.3   Expectations and Attitudes

How long will users wait for the computer to respond before they become annoyed? This simple question has provoked much discussion and several experiments. There is no simple answer to the question; more important, it may be the wrong question to ask.

Related design issues may clarify the question of acceptable response time. For example, how long should users have to wait before they hear a dial tone on a telephone or see a picture on their television? If the cost is not excessive, the frequently mentioned 2-second limit (Miller, 1968) seems appropriate for many tasks. In some situations, however, users expect responses within 0.1 second, such as when turning the wheel of a car; pressing a key on a typewriter, piano, or telephone; or changing channels on a television. Two-second delays in these cases might be unsettling because users have adapted a working style and expectation based on responses within a fraction of a second. In other situations, users are accustomed to longer response times, such as waiting 30 seconds for a red traffic light to turn green, two days for a letter to arrive, or one month for flowers to grow.

The first factor influencing acceptable response time is that people have established expectations based on their past experiences of the time required to complete a given task. If a task is completed more quickly than expected, people will be pleased; but if the task is completed much more quickly than

expected, they may become concerned that something is wrong. Similarly, if a task is completed much more slowly than expected, users become concerned or frustrated. Even though people can detect 8-percent changes in a 2- or 4-second response time (Miller, 1968), users apparently do not become concerned until the change is much greater.

Two installers of shared computer systems have reported a problem concerning user expectations with new systems. The first users are delighted because the response is short when the load is light. As the load builds, however, these first users become unhappy because the response time deteriorates. The users who come on later may be satisfied with what they perceive as normal response times. Both installers devised a *response-time choke* by which they could slow down the system when the load was light. This surprising policy makes the response time uniform over time and across users, thus reducing complaints.

Computer-center managers have similar problems with varying response times as new equipment is added or as large projects begin or complete their work. The variation in response time can be disruptive to users who have developed expectations and working styles based on a specific response time. There are also periods within each day when the response time is short, such as at lunch time, or when it is long, such as midmorning or late afternoon. Some users rush to complete a task when response times are short, and as a result they may make more errors. Some workers refuse to work when the response time is slow relative to their expectations.

A second factor influencing response-time expectations is the individual's tolerance for delays. Novice computer users may be willing to wait much longer than are experienced users. In short, there are large variations in what individuals consider acceptable waiting time. These variations are influenced by many factors, such as personality, costs, age, mood, cultural context, time of day, noise, and perceived pressure to complete work. The laid-back web surfer may enjoy chatting with friends while pages appear, but the anxious deadline-fighting journalist may start banging on desks or keys in a vain attempt to push the computer along.

Other factors influencing response-time expectations are the task complexity and the users' familiarity with the task. For simple repetitive tasks that require little problem solving, users want to perform rapidly and are annoyed by delays of more than a few tenths of a second. For complex problems, users can plan ahead during longer response times and will perform well even as response time grows. Users are highly adaptive and can change their working style to accommodate different response times. This factor was found in early studies of batch-programming environments and in recent studies of interactive-system usage. If delays are long, users will seek alternate strategies that reduce the number of interactions, whenever possi-

ble. They will fill in the long delays by performing other tasks, daydreaming, or planning ahead in their work. These long delays may or may not increase error rates when they are in the range of 3 to 15 seconds, but they probably will increase error rates when they are above 15 seconds if people must remain at the keyboard waiting for a response. Even if diversions are available, dissatisfaction grows with longer response times.

An increasing number of tasks place high demands on rapid system performance; examples are user-controlled three-dimensional animations, flight simulations, graphic design, and dynamic queries for information visualization. In these applications, users are continuously adjusting the input controls, and they expect changes to appear with no perceived delay—that is, within less than 100 milliseconds.

In summary, three primary factors influence users' expectations and attitudes regarding response time:

1. Previous experiences
2. Individual personality differences
3. Task differences

Experimental results show interesting patterns of behavior for specific backgrounds, individuals, and tasks, but it is difficult to distill a simple set of conclusions. Several experiments attempted to identify acceptable waiting times by allowing subjects to press a key if they thought that the waiting time was too long. Subjects who could shorten the response time in future interactions took advantage of that feature as they became more experienced. They forced response times for frequent commands down to well below 1 second. It seems appealing to offer users a choice in the pace of the interaction. Video-game designers recognize the role of user-controlled pace setting and the increased challenge from fast pacing. Differing desires open opportunities to charge premiums for faster service; for example, many World Wide Web users are willing to pay extra for faster network performance.

In summary, three conjectures emerge:

1. Individual differences are large and users are adaptive. They will work faster as they gain experience, and will change their working strategies as response times change. It may be useful to allow people to set their own pace of interaction.
2. For repetitive tasks, users prefer and will work more rapidly with short response times.
3. For complex tasks, users can adapt to working with slow response times with no loss of productivity, but their dissatisfaction increases as response times lengthen.

10.4 User Productivity

Shorter system response times usually lead to higher productivity, but in some situations users who receive long system response times can find clever shortcuts or ways to do concurrent processing to reduce the effort and time to accomplish a task. Working too quickly may lead to errors that reduce productivity.

In computing, just as in driving, there is no general rule about whether the high-speed highway or the slower, clever shortcut is better. The designer must survey each situation carefully to make the optimal choice. The choice is not critical for the occasional excursion, but becomes worthy of investigation when the frequency is great. When computers are used in high-volume situations, more effort can be expended in discovering the proper response time for a given task and set of users. It should not be surprising that a new study must be conducted when the tasks and users change, just as a new route evaluation must be done for each trip.

10.4.1 Repetitive tasks

The nature of the task has a strong influence on whether changes in response time alter user productivity. A repetitive control task involves monitoring a display and issuing commands in response to changes in the display. Although the operator may be trying to understand the underlying process, the basic activities are to respond to a change in the display, to issue commands, and then to see whether the commands produce the desired effect. When there is a choice among commands, the problem becomes more interesting and the operator tries to pick the optimal command in each situation. With shorter system response times, the operator picks up the pace of the system and works more quickly, but decisions on commands may be less than optimal. On the other hand, with short response times, the penalty for a poor choice may be small because it may be easy to try another command. In fact, operators may learn to use the system more quickly with short system response times because they can explore alternatives more easily.

Goodman and Spence (1981) studied a control task involving multiparameter optimization. The goal was to force "a displayed graph to lie wholly within a defined acceptance region." Operators could adjust five parameters by using lightpen touches, thus altering the shape of the graph. There were response times of 0.16, 0.72, or 1.49 seconds. Each of the 30 subjects worked at each of the three response times in this repeated-measures experiment.

The total times to solution (just over 500 seconds) and the total user think time (around 300 seconds) were the same for the 0.16- and 0.72-second treatments. The 1.49-second treatment led to a 50-percent increase in solution time and to a modest increase in user think time. In this case, reducing the response time to less than one second was beneficial in terms of human productivity. A pilot study of this task with six subjects provided further support for the productivity benefit of short response time: A response time of three seconds drove the solution time up to more than 1200 seconds.

In a data-entry task, users adopted one of three strategies, depending on the response time (Teal and Rudnicky, 1992). With response times under one second, users worked automatically without checking whether the system was ready for the next data value. This behavior resulted in numerous anticipation errors, in which the users typed data values before the system could accept those values. With response times above two seconds, users monitored the display carefully to make sure that the prompt appeared before they type. In the middle ground of one to two seconds, users paced themselves and waited an appropriate amount of time before attempting to enter data values.

10.4.2   Problem-solving tasks

When complex problem solving is required and many approaches to the solution are possible, users will adapt their work style to the response time. A demonstration of this effect emerged from early studies (Grossberg et al., 1976) using four experienced subjects doing complex matrix manipulations. The response time means were set at 1, 4, 16, and 64 seconds for commands that generated output or an error message. Nonoutput commands were simply accepted by the system. Each subject performed a total of 48 tasks of approximately 15 minutes duration each, distributed across the four response-time treatments.

The remarkable outcome of this study was that the time to solution was invariant with respect to response time! When working with 64-second delays, subjects used substantially fewer output commands and also fewer total commands. Apparently, with long response times, subjects thought carefully about the problem solution, since there were also longer intervals between commands. There were differences across subjects, but all subjects stayed within a limited range of solution times across the four system response times with which they worked.

Although the number of subjects was small, the results are strong in support of the notion that, if possible, users will change their work habits as the response time changes. As the cost in time of an error or an unnecessary output command rose, subjects made fewer errors and issued fewer commands. These results were closely tied to the study's complex, intellectually demanding task, for which there were several solutions.

Productivity with statistical problem-solving tasks was also found to be constant despite response-time changes over the range of 5.0 to 0.1 seconds (Martin and Corl, 1986). The same study with 24 regular users found linear productivity gains for simple data-entry tasks. The simpler and more habitual the task, the greater the productivity benefit of a short response time.

Barber and Lucas (1983) studied 100 professional circuit-layout clerks who assigned telephone equipment in response to service requests. Ten or more interactions were needed to complete each of these complex tasks. Data were collected about normal performance for 12 days with an average response time of 6 seconds. Then, 29 clerks were given response times averaging 14 seconds for 4 days. When the response time was as short as 4 seconds, there were 49 errors out of 287 transactions. As the response time increased to 12 seconds, the errors decreased to 16 of 222 transactions; and as the response time increased further to 24 seconds, the errors *increased* to 70 of 151 transactions (Fig. 10.3). The volume of transactions was recorded during sessions of 200 minutes. For this complex task, the data reveal that the lowest error rate occurred with a 12-second response time. With shorter response times, the workers made hasty decisions; with longer response times, the frustration of waiting burdened short-term memory. It is important to recognize that the number of productive transactions (total minus errors) increased almost linearly with reductions in response time. Apparently, reduced error rates were not sufficient to increase satisfaction, since subjective preference was consistently in favor of the shorter response time.
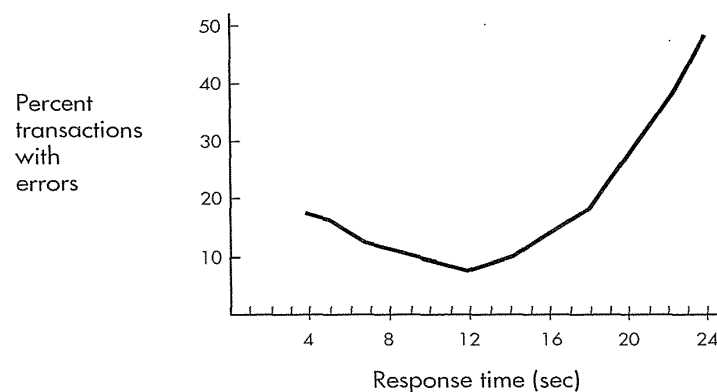
Figure 10.3

Error rates as a function of response time for complex telephone-circuit–layout task by Barber and Lucas (1983). Although error rates were lowest with long response times (12 seconds), productivity increased with shorter times because the system could detect errors and thus users could rapidly correct them.

10.4.3   Summary

It is clear that users pick up the pace of the system to work more quickly with shorter response times and that they consistently prefer a faster pace. The profile of error rates at shorter response times varies across tasks. Not surprisingly, each user-task situation appears to have an optimal pace—response times that are shorter or longer than this pace lead to increased errors. The ease of error recovery and the damage caused by an error must be evaluated carefully by managers who are choosing the optimal pace of interaction. If they desire higher throughput of work, then they must pay attention to minimizing the cost and delay of error recovery. In short, the optimal response time may be longer than the minimum possible response time.

10.5   Variability

People are willing to pay substantial amounts of money to reduce the variability in their life. The entire insurance industry is based on the reduction of present pleasures, through the payment of premiums, to reduce the severity of a future loss. Most people appreciate predictable behavior that lessens the anxiety of contemplating unpleasant surprises.

When they use computers, users cannot see into the machine to gain reassurance that the commands are being executed properly, but the response time can provide a clue. If users come to expect a response time of three seconds for a common operation, they may become apprehensive if this operation takes 0.5 or 15 seconds. Such extreme variation is unsettling and should be prevented or acknowledged by the system, with some indicator for unusually fast response, and a progress report for an unusually slow response.

The more difficult issue is the effect of modest variations in response time. As discussed earlier, Miller (1968) raised this issue and reported that 75 percent of subjects tested could perceive 8-percent variations in time for periods in the interval of two to four seconds. These results prompted some designers to suggest restrictive rules for variability of response time.

Since it may not be technically feasible to provide a fixed short response time (such as one second) for all commands, several researchers have suggested that the time be fixed for classes of commands. Many commands could have a fixed response time of less than one second, other commands could take four seconds, and still other commands could take 12 seconds. Experimental results suggest that modest variations in response time do not severely affect performance. Users are apparently capable of adapting to varying situations, although some of them may become frustrated when performing certain tasks.

Goodman and Spence (1981) attempted to measure performance changes in a problem-solving situation (a similar situation was used in their earlier experiment, described in Section 10.4.1). Subjects used lightpen touches to manipulate a displayed graph. The mean response time was set at 1.0 second with three levels of variation: quasinormal distributions with standard deviations of 0.2, 0.4, and 0.8 seconds. The minimum response time was 0.2 seconds, and the maximum response time was 1.8 seconds. Goodman and Spence found no significant performance changes as the variability was increased. The time to solution and the profile of command use were unchanged. As the variability increased, they did note that subjects took more advantage of fast responses by entering subsequent commands immediately, balancing the time lost in waiting for slower responses.

Similar results were found by researchers using a mean response time of 10 seconds and three variations: standard deviations of 0.0, 2.5, and 7.5 seconds (Bergman et al., 1981). The authors concluded that an increase in variability of response time "does not have any negative influence on the subject's performance on a rather complicated problem-solving task."

Two studies detected modest increases in user think time as variability increased. Butler (1983) studied six subjects who worked for two hours at each of 10 response-time conditions: the means were 2, 4, 8, 16, and 32 seconds, each with low and high variability. Subjects performed simple data-entry tasks but had to wait for the system response before they could proceed. Accuracy and rate of typing were unaffected by the duration or variability of response time. User think time increased with the duration and with variability of the computer's response time. Butler describes a second experiment with a more complex task whose results are similar.

The physiological effect of response-time variability was studied by Kuhmann and colleagues (1987), who found no dramatic effects between constant and variable treatments for detection and correction tasks with 68 subjects. Constant response times of two and eight seconds were compared with variable response times ranging over 0.5 to 5.75 seconds (mean two seconds) and 2.0 to 22.81 seconds (mean eight seconds). Statistically significantly higher error rates, higher systolic blood pressure, and more pronounced pain symptoms were found with shorter response times. However, no significant differences were found for response-time variability at either the short or long response times. Similarly, Emurian (1991) compared an 8-second constant response time to a variable response time ranging from one to 30 seconds (mean eight seconds). His 10 subjects solved 50 database queries with 45-second time limits. Although diastolic blood pressure and masseter (jaw-muscle) tension did increase when compared to resting baseline values, there were no significant differences in these physiological measures between constant and variable treatments.

In summary, modest variations in response time (plus or minus 50 percent of the mean) appear to be tolerable and to have little effect on performance.

As the variability grows, performance speed may decrease some. Frustration emerges only if delays are unusually long—at least twice the anticipated time. Similarly, anxiety about an erroneous command may emerge only if the response time is unusually short—say, less than one-quarter of the anticipated time. But even with extreme changes, users appear to be adaptable enough to complete their tasks.

It may be useful to slow down unexpected fast responses to avoid surprising the user. This proposal is controversial but would affect only a small fraction of user interactions. Certainly, designers should make a serious effort to avoid extremely slow responses, or, if responses must be slow, should give users information to indicate progress toward the goal. One graphics system displays a large clock ticking backward; the output appears only when the clock has ticked down to zero. Many print-spooling programs and document-formatting systems display the page numbers to indicate progress and to confirm that the computer is at work productively on the appropriate document.

10.6   Practitioner's Summary

Computer-system response time and display rate are important determinants of user productivity, error rates, working style, and satisfaction (Box 10.1). In most situations, shorter response times (less than one second) lead to higher productivity. For mouse actions, direct manipulation, typing feedback, and animation, even faster performance is necessary (less than 0.1 second). Satisfaction generally increases as the response time decreases, but there may be a danger from stress induced by a rapid pace. As users pick up the pace of the system, they may make more errors; if these errors are detected and corrected easily, then productivity will generally increase. If errors are hard to detect or are excessively costly, then a moderate pace may be most beneficial.

Designers can determine the optimal response time for a specific application and user community by measuring the change in productivity associated with, cost of errors resulting from, and cost of providing short response times. Managers must be alert to changes in work style as the pace quickens; productivity is measured by correctly completed tasks rather than by interactions per hour. Novices may prefer a slower pace of interaction. When technical feasibility or costs prevent response times of less than one second, each class of commands can be assigned to a response-time category—for example, two to four seconds, four to eight seconds, eight to 12 seconds, and more than 12 seconds. Modest variations around the mean response time are acceptable, but large variations (less than one-quarter of the mean or more than twice the mean) should be accompanied by an informative message. An alternative approach is to slow down overly rapid responses and thus to avoid the need for a message.

Box 10.1

Response-time guidelines.

- Users prefer shorter response times
- Longer response times (> 15 secs) are disruptive
- Users change usage profile with response time
- Shorter response time leads to shorter user think time
- A faster pace may increase productivity, but may increase error rates
- Error recovery ease and time influence optimal response time
- Response time should be appropriate to the task:
  - Typing, cursor motion, mouse selection: 50–150 milliseconds
  - Simple frequent tasks: 1 second
  - Common tasks: 2–4 seconds
  - Complex tasks: 8–12 seconds
- Users should be advised of long delays
- Modest variability in response time is acceptable
- Unexpected delays may be disruptive
- Empirical tests can help to set suitable response times

10.7   Researcher's Agenda

In spite of the experiments described here, many unanswered questions remain. The taxonomy of issues provides a framework for research, but a finer taxonomy of tasks, of relevant cognitive-style differences, and of work situations is needed if we are to specify adequate experimental controls. Next, a sound theory of problem-solving behavior with computers is necessary if we are to generate useful hypotheses.

Doherty and Kelisky (1979) suggest that longer response times lead to slower work, more emotional upset, and more errors. This statement appears to be true with long response times of more than 15 seconds, but there is little evidence to support the claim that fewer errors are made with short response times of less than one second. Barber and Lucas (1983) found a U-shaped error curve, with the lowest error rate at a 12-second response time. It would be productive to study error rates as a function of response time for a range of tasks and users.

It is understandable that error rates vary with response times, but how else are users' work styles affected? Do users issue more commands as response times shorten? Grossberg et al. (1976) found this result for a complex task with extremely long response times of up to 64 seconds, but there is

little evidence with more common tasks and speeds. Does the profile of commands shift to a smaller set of more familiar commands as the response time shortens? Does the session length increase or decrease with response-time increases? Are workers more willing to pursue higher-quality results when they are given shorter response times that enable multiple quick changes?

Many other questions are worthy of investigation. When technical feasibility prevents short responses, can users be satisfied by diversionary tasks, or are progress reports sufficient? Do warnings of long responses relieve anxiety or simply further frustrate users?

Operating-system implementers can also contribute by providing better user control over response time. It should be possible for a user-interface designer to specify upper and lower limits for response time for each command. It is still difficult on large shared computers to specify a response time, even on an experimental basis.

World Wide Web Resources   WWW

Response time issues have a modest presence on the net, although the issue of long network delays gets discussed frequently.

http://www.aw.com/DTUI

References

Barber, Raymond E. and Lucas, H. C., System response time, operator productivity and job satisfaction, *Communications of the ACM*, 26, 11 (November 1983), 972–986.

Bergman, Hans, Brinkman, Albert, and Koelega, Harry S., System response time and problem solving behavior, *Proc. of the Human Factors Society—Twenty-fifth Annual Meeting*, Rochester, NY (October 12–16, 1981), 749–753.

Butler, T. W., Computer response time and user performance, ACM CHI '83 Proceedings: Human Factors in Computer Systems (December 1983), 56–62.

Carbonell, J. R., Elkind, J. I., and Nickerson, R. S., On the psychological importance of time in a timesharing system, *Human Factors*, 10, 2 (1968), 135–142.

Doherty, W. J. and Kelisky, R. P., Managing VM/CMS systems for user effectiveness, *IBM Systems Journal*, 18, 1, (1979) 143–163.

Emurian, Henry H., Physiological responses during data retrieval: Consideration of constant and variable system response times, *Computers and Human Behavior*, 7 (1991), 291–310.

Emurian, Henry H., Human–computer interactions: Are there adverse health consequences?, *Computers and Human Behavior*, 5, (1989), 265–275.

Goodman, T. J., and Spence, R., The effect of computer system response time variability on interactive graphical problem solving, *IEEE Transactions on Systems, Man, and Cybernetics*, 11, 3 (March 1981), 207–216.

Goodman, Tom and Spence, Robert, The effects of potentiometer dimensionality, system response time, and time of day on interactive graphical problem solving, *Human Factors*, 24, 4 (1982), 437–456.

Grossberg, Mitchell, Wiesen, Raymond A., and Yntema, Douwe B., An experiment on problem solving with delayed computer responses, *IEEE Transactions on Systems, Man, and Cybernetics*, 6, 3 (March 1976), 219–222.

Kuhmann, Werner, Experimental investigation of stress-inducing properties of system response times, *Ergonomics*, 32, 3 (1989), 271–280.

Kuhmann, Werner, Boucsein, Wolfram, Schaefer, Florian, and Alexander, Johanna, Experimental investigation of psychophysiological stress-reactions induced by different system response times in human–computer interaction, *Ergonomics*, 30, 6 (1987), 933–943.

Lambert, G. N., A comparative study of system response time on program developer productivity, *IBM System Journal*, 23, 1 (1984), 36–43.

Long, John, Effects of delayed irregular feedback on unskilled and skilled keying performance, *Ergonomics*, 19, 2 (1976), 183–202.

Martin, G. L. and Corl, K. G., System response time effects on user productivity, *Behaviour and Information Technology*, 5, 1 (1986), 3–13.

Meyer, Joachim, Bitan, Yuval, and Shinar, David, Displaying a boundary in graphic and symbolic "wait" displays: Duration estimates and users' preferences, *International Journal of Human–Computer Interaction*, 7, 3 (1995), 273–290.

Meyer, Joachim, Shinar, David, Bitan, Yuval, and Leiser, David, Duration estimates and users' preferences in human–computer interaction, *Ergonomics*, 39, (1996), 46–60.

Miller, G. A., The magical number seven, plus or minus two: Some limits on our capacity for processing information, *Psychological Science*, 63, (1956), 81–97.

Miller, Robert B., Response time in man–computer conversational transactions, *Proceedings Spring Joint Computer Conference 1968*, 33, AFIPS Press, Montvale, NJ (1968), 267–277.

Neal, Alan S., Time interval between keystrokes, records, and fields in data entry with skilled operators, *Human Factors*, 19, 2 (1977), 163–170.

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

Teal, Steven L. and Rudnicky, Alexander I., A performance model of system delay and user strategy selection, *Proc. CHI '92 Human Factors in Computer Systems*, ACM, New York (1992), 295–305.

Wickelgren, Wayne A., Speed-accuracy tradeoff and information processing dynamics, *Acta Psychologica*, 41, (1977), 67–85.
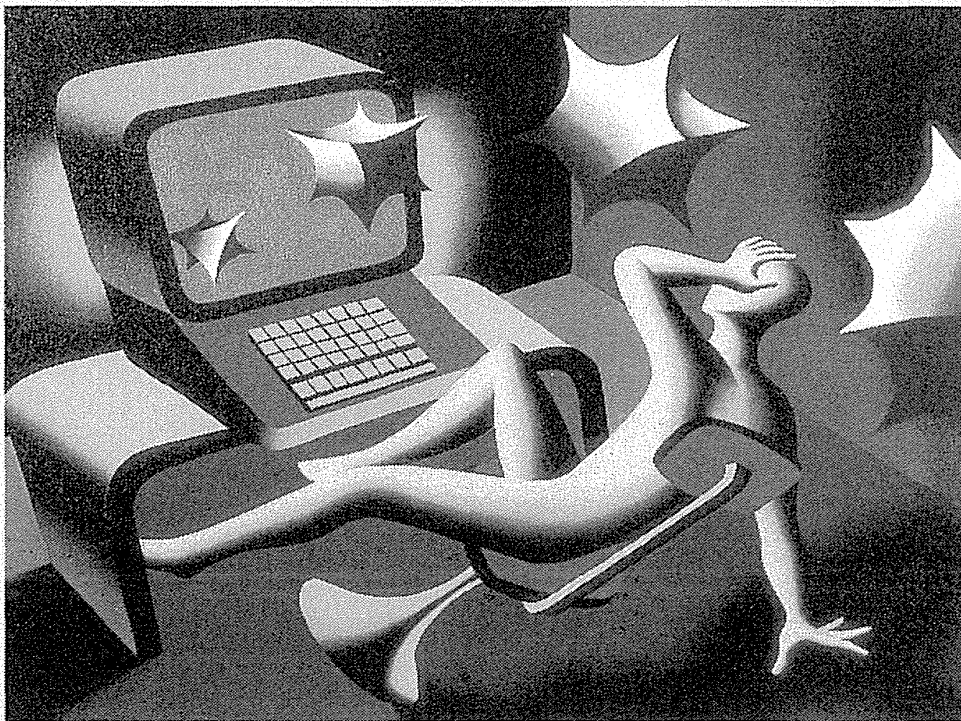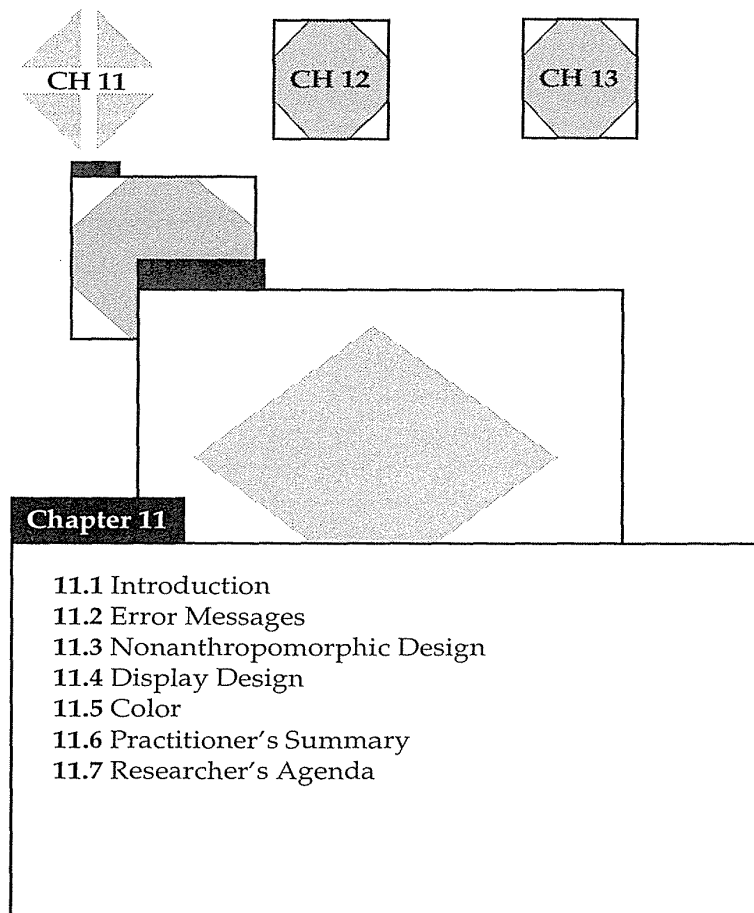
Mark Kostabi, *Update*, 1996

Presentation Styles: Balancing Function and Fashion

Words are sometimes sensitive instruments of precision with which delicate operations may be performed and swift, elusive truths may be touched.

Helen Merrell Lynd, *On Shame and the Search for Identity*

CH 11

CH 12

CH 13

11.1 Introduction

Interface design has yet to match the high art of architecture or trendiness of clothing design. However, we can anticipate that, as the audience for computers expands, competition over design will heighten. Early automobiles were purely functional and Henry Ford could joke about customers getting any color as long as it was black, but modern car designers have learned to balance function and fashion. This chapter deals with four design matters that are functional issues with many human factors criteria, but that also leave room for varying styles to suit a variety of customers. They are error messages, nonanthropomorphic design, display design, and color.

User experiences with computer-system prompts, explanations, error diagnostics, and warnings play a critical role in influencing acceptance of software systems. The wording of messages is especially important in systems designed for novice users; experts also benefit from improved messages. Messages are sometimes meant to be conversational, as modeled by human–human communication, but this strategy has its limits because people are different from computers and computers are different from people. This fact may be obvious, but a section on nonanthropomorphic design seems necessary to steer designers toward comprehensible, predictable, and controllable interfaces.

Another opportunity for design improvements lies in the layout of information on a display. Cluttered displays may overwhelm even knowledgeable users; but with only modest effort, we can create well-organized information-abundant layouts that reduce search time and increase subjective satisfaction. Large, fast, high-resolution color displays offer many possibilities and challenges for designers. Some guidelines are useful, but there are too many variables and situations to ensure success without repeated trials even by experienced designers.

Recognition of the creative challenge of balancing function and fashion might be furthered by having designers put their names and photos on a title or credits page, just as authors do in a book. Such acknowledgment is common in games and in some educational software, and seems appropriate for all software. Credits provide recognition for good work, and identify the people responsible. Having their name in lights may also encourage designers to work a bit harder, since their identities will be public.

11.2   Error Messages

Normal prompts, advisory messages, and system responses to user actions may influence user perceptions, but the phrasing of error messages or diagnostic warnings is critical. Since errors occur because of lack of knowledge, incorrect understanding, or inadvertent slips, users are likely to be confused, to feel inadequate, and to be anxious. Error messages with an imperious tone that condemns users can heighten anxiety, making it more difficult to correct the error and increasing the chances of further errors. Messages that are too generic, such as WHAT? or SYNTAX ERROR, or that are too obscure, such as FAC RJCT 004004400400 or 0C7, offer little assistance to most users.

These concerns are especially important with respect to novices, whose lack of knowledge and confidence amplify the stress that can lead to a

sequence of failures. The discouraging effects of a bad experience in using a computer are not easily overcome by a few good experiences. In some cases, systems are remembered more for what happens when things go wrong than for when things go right. Although these concerns apply most forcefully to novice computer users, experienced users also suffer. Experts in one system or part of a system are still novices in many situations.

Producing a set of guidelines for writing system messages is not an easy task because of differences of opinion and the impossibility of being complete (Dean, 1982). However, explicit guidelines generate discussions and help less experienced designers to produce better systems. Improving the error messages is one of the easiest and most effective ways to improve an existing system. If the software can capture the frequency of errors, then people can focus on fixing the most important messages.

Error-frequency distributions also enable system designers and maintainers to revise error-handling procedures, to improve documentation and training manuals, to alter online help, or even to change the permissible actions. The complete set of messages should be reviewed by peers and managers, tested empirically, and included in user manuals.

Specificity, constructive guidance, positive tone, user-centered style, and appropriate physical format are recommended as the bases for preparing system messages. These guidelines are especially important when the users are novices, but they can benefit experts as well. The phrasing and contents of system messages can significantly affect user performance and satisfaction.

11.2.1   Specificity

Messages that are too general make it difficult for the novice to know what has gone wrong. Simple and condemning messages are frustrating because they provide neither enough information about what has gone wrong nor the knowledge to set things right. The right amount of specificity therefore is important.

Poor	Better
SYNTAX ERROR	Unmatched left parenthesis
ILLEGAL ENTRY	Type first letter: Send, Read, or Drop
INVALID DATA	Days range from 1 to 31
BAD FILE NAME	File names must begin with a letter

Execution-time messages in programming languages should provide the user with specific information about where the problem arose, what objects were involved, and what values were improper. One system for hotel checkin required the desk clerk to enter a 40- to 45-character string

containing the name, room number, credit-card information, and so on. If the clerk made a data-entry error, the only message was INVALID INPUT. YOU MUST RETYPE THE ENTIRE RECORD. This led to frustration for users and delays for irritated guests. Interactive systems should be designed to minimize input errors by proper form-fillin strategies (see Chapter 7); when an error occurs, the users should have to repair only the incorrect part.

Systems that offer an error-code number leading to a paragraph-long explanation in a manual are also annoying because the manual may not be available, or consulting it may be disruptive and time consuming. In most cases, system developers can no longer hide behind the claim that printing meaningful messages consumes too many system resources.

11.2.2   Constructive guidance and positive tone

Rather than condemning users for what they have done wrong, messages should, where possible, indicate what users need to do to set things right:

Poor: DISASTROUS STRING OVERFLOW. JOB ABANDONED. (From a well-known compiler-compiler.)

Better: String space consumed. Revise program to use shorter strings or expand string space

Poor: UNDEFINED LABELS (From a FORTRAN compiler.)

Better: Define statement labels before use

Poor: ILLEGAL STA. WRN. (From a FORTRAN compiler.)

Better: RETURN statement cannot be used in a FUNCTION subprogram.

Unnecessarily hostile messages using violent terminology can disturb nontechnical users. An interactive legal-citation–searching system uses this message: FATAL ERROR, RUN ABORTED. A popular operating-system threatens many users with CATASTROPHIC ERROR; LOGGED WITH OPERATOR. There is no excuse for these hostile messages; they can easily be rewritten to provide more information about what happened and what must be done to set things right. Such negative words as ILLEGAL, ERROR, INVALID, or BAD should be eliminated or used infrequently.

It may be difficult for the software writer to create a program that accurately determines what was the user's intention, so the advice to be constructive is often difficult to apply. Some designers argue for automatic error correction, but the disadvantage is that the user may fail to learn proper syntax and may become dependent on alterations that the system makes.

Another approach is to inform the user of the possible alternatives and to let the user decide. A preferred strategy is to prevent errors from occurring (see Section 2.6).

11.2.3   User-centered phrasing

The term *user-centered* suggests that the user controls the system—initiating more than responding. Designers partially accomplish this scenario by avoiding the negative and condemning tone in messages and by being courteous to the user. Prompting messages should avoid such imperative forms as ENTER DATA, and should focus on such user control as READY FOR COMMAND or simply READY.

Brevity is a virtue, but the user should be allowed to control the kind of information provided. Possibly, the standard system message should be less than one line; but, by keying a ?, the user should be able to obtain a few lines of explanation. ?? might yield a set of examples, and ??? might produce explanations of the examples and a complete description. Most application software offer a special HELP button to provide contest-sensitive explanations when the user needs assistance.

Some telephone companies, long used to dealing with nontechnical users, offer this tolerant message: "We're sorry, but we were unable to complete your call as dialed. Please hang up, check your number, or consult the operator for assistance." They take the blame and offer constructive guidance for what to do. A thoughtless programmer might have generated a harsher message: "Illegal telephone number. Call aborted. Error number 583-2R6.9. Consult your user manual for further information."

11.2.4   Appropriate physical format

Although professional programmers have learned to read uppercase-only text, most users prefer and find easier to read mixed uppercase and lowercase messages (Section 11.4). Uppercase-only messages should be reserved for brief, serious warnings. Messages that begin with a lengthy and mysterious code number serve only to remind the user that the designers were insensitive to the user's real needs. If code numbers are needed at all, they might be enclosed in parentheses at the end of a message.

There is disagreement about the optimal placement of messages in a display. One school of thought argues that the messages should be placed on the display near where the problem has arisen. A second opinion is that the messages clutter the display and should be placed in a consistent position on the bottom of the display. The third approach simply pops up a dialog box in the middle of the display, possibly obscuring the relevant part.

Some applications ring a bell or sound a tone when an error has occurred. This alarm can be useful if the operator could miss the error, but can be embarrassing if other people are in the room and is potentially annoying even if the operator is alone. The use of audio signals should be under user control.

The early high-level language Michigan Algorithmic Decoder (MAD) printed out a full-page picture of Alfred E. Neuman if syntactic errors were in the program. Novices enjoyed this playful approach, but after they had accumulated a drawer full of pictures, the portrait became an annoying embarrassment. Designers must walk a narrow path between calling attention to a problem and avoiding embarrassment to the user. Considering the wide range of experience and temperament in users, maybe the best solution is to offer the user control over the alternatives—this approaches coordinates well with the user-centered principle.

11.2.5   Development of effective messages

The designer's intuition can be supplemented by simple, fast, and inexpensive design studies with actual users and several alternative messages. If the project goal is to serve novice users, then ample effort must be dedicated to designing, testing, and implementing the user interface. This commitment must extend to the earliest design stages so that interfaces can be modified in a way that contributes to the production of specific error messages. Messages should be evaluated by several people and tested with suitable subjects (Isa et al., 1983). Messages should appear in user manuals and be given greater visibility. Records should be kept on the frequency of occurrence of each error. Frequent errors should lead to software modifications that provide better error handling, to improved training, and to revisions in user manuals.

Users may remember the one time when they had difficulties with a computer system rather than the 20 times when everything went well. Their strong reaction to problems in using computer systems comes in part from the anxiety and lack of knowledge that novice users have. This reaction may be exacerbated by a poorly designed, excessively complex system; by a poor manual or training experience; or by hostile, vague, or irritating system messages. Improving the messages will not turn a bad system into a good one, but it can play a significant role in improving the user's performance and attitude.

Five controlled experiments explored the influence of error messages on user performance (Shneiderman, 1982). In one study, COBOL syntax error messages were modified, and undergraduate novice users were asked to repair the COBOL statements. Messages with increased specificity generated 28-percent higher repair scores.

Subjects using a text editor with only a ? for an error message made an average of 10.7 errors, but made only 6.1 errors when they switched to an

editor offering brief explanatory messages. In another experiment, students corrected 4.1 out of 10 erroneous text-editor commands using the standard system messages. Using improved messages, the experimental group corrected 7.5 out of the 10 commands.

In a study of the comprehensibility of job-control–language error messages from two popular contemporary systems, students scored 2.9 and 3.8 out of 6, whereas students receiving improved messages scored 4.8. Subjective preferences also favored the improved messages.

Mosteller (1981) studied error patterns in IBM's MVS Job Entry Control Language by capturing actual runs in a commercial environment. Analysis of the 2073 errors resulted in specific suggestions for revisions to the error messages, parser, and command language. Remarkably, 513 of the errors were exact retries of the previous runs, confirming concerns over the persistence of errors when messages are poor. As improvements were made to the messages, Mosteller found lower error rates.

These early experiments support the contentions that improving messages can upgrade performance and result in greater job satisfaction. They have led to the following recommendations for system developers (Box 11.1):

1. *Increase attention to message design*   The wording of messages should be considered carefully. Technical writers or copy editors can be consulted about the choice of words and phrasing to improve both clarity and consistency.

2. *Establish quality control* Messages   should be approved by an appropriate quality-control committee consisting of programmers, users, and human-factors specialists. Changes or additions should be monitored and recorded.

3. *Develop guidelines*   Error messages should meet these criteria:

   - *Have a positive tone*   Indicate what must be done, rather than condemning the user for the error. Reduce or eliminate the use of such terms as ILLEGAL, INVALID, ERROR, or WRONG PASSWORD. Try Your password did not match the stored password. Please try again.
   - *Be specific and address the problem in the user's terms*   Avoid the vague SYNTAX ERROR or obscure internal codes. Use variable names and concepts known to the user. Instead of INVALID DATA in an inventory application, try Dress sizes range from 5 to 16.
   - *Place the users in control of the situation*   Provide them with enough information to take action. Instead of INCORRECT COMMAND, try Permissible commands are: SAVE, LOAD, or EXPLAIN.
   - *Have a neat, consistent, and comprehensible format*   Avoid lengthy numeric codes, obscure mnemonics, and cluttered displays.

Writing good messages—like writing good poems, essays, or advertisements—requires experience, practice, and a sensitivity to how the reader

Box 11.1

Error-message guidelines for the end product and for the development process. These guidelines are derived from practical experience and empirical data.

- Product
  - Be as specific and precise as possible.
  - Be constructive: Indicate what the user needs to do.
  - Use a positive tone: Avoid condemnation.
  - Choose user-centered phrasing.
  - Consider multiple levels of messages.
  - Maintain consistent grammatical form, terminology, and abbreviations.
  - Maintain consistent visual format and placement.

- Process
  - Establish a message quality-control group.
  - Include messages in the design phase.
  - Place all messages in a file.
  - Review messages during development.
  - Design the product to eliminate the need for most messages.
  - Carry out acceptance tests.
  - Collect frequency data for each message.
  - Review and revise messages over time.

will react. It is a skill that can be acquired and refined by programmers and designers who are intent on serving the user. However, perfection is impossible and humility is the mark of the true professional.

4. *Carry out usability tests*   System messages should be subjected to a usability test with an appropriate user community to determine whether they are comprehensible. The test could range from a rigorous experiment with realistic situations (for life-critical or high-reliability systems) to an informal reading and review by interested users (for personal computing or low-threat applications). Complex interactive systems that involve thousands of users are never really complete until they are obsolete. Under these conditions, the most effective designs emerge from iterative testing and evolutionary refinement (Chapter 4).

5. *Collect user-performance data*   Frequency counts should be collected for each error condition on a regular basis. If possible, the user's actions

should be captured for a more detailed study. If you know where users run into difficulties, you can then revise the message, improve the training, modify the manual, or change the interface. The error rate per 1000 actions should be used as a metric of system quality and a gauge of how improvements affect performance. An error-counting option is useful for internal systems and can be a marketing feature for software products.

Improved messages will be of the greatest benefit to novice users, and regular users and experienced professionals will also benefit. As examples of excellence proliferate, complex, obscure, and harsh systems will seem increasingly out of place. The crude environments of the past will be replaced gradually by systems designed with the user in mind. Resistance to such a transition should not be allowed to impede progress toward the goal of serving the growing user community.

11.3   Nonanthropomorphic Design

There is a great temptation to have computers "talk" as though they were people. It is a primitive urge that designers often follow, and that children and many adults accept without hesitation (Nass et al., 1994, 1995). Children accept human-like references and qualities for almost any object, from Humpty Dumpty to Tootle the Train. Adults reserve the *anthropomorphic* references for objects of special attraction, such as cars, ships, or computers.

The words and graphics in user interfaces can make important differences in people's perceptions, emotional reactions, and motivations. Attributions of intelligence, autonomy, free will, or knowledge to computers can deceive, confuse, and mislead users. The suggestion that computers can think, know, or understand may give users an erroneous model of how computers work and what the machines' capacities are. Ultimately, the deception becomes apparent, and users may feel poorly treated. Martin (1995/96) carefully traces the media impact of the 1946 ENIAC announcements: "Readers were given hyperbole designed to raise their expectations about the use of the new electronic brains. . . . This engendered premature enthusiasm, which then led to disillusionment and distrust of computers on the part of the public when the new technology did not live up to these expectations."

A second reason for using nonanthropomorphic phrasing is to clarify the differences between people and computers. Relationships with people are different from relationships with computers. Users operate and control computers, but they respect the unique identity and autonomy of individuals. Furthermore, users and designers must accept responsibility for misuse of computers rather than blaming the machine for errors. It is worrisome that,

in one study (Friedman, 1995), 24 of 29 computer-science students "attributed aspects of agency—either decision-making and/or intentions—to computers" and six "consistently held computers morally responsible for errors."

A third motivation is that, although an anthropomorphic interface may be attractive to some people, it can be anxiety producing for others. Some people express anxiety about using computers and believe that computers "make you feel dumb." Presenting the computer through the specific functions it offers may be a stronger stimulus to user acceptance than is promoting the fantasy that the computer is a friend, parent, or partner. As users become engaged, the computer becomes transparent, and they can concentrate on their writing, problem solving, or exploration. At the end, they have the experience of accomplishment and mastery, rather than the feeling that some magical machine has done their job for them.

Individual differences in the desire for internal locus of control will be important, but there may be an overall advantage to clearly distinguishing human abilities from computer powers for most tasks and users (Shneiderman, 1995). On the other hand, there are advocates of creating an anthropomorphic computer and of creating lifelike autonomous agents (Laurel, 1990; Maes, 1995). Apple Computer created a videotape in 1987, "The Knowledge Navigator," with a preppie bow-tied young male agent carrying out tasks for an environmental researcher. Some futurists celebrated this vision, but skeptics scorned the scenario as a deception; most viewers, meanwhile, seemed mildly amused. Advocates of anthropomorphic interfaces assume that human–human communication is an appropriate model for human operation of computers. It may be a useful starting point, but I find it hard to understand why some designers pursue the human imitation approach long after it becomes counterproductive. Mature technology has managed to overcome the *obstacle of animism*, which has been a trap for technologists for centuries (Mumford, 1934). A visit to the Museum of Automata in York, England, reveals the ancient sources and persistent fantasies of animated dolls and robotic toys.

Historical precedents of failed anthropomorphic bank tellers (Tillie the Teller, Harvey Wallbanker, BOB (Bank of Baltimore)) or abandoned talking automobiles and soda machines do not seem to register on some designers. The bigger-than-life-sized Postal Buddy was supposed to be cute and friendly while providing several useful automated services, but this pseudo-postal clerk was rejected by users after incurring costs of over $1 billion.

Empirical studies offer further evidence. In an experimental test with 26 college students, the anthropomorphic interface (HI THERE, JOHN! IT'S NICE TO MEET YOU, I SEE YOU ARE READY NOW) was seen as less honest than a mechanistic dialog (PRESS THE ENTER KEY TO BEGIN SESSION) (Quintanar et al., 1982). In this computer-assisted instruction task, subjects took longer with the anthropomorphic design, possibly contributing to the observed improved scores on a quiz, but the students felt less responsible for their performance.

In another study, a stern face and a neutral talking face were compared with a text-only display (Walker et al., 1994). The authors concluded that "incautiously adding human characteristics like face, voice, and facial expressions could make the experience for users worse rather than better." The designers generated the talking faces by texture mapping an image onto a geometric wire frame model to produce a 512×320 pixel face. The lip movements were synchronized with the voice-generation algorithm; the stern expression was produced by contraction of the corrugator muscles in the underlying physical model to pull the inner eyebrows in and down. The 42 experienced users rated the text-only version as statistically significantly more likable, friendly, comfortable, happy, less stiff, and less sad than the talking faces. Subjects also found the questions clearer and were more willing to continue with the text-only versions. Evidence in favor of the faces was that subjects in the face treatments produced fewer invalid answers and wrote lengthier commentaries, especially with the stern face. In a follow-up study to assess willingness to cooperate, subjects "kept their promises as much with a text-only computer as with a person, but less with a more human-like computer" (Kiesler et al., 1996).

A more elaborate computer-generated face (16 muscles and 10 parameters controlling 500 polygons) was compared with a three-dimensional arrow in guiding user attention to moves in a card game (Takeuchi and Naito, 1995). Although the face was appreciated as being "entertaining," the arrow was seen as "useful." The authors noted that "subjects tend to try to interpret facial displays and head behaviors. Such involvement prevents them from concentrating on the game" and led to fewer wins than subjects had in the arrow treatments.

Similar questions arise in the use of value judgments as reinforcement for correct answers in educational software. Our study with 24 third-grade students found that positive reinforcement with value-judgment phrases (EXCELLENT, THAT'S GOOD!, YOU'RE DOING GREAT, and so on) did not improve performance or satisfaction in an arithmetic drill-and-practice lesson. On the other hand, the presence of a simple numerical counter (6 CORRECT 2 INCORRECT) improved learning.

In a study with 36 junior–high-school students conducted by Lori Gay and Diane Lindwarm under my direction, the style of interaction was varied. Students received a computer-assisted instruction session in one of three forms:

1. *I:* HI! I am the computer. I am going to ask you some questions.
2. *You:* You will be answering some questions. You should. . . .
3. *Neutral:* This is a multiple-choice exercise.

Before and after the three sessions at the computer, subjects were asked to describe whether using a computer was "easy" or "hard." Most subjects ini-

tially thought that using a computer was "hard" and did not change their opinion. Of the seven who changed their minds, the five who moved toward "hard" were all using the *I* or *neutral* interface. The two subjects who moved toward "easy" were using the *you* interface. Performance measures on the tasks were not significantly different, but anecdotal evidence and the positive shift for the group that used *you* messages warrant further study.

A study of error-message wording found similar results with 49 business-school undergraduates (Resnik and Lammers, 1986). Subjects reported being less confused and nervous with constructive (Use letters only) than with human-like (I don't understand these numbers) or condemning (Numerics illegal) message tones.

These results suggest that anthropomorphic interfaces that use first-person pronouns may be counterproductive because they deceive, mislead, and confuse. It may seem cute on first encounter to be greeted by I am SOPHIE, the sophisticated teacher, and I will teach you to spell correctly. By the second session, however, this approach strikes people as uselessly repetitive; by the third session, it is an annoying distraction from the task.

The alternative for the software designer is to focus on the user and to use third-person singular pronouns or to avoid pronouns altogether; for example,

Poor: I will begin the lesson when you press RETURN.

Better: You can begin the lesson by pressing RETURN.

Better: To begin the lesson, press RETURN.

The *you* form seems preferable for introductory screens; however, once the session is underway, reducing the number of pronouns and words avoids distractions from the task. A travel-reservation task was carried out by 33 students with a simulated natural-language interface using the *I, you,* or *neutral* styles (called *anthropomorphic, fluent,* and *telegraphic* by the authors; Brennan and Ohaeri, 1994). Users' messages mimicked the style of messages they received, leading to lengthier user inputs and longer task completion times in the anthropomorphic treatment. Users did not attribute greater intelligence to the anthropomorphic computer.

Some designers of children's educational software believe that it is appropriate and acceptable to have a fantasy character, such as a teddy bear or busy beaver, serve as a guide through a lesson. A cartoon character can be drawn on the screen and possibly animated, adding visual appeal. Successful software packages such as Reader Rabbit provide support for this position. Unfortunately, cartoon characters were not successful in the heavily promoted, but short-lived, home-computing product from Microsoft called *BOB*. Users could choose from a variety of on-screen characters who spoke in cartoon bubbles with phrases such as: What a team we are, What shall

we do next, Ben? and Good job so far, Ben. This style might be accept-
able in children's games and educational software, but is probably not
acceptable for adults performing meaningful tasks.

A more likely approach is to identify the human author of a lesson or soft-
ware package, and to allow that person to speak to the reader, much as tele-
vision news announcers speak to the viewer. Instead of making the computer
into a person, designers can show identifiable and appropriate personalities.
For example, President Clinton might welcome visitors to a web site about
the White House, or Bill Gates might provide a greeting for new users of
Windows.

Once past these introductions, several styles are possible. One is a contin-
uation of the guided-tour metaphor, in which the respected personality
introduces segments, but allows users to control the pace, to repeat seg-
ments, and to decide when they are ready to move on. This approach works
for museum tours, tutorials on software, and certain educational lectures. A
second strategy is to support user control by showing an overview of the
modules from which users can choose. Users decide how much time to
spend visiting parts of museums, browsing a timeline with details of events,
or jumping among articles in hyperlinked encyclopedia.

These overviews give users a sense of the magnitude of information avail-
able and allow them to see their progress in covering the topics. Overviews
also support users' needs for closure and the satisfaction of completely tour-
ing the contents. Overviews also offer a comprehensible environment with
predictable actions that foster a comforting sense of control. Furthermore,
they support the need for replicability of actions (to revisit an appealing or
confusing module, or to show it to a colleague) and reversibility to return to
a known landmark. By contrast, game designers have long understood the
challenge of confusion, hidden controls, and unpredictability, but games are
certainly different from most applications. A summary of nonanthropomor-
phic guidelines appears in Box 11.2.

11.4   Display Design

For most interactive systems, the displays are a key component to successful
designs, and are the source of many lively arguments. Dense or cluttered dis-
plays can provoke anger, and inconsistent formats can inhibit performance.
The complexity of this issue is suggested by the 162 guidelines for data display
offered by Smith and Mosier (1986). This diligent effort (see Box 11.3 for exam-
ples) represents progress over the vague guidelines given in earlier reviews.
Display design will always have elements of art and require invention, but per-

Box 11.2

Guidelines for avoiding anthropomorphism and building appealing interfaces.

Nonanthropomorphic Guidelines
- Avoid presenting computers as people.
- Choose appropriate humans for introductions or guides.
- Use caution in designing computer-generated human faces or cartoon characters.
- Use cartoon characters in games or children's software, but usually not elsewhere.
- Design comprehensible, predictable, and controllable interfaces.
- Provide user-centered overviews for orientation and closure.
- Do not use "I" when the computer responds to human actions.
- Use "you" to guide users, or just state facts.

---

ceptual principles are becoming clearer (Tullis, 1988a, 1988b; Tufte, 1990; Marcus, 1992; Galitz, 1994), and theoretical foundations are emerging (Mackinlay, 1986; Casner, 1991; Lohse, 1991), and are being applied in research prototypes (Roth et al., 1994). Innovative information visualizations with user interfaces to support dynamic control is a rapidly emerging theme (Chapter 15).

Designers should begin, as always, with a thorough knowledge of the users' tasks, free from the constraints of display size or available fonts. Effective display designs must provide all the necessary data in the proper sequence to carry out the task. Meaningful groupings of items (with labels suitable to the user's knowledge), consistent sequences of groups, and orderly formats all support task performance. Groups can be surrounded by blank spaces or boxes. Alternatively, related items can be indicated by highlighting, background shading, color, or special fonts. Within a group, orderly formats can be accomplished by left or right justification, alignment on decimal points for numbers, or markers to decompose lengthy fields.

Graphic designers have produced principles suited to print formats, and are now adapting these principles for display design. Mullet and Sano (1995) offer thoughtful advice with examples of good and bad design in commercial systems. They propose six categories of principles that reveal the complexity of the designer's task:

1. *Elegance and simplicity*   Unity, refinement, and fitness
2. *Scale, contrast, and proportion*   Clarity, harmony, activity, and restraint
3. *Organization and visual structure*   Grouping, hierarchy, relationship, and balance

Box 11.3

Samples of the 162 data-display guidelines from Smith and Mosier (1984).

- Ensure that any data that a user needs, at any step in a transaction sequence, are available for display.
- Display data to users in directly usable form; do not require that users convert displayed data.
- Maintain consistent format, for any particular type of data display, from one display to another.
- Use short, simple sentences.
- Use affirmative statements, rather than negative statements.
- Adopt a logical principle by which to order lists; where no other principle applies, order lists alphabetically.
- Ensure that labels are sufficiently close to their data fields to indicate association, yet are separated from their data fields by at least one space.
- Left-justify columns of alphabetic data to permit rapid scanning.
- Label each page in multipaged displays to show its relation to the others.
- Begin every display with a title or header, describing briefly the contents or purpose of the display; leave at least one blank line between the title and the body of the display.
- For size coding, make larger symbols be at least 1.5 times the height of the next-smaller symbol.
- Consider color coding for applications in which users must distinguish rapidly among several categories of data, particularly when the data items are dispersed on the display.
- When you use blink coding, make the blink rate 2 to 5 Hz, with a minimum duty cycle (ON interval) of 50 percent.
- For a large table that exceeds the capacity of one display frame, ensure that users can see column headings and row labels in all displayed sections of the table.
- Provide a means for users (or a system administrator) to make necessary changes to display functions, if data-display requirements may change (as is often the case).

4. *Module and program*   Focus, flexibility, and consistent application
5. *Image and representation*   Immediacy, generality, cohesiveness, and characterization
6. *Style*   Distinctiveness, integrity, comprehensiveness, and appropriateness

This section deals with a fraction of the issues, and offers empirical support for concepts where available.

11.4.1   Field layout

Exploration with a variety of layouts can be a helpful process. These design alternatives should be developed directly on a display screen. An employee record with information about a spouse and children could be displayed crudely as

Poor: TAYLOR, SUSAN034787331WILLIAM TAYLOR
THOMAS102974ANN082177ALEXANDRA090872

This record may contain the necessary information for a task, but extracting the information will be slow and error-prone. As a first step at improving the format, blanks and separate lines can distinguish fields:

Better: TAYLOR, SUSAN 034787331   WILLIAM TAYLOR
THOMAS 102974
ANN 082177
ALEXANDRA 090872

The children's names can be listed in chronological order, with alignment of the dates. Familiar separators for the dates and the employee's social-security number also aid recognition:

Better: TAYLOR, SUSAN 034-78-7331   WILLIAM TAYLOR
ALEXANDRA 09-08-72
THOMAS   10-29-74
ANN   08-21-77

The reversed order of "last name, first name" for the employee may be desired to highlight the lexicographic ordering in a long file. However, the "first name, last name" order for the spouse is usually more readable. Consistency seems important, so a compromise might be made to produce

Better: SUSAN TAYLOR   034-78-7331   WILLIAM TAYLOR
ALEXANDRA 09-08-72
THOMAS   10-29-74
ANN   08-21-77

For frequent users, this format may be acceptable, since labels have a cluttering effect; for most users, however, labels will be helpful:

Better: Employee: SUSAN TAYLOR Social Security
Number: 034-78-7331
Spouse: WILLIAM TAYLOR
Children: Names   Birthdates
ALEXANDRA 09-08-72
THOMAS   10-29-74
ANN   08-21-77

Lowercase letters have been used for labels, but the coding might be switched with bold face for the contents. The lengthy label for social-security number might be abbreviated if the users are knowledgeable. Indenting the information about children might help to convey the grouping of these repeating fields:

```
Better:  EMPLOYEE: Susan Taylor     SSN: 034-78-7331
         SPOUSE: William Taylor
         CHILDREN:
            NAMES      BIRTHDATES
         Alexandra  09-08-72
         Thomas     10-29-74
         Ann        08-21-77
```

Finally, if boxes are available, then an orderly pattern is sometimes more appealing, although it may consume more screen space:

```
Better:  ┌─────────────────────────────────────────────┐
         │ EMPLOYEE: Susan Taylor SSN: 034-78-7331      │
         │                                              │
         │ SPOUSE: William Taylor                       │
         ├─────────────────────────────────────────────┤
         │ CHILDREN                                     │
         │ NAMES              BIRTHDATES                │
         ├─────────────────────────────────────────────┤
         │ Alexandra          09-08-72                  │
         │ Thomas             10-29-74                  │
         │ Ann                08-21-77                  │
         └─────────────────────────────────────────────┘
```

Even in this simple example, the possibilities are numerous. In any situation, a variety of designs should be explored. Further improvements could be made with other coding strategies, such as background shading, color, and graphic icons. An experienced graphic designer can be a great benefit to the design team. Pilot testing with prospective users can yield subjective satisfaction scores and objective times to complete tasks plus error rates for a variety of proposed formats.

11.4.2 Empirical results

A few empirical tests of alternative display designs have been conducted. A narrative form (Fig. 11.1a), taken from a telephone-line–testing program, was replaced with a structured form (Fig. 11.1b) (Tullis, 1981). The structured form eliminated unnecessary information, grouped related information, and emphasized the information relevant to the required tasks. After practice in reading these displays, Bell System employees were required to carry out typical tasks. The narrative form required an average of 8.3 seconds per task, whereas the structured form took only 5.0 seconds, resulting in an estimated saving of 79 person-years over the life of the system.

A NASA study with space-shuttle displays demonstrated that improving the data labels, clustering related information, using appropriate indentation and underlining, aligning numeric values, and eliminating extraneous characters could improve performance (Burns et al., 1986). Task times were reduced by 31 percent and error rates by 28 percent for a population of 16 technical and clerical employees at NASA and Lockheed who were unfamiliar with either version. Sixteen experts with the existing system did not perform statistically significantly faster with the improved displays, but they did perform significantly more accurately. A follow-up study validated the benefit of redesign and showed that appropriate highlighting further reduced search times (Donner et al., 1991).

Expert users can deal with dense displays and may prefer these displays because they are familiar with the format and they must initiate fewer actions. Performance times are likely to be shorter with fewer, but denser displays than with more numerous but sparse displays. This improvement will be especially marked if tasks require comparison of information across displays. Systems for stock-market data, air-traffic control, and airline reservations are examples of successful applications that have dense packing, limited labels, and highly coded fields.

In a study of 12 telephone operators, Springer (1987) found that suppressing the presentation of redundant family names in a directory-assistance listing reduced target-location time by 0.8 seconds. She also found that, when the target was in the upper quarter of the display, users found it more quickly if the screen were only one-quarter full, as opposed to one-half full or completely full. This result suggests that screen contents should contain only task-relevant information, and that extraneous information does slow performance.

In another study, 110 nurses were shown laboratory reports of blood tests in the standard commercial format of three screens, in a compressed

```
TEST RESULTS    SUMMARY: GROUND

   GROUND, FAULT T-G
   3 TERMINAL DC RESISTANCE
      >   3500.00 K OHMS T-R
      =     14.21 K OHMS T-G
      >   3500.00 K OHMS R-G
   3 TERMINAL DC VOLTAGE
      =      0.00 VOLTS   T-G
      =      0.00 VOLTS   R-G
   VALID AC SIGNATURE
   3 TERMINAL AC RESISTANCE
      =      8.82 K OHMS T-R
      =     14.17 K OHMS T-G
      =    628.52 K OHMS R-G
   LONGITUDINAL BALANCE POOR
      =     39    DB
   COULD NOT COUNT RINGERS DUE TO
     LOW RESISTANCE
   VALID LINE CKT CONFIGURATION
   CAN DRAW AND BREAK DIAL TONE
```

(a)

```
        ***********************************
        *                                 *
        *   TIP GROUND         14 K       *
        *                                 *
        ***********************************

DC RESISTANCE      DC VOLTAGE        AC SIGNATURE

3500 K T-R                              9 K T-R
  14 K T-G         0 V T-G            14 K T-G
3500 K R-G         0 V R-G           629 K R-G

   BALANCE                          CENTRAL OFFICE

   39 DB                            VALID LINE CKT
                                    DIAL TONE OK
```

(b)

Figure 11.1

Two versions of screens in a Bell Laboratories study (Tullis, 1981). (a) The narrative format. (b) The structured format.

two-screen version, and in a densely packed one-screen version (Staggers. 1993). Search times dropped in half (approximately) over the five trial blocks for novice and experienced nurses, demonstrating a strong learning effect. The dramatic performance result was that search times were longest on the three-screen version (9.4 seconds per task) compared to the densely packed one-screen version (5.3 seconds per task) (Fig. 11.2). The high cost of turning pages and of reorienting to the new material appears to be far more destructive of concentration than scanning dense displays. Accuracy and subjective satisfaction were not significantly different across the three versions.

Every guideline document implores designers to preserve consistent location, structure, and terminology across displays. Supportive evidence for consistent location comes from a study of 40 inexperienced computer users of a menu system (Teitelbaum and Granda, 1983). The position of the title, page number, topic heading, instruction line, and entry area were varied across displays for one-half of the subjects, whereas the other half saw constant positions. Mean response time to questions about these items for subjects in the varying condition was 2.54 seconds, but was only 1.47 seconds for those seeing constant positions. A student project with 60 experienced computer users showed similar benefits from consistent placement, size, and color of buttons in GUIs.

Sequences of displays should be similar throughout the system for similar tasks, but exceptions will certainly occur. Within a sequence, users should be offered some sense of how far they have come and how far they have to go to reach the end. It should be possible to go backward in a sequence to correct errors, to review decisions, or to try alternatives.

11.4.3   Display-complexity metrics

Although knowledge of the users' tasks and abilities is the key to designing effective screen displays, an objective, automatable metric of screen complexity is an attractive aid. After a thorough review of the literature, Tullis (1988a) developed four task-independent metrics for alphanumeric displays:

1. *Overall density*   Number of filled character spaces as a percentage of total spaces available
2. *Local density*   Average number of filled character spaces in a five-degree visual angle around each character, expressed as a percentage of available spaces in the circle and weighted by distance from the character

Low Density Screens

```
Patient Laboratory Inquiry     Large University Medical Center     Pg 1 of 3

Robinson, Christopher   #XXX-20-4627   Unit: 5E, 5133D   M/13   Ph:301-XXX-5885

        <CBC>       Result      Normal    Range        Units
        ----------------------------------------------------------
11/20   Wbc          5.0         4.8  -   10.8         th/cumm
22:55   Rbc          4.78        4.7  -    6.1         m/cumm
        Hgb         12.8        14.0  -   18.0         g/dL
        Hct         37.9        42.0  -   52.0         %
        Plt        163.0       130.0  -  400.0         th/cumm
        Mcv         88.5        82.0  -  101.0         fL
        Mch         30.6        27.0  -   34.0         picogms
        Mchc        34.6        32.0  -   36.0         g/dL
        Rdw         14.5        11.5  -   14.5         %
        Mpv          9.3         7.4  -   10.4         fL
        ----------------------------------------------------------
        Key:   * = abnormal
                                                  PgDn for more


Patient Laboratory Inquiry     Large University Medical Center     Pg 2 of 3

Robinson, Christopher   #XXX-20-4627   Unit: 5E, 5133D   M/13   Ph:301-XXX-5885

        <DIFF>       Result      Norm Range        Unit
        ----------------------------------------------------------
11/20   Segs         35          34  - 75           %
22:55   Bands         5           0  -  9           %
        Lymphs       33          10  - 49           %
        Monos        33           2  - 14           %
        Eosino        5           0  -  8           %

        Baso          2           0  -  2           %
        Atyplymph    20           0  -  0           %
        Meta          0           0  -  0           %
        Myleo         0           0  -  0           %
        Platelets(estimated)                        adeq
        ----------------------------------------------------------
        Key:   * = Abnormal
                                                  PgDn for more


Patient Laboratory Inquiry     Large University Medical Center     Pg 3 of 3

Robinson, Christopher   #XXX-20-4627   Unit: 5E, 5133D   M/13   Ph:301-XXX-5885

11/20   22:55

<MORPHOLOGY      Macrocytosis  1+   Basophilic Stippling 1+   Toxic Gran Occ
Hypochromia 1+  Polychromasia 1+   Target Cells         3+   Normocytic  No
Key: * = Abnormal Priority:  Routine          Acc#: 122045-015212
Ordered by: Holland, Daniel on 10/22/91, 10:00       Ord#: 900928-HH1131
   Personal Data   -   PRIVACY ACT OF 1974   (PL   93-579)
       End of report
```

Figure 11.2

In a study with 110 nurses, results showed an average task time of 9.4 seconds with the low-density version, versus 5.3 seconds with the high-density version (Staggers, 1993).

High Density Screen

```
Patient Laboratory Inquiry        Large University Medical Center      Pg 1 of 1
```

Robinson, Christopher   #XXX-20-4627   Unit: 5E, 5133D   M/13   Ph:301-XXX-5885

	<CBC>	Result	Normal Range	Units	<DIFF>	Result	Norm Range	Unit
10/23	Wbc	5.0	4.8 - 10.8	th/cumm	Segs	40	34 - 74	%
0600	Rbc	4.78	4.7 - 6.1	m/cumm	Bands	5	0 - 9	%
	Hgb	15.1	14.0 - 18.0	g/dL	Lymphs	33	10 - 49	%
	Hct	47.9	42.0 - 52.0	%	Monos	10	2 - 14	%
	Plt	163.0	130.0 - 400.0	th/cumm	Eosino	5	0 - 8	%
	Mcv	88.5	82.0 - 101.0	fL	Baso	2	2 - 2	%
	Mch	30.6	27.0 - 34.0	picogms	Atyplymph	0	0 - 0	%
	Mchc	34.6	32.0 - 36.0	g/dL	Meta	0	0 - 0	%
	Rdw	14.5	11.5 - 14.5	%	Myelo	0	0 - 0	%
	Mpv	8.3	7.4 - 10.4	fL		Plt	(estm)	adeq

```
<MORPHOLOGY       Macrocytosis  1+    Basophilic Stippling  1+    Toxic Gran Occ
Hypochromia 1+    Polychromasia 1+    Target Cells          3+    Normocytic  No
```

```
Key: * = Abnormal Priority:  Routine       Acc#: 122045-015212
Ordered by:  Holland, Daniel on 10/22/91, 10:00        Ord#: 900928-HH1131
   Personal Data  -  PRIVACY ACT OF 1974  (PL  93-579)
```

Figure 11.2 *(continued)*

3. *Grouping* (1)   Number of groups of "connected" characters, where a connection is any pair of characters separated by less than twice the mean of the distances between each character and its nearest neighbor; (2) average visual angle subtended by groups, and weighted by number of characters in the group

4. *Layout complexity*   Complexity, as defined in information theory, of the distribution of horizontal and vertical distances of each label and data item from a standard point on the display

The argument for local density emerges from studies of visual perception indicating that concentration is focused in a five-degree visual angle. At normal viewing distances from displays, this area translates into a circle approximately 15 characters wide and seven characters high. Lower local and overall densities should yield easier-to-read displays. The grouping metric was designed to yield an objective, automatable value that assesses the number of clusters of fields on a display. Typically, clusters are formed by characters that are separated by no more than one intervening space horizontally and that are on adjacent lines. Layout complexity measures the variety of shapes that confront the user on a display. Neat blocks of fields that start in the same column will have a lower layout complexity. These metrics do not account for coding techniques, uppercase versus lowercase characters, continuous text, graphics, or multidisplay issues.

Ten Bell Laboratories employees did motel- and airline-information retrieval tasks on 520 different displays in a variety of formats (Fig. 11.3). Performance times and subjective evaluations were collected to generate a predictive equation. The efficacy of the predictor equations for performance times and subjective ratings were validated in a second study, in which 14 Bell Laboratories employees did author- and book-information retrieval tasks on 150 displays using 15 different display formats (Fig. 11.4). Correlations between predicted and actual values were 0.80 for search times and 0.79 for subjective ratings.

This impressive result is encouraging; however, the metrics require a computer program to do the computations on the alphanumeric-only displays, and they do not include coding techniques, user-experience levels, or multi-display considerations. Tullis is cautious in interpreting the results and emphasizes that displays that optimize search times do not necessarily optimize subjective ratings. Grouping of items led to fast performance, but high subjective ratings were linked to low local density and low layout complexity. A simple interpretation of these results is be that effective display designs contain a middle number of (six to 15) groups that are neatly laid out, surrounded by blanks, and similarly structured. This conclusion is a satisfying confirmation of a principle that, when stated, seems intuitively obvious, but has not emerged explicitly in the numerous guidelines documents. Further study of human visual search strategies would be helpful in preparing design guidelines (Treisman, 1982).

A more accurate prediction of performance is likely to come with metrics that integrate task frequencies and sequences. Sears' (1993) developed a task-dependent metric called *layout appropriateness* to assess whether the spatial layout is in harmony with the users' tasks (Fig. 11.5). If users can accomplish frequent tasks by moving through a display in a top-to-bottom pattern, then faster performance is likely, compared to that with a layout that requires numerous jumps around widely separated parts of the display. Layout appropriateness is a widget-level metric that deals with buttons, boxes, and lists. Designers specify the sequences of selections that users make and the frequencies for each sequence. Then, the given layout of widgets is evaluated by how well it matches the tasks. An optimal layout that minimizes visual scanning can be produced, but since it may violate user expectations about positions of fields, the designers must make the final layout decisions.

Layouts in which related information was clustered were found to benefit users when the cognitive load on working memory was large. Accuracy increased when related items were clustered, thus reducing the scanning needed to locate distant items (Vincow and Wickens, 1993).

```
To: Atlanta, GA

     Departs    Arrives    Flight

Asheville, NC          First: $92.57    Coach: $66.85
     7:20a      8:05a      PI 299
    10:10a     10:55a      PI 203
     4:20p      5:00p      PI 259

Austin, TX             First: $263.00   Coach: $221.00
     8:15a     11:15a      EA 530
     8:40a     11:39a      DL 212
     2:00p      5:00p      DL 348
     7:15p     11:26p      DL 1654

Baltimore, MD          First: $209.00   Coach: $167.00
     7:00a      8:35a      DL 1767
     7:50a      9:32a      EA 631
     8:45a     10:20a      DL 1610
    11:15a     12:35p      EA 147
     1:35p      3:10p      DL 1731
     2:35p      4:16p      EA 141
```

(a)

```
To: Knoxville, TN
Atlanta, GA   Dp: 9:28a   Ar: 10:10a   Flt: DL 1704   1st: 97.00   Coach: 86.00
Atlanta, GA   Dp: 12:28p  Ar: 1:10p    Flt: DL 152    1st: 97.00   Coach: 86.00
Atlanta, GA   Dp: 4:58p   Ar: 5:40p    Flt: DL 418    1st: 97.00   Coach: 86.00
Atlanta, GA   Dp: 7:41p   Ar: 8:25p    Flt: DL 1126   1st: 97.00   Coach: 86.00
Chicago, Ill.   Dp: 1:45p   Ar: 5:39p    Flt: AL 58    1st: 190.00  Coach: 161.00
Chicago, Ill.   Dp: 6:30p   Ar: 9:35p    Flt: DL 675   1st: 190.00  Coach: 161.00
Chicago, Ill.   Dp: 6:50p   Ar: 9:55p    Flt: RC 398   1st: 190.00  Coach: 161.00
Cincinnati, OH  Dp: 12:05p  Ar: 1:10p   Flt: FW 453   1st: 118.00  Coach: 66.85
Cincinnati, OH  Dp: 5:25p   Ar: 6:30p   Flt: FW 455   1st: 118.00  Coach: 66.85
Dallas, TX    Dp: 5:55p   Ar: 9:56p    Flt: AL 360   1st: 365.00  Coach: 215.00
Dayton, OH    Dp: 11:20a  Ar: 1:10p    Flt: FW 453   1st: 189.00  Coach: 108.00
Dayton, OH    Dp: 4:40    Ar: 6:30p    Flt: FW 455   1st: 189.00  Coach: 108.00
Detroit, Mich.  Dp: 9:10a   Ar: 1:10p   Flt: FW 453   1st: 183.00  Coach: 106.00
Detroit, Mich.  Dp: 2:35p   Ar: 6:30p   Flt: FW 455   1st: 183.00  Coach: 106.00
```

(b)

Figure 11.3

Two versions of screens from the first experiment by Tullis (1984). (a) A structured format that leads to superior performance and preference. (b) Unstructured format. The results of this experiment led to predictive equations.

```
Books

Author:     Aird, C
Author#:    33
Title:      Henrietta Who?
Price:      $5
Publisher:  Macmillan
#Pages:     253

Author:     Aird, C
Author#:    33
Title:      His Burial Too
Price:      $4
Publisher:  Macmillan
#Pages:     287

Author:     Aird, C
Author#:    33
Title:      Late Phoenix
Price:      $8
Publisher:  McGraw
#Pages:     362
```

(a)

```
Books

Silverberg,R    #112    Downward to the Earth    $8    McGraw    314p

Silverberg,R    #112    Dying Inside    $6    McGraw    284p

Silverberg,R    #112    Earth's Other Shadow    $4    Harper    295p

Silverberg,R    #112    Invaders from Earth    $3    McGraw    302p

Silverberg,R    #112    Lord Valentine's Castle    $12    Macmillan    354p

Silverberg,R    #112    Man in the Maze    $7    McGraw    322p

Springer, N     #204    Sable Moon    $3    Prentice    185p

Springer, N     #204    Silver Sun    $4    Norton    198p

Springer, N     #204    White Hart    $5    Prentice    215p

Stewart, M      #64    Crystal Cave    $11    McGraw    428p

Stewart, M      #64    Hollow Hills    $8    Macmillan    403p
```

(b)

Figure 11.4

Two versions of screens in the second experiment by Tullis (1984). Equations based on objective metrics predicted performance and preference scores accurately, indicating the superiority of version (a) over version (b).
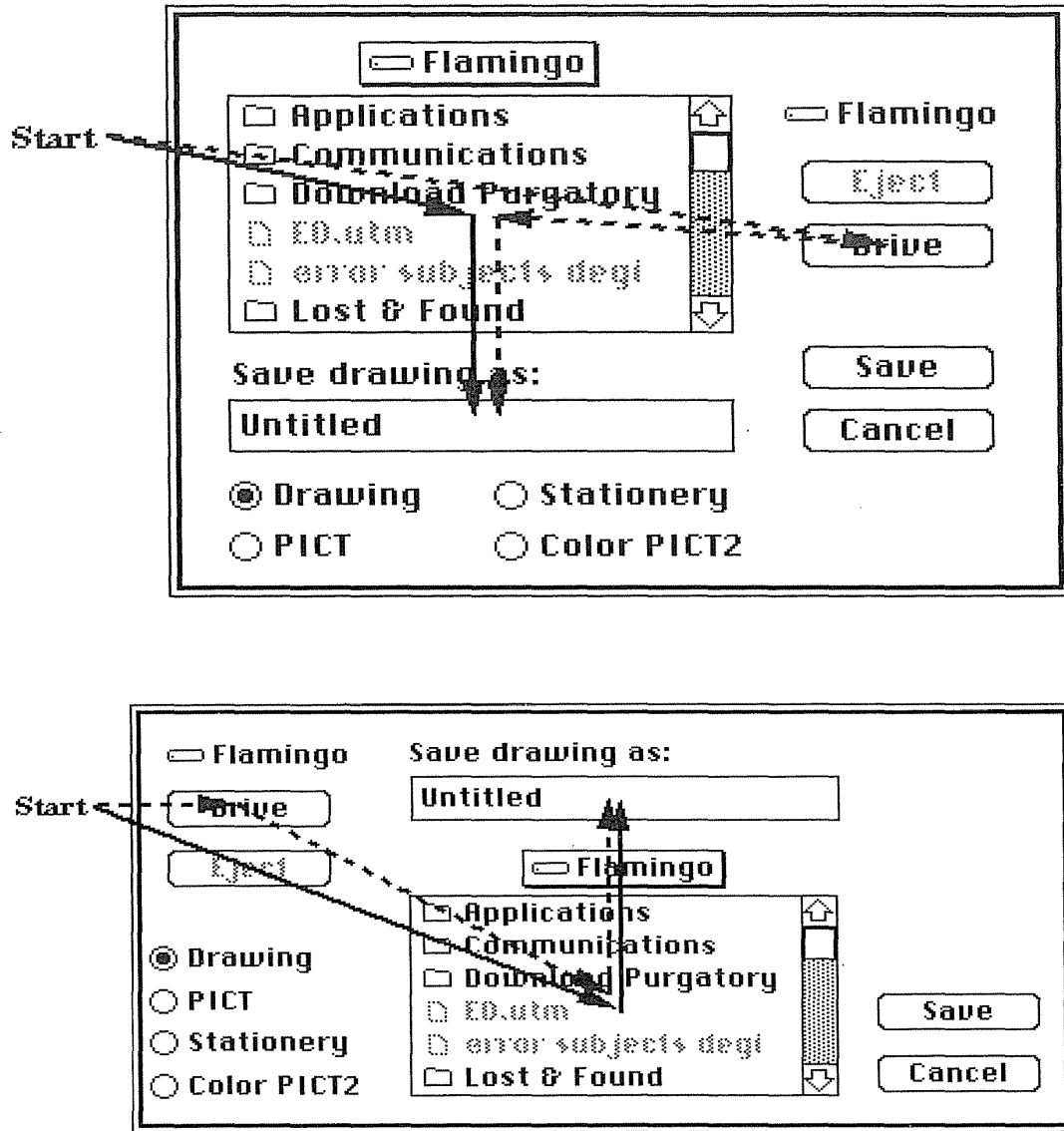
Figure 11.5

Layout appropriateness can help designers to analyze and redesign dialog boxes. (a) The existing dialog box. (b) The redesigned versions based on frequencies of action sequences. The solid line represents the most frequent sequence of actions; the dashed line represents the second most frequent sequence of actions. (Sears, 1993.)

11.5   Color

Color displays are attractive to users and can often improve task performance, but the danger of misuse is high. Color can

- Soothe or strike the eye
- Add accents to an uninteresting display
- Facilitate subtle discriminations in complex displays
- Emphasize the logical organization of information
- Draw attention to warnings
- Evoke strong emotional reactions of joy, excitement, fear, or anger

The principles developed by graphic artists for using color in books, magazines, highway signs, and television are being adapted for computer displays (Thorell and Smith, 1990; Travis, 1991; Marcus, 1992; Shubin et al., 1996). Programmers and interactive-systems designers are learning how to create effective computer displays and to avoid the pitfalls (Weitzman, 1985; Brown, 1988; Salomon, 1990; Galitz, 1994) (see Color Plates for examples).

There is no doubt that color makes video games more attractive to users, conveys more information on power-plant or process-control diagrams, and is necessary for realistic images of people, scenery, or three-dimensional objects (Foley et al., 1990; Gardiner, 1994). These applications require color. Greater controversy exists about the benefits of color for alphanumeric displays, spreadsheets, graphs, and user-interface components. High-resolution displays with multiple type fonts, sizes, and styles offer designers many options with black-on-white displays, which have been shown to be more readable than white-on-black (Snyder et al., 1990) however, color still has powerful attractions.

No simple set of rules governs use of color, but these guidelines are a starting point for designers:

- *Use color conservatively*   Many programmers and novice designers are eager to use color to brighten up their displays, but the results are often counterproductive. One home information system had the seven letters in its name in large letters, each with a different color. At a distance, the display appeared inviting and eye-catching; up close, however, it was difficult to read.

    Instead of showing meaningful relationships, inappropriately colored fields mislead users into searching for relationships that do not exist. In one poorly designed display, white lettering was used for input fields and for explanations of PF (Programmed Function) keys, leading users to think that they had to type the letters PF3 or PF9.

Using a different color for each of 12 items in a menu produces an overwhelming effect. Using four colors (such as red, blue, green, and yellow) for the 12 items will still mislead users into thinking that all the similarly colored items are related. An appropriate strategy would be to show all the menu items in one color, the title in a second color, the instructions in a third color, and error messages in a fourth color. Even this strategy can be overwhelming if the colors are too striking visually. A safe approach is always to use black letters on a white background, with italics or bold for emphasis, and to reserve color for special highlighting.

- *Limit the number of colors*   Many design guides suggest limiting the number of colors in a single alphanumeric display to four, with a limit of seven colors in the entire sequence of displays. Experienced users may be able to benefit from a larger number of color codes.

- *Recognize the power of color as a coding technique*   Color speeds recognition for many tasks, and is more effective than texture coding (Perlman and Swan, 1993). However, color coding can inhibit performance of tasks that go against the grain of the coding scheme. For example, in an accounting application, if data lines with accounts overdue more than 30 days are coded in red, they will be readily visible among the nonoverdue accounts coded in green. In air-traffic control, high-flying planes might be coded differently from low-flying planes to facilitate recognition. In programming workstations, newly added programming-language statements might be coded differently from the old statements, to show progress in writing or maintaining programs.

- *Ensure that color coding supports the task*   If, in the accounting application with color coding by days overdue, the task is now to locate accounts with balances of more than $55, the coding by days overdue may inhibit performance on the second task. In the programming application, the coding of recent additions may make it more difficult to read the entire program. Designers should attempt to make a close linkage between the users' tasks and the color coding.

- *Have color coding appear with minimal user effort*   In general, the color coding should not have to be assigned by the users each time that they perform a task, but rather should appear because they, for example, initiate the program to check for accounts overdue by more than 30 days. When the users perform the task of locating accounts with balances of more than $55, the new color coding should appear automatically.

- *Place color coding under user control*   When appropriate, the users should be able to turn off the color coding. For example, if a spelling checker color codes possibly misspelled words in red, then the user should be able to accept or change the spelling and to turn off the coding. The

presence of the highly visible red coding is a distraction from reading the text for comprehension.

- *Design for monochrome first*   The primary goal of a display designer should be to lay out the contents in a logical pattern. Related fields can be shown by contiguity or by similar structural patterns; for example, successive employee records may have the same indentation pattern. Related fields can also be grouped by a box drawn around the group. Unrelated fields can be kept separate by blank space—at least one blank line vertically or three blank characters horizontally. It may be advantageous to design for monochrome because color displays may not be universally available.

- *Consider the needs of color-deficient users*   Approximately 8 percent of North American and European users have some color deficiency in their vision. The most common deficiency is red–green blindness, in which both of these colors appear gray. Black on white or white on black will work for these and most other users.

- *Use color to help in formatting*   In densely packed displays where space is at a premium, similar colors can be used to group related items. For example, in a police dispatcher's tabular display of assignments, the police cars on emergency calls might be coded in red, and the police cars on routine calls might be coded in green. Then, when a new emergency arose, it would be relatively easy to identify the cars on routine calls and to assign one to the emergency. Dissimilar colors can be used to distinguish physically close but logically distinct fields. In a block-structured programming language, designers could show the nesting levels by coding the statements in a progression of colors—for example, dark green, light green, yellow, light orange, dark orange, red, and so on.

- *Be consistent in color coding*   Use the same color-coding rules throughout the system. If some error messages are displayed in red, then make sure that every error message appears in red; a change to yellow may be interpreted as a change in importance of the message. If colors are used differently by several designers of the same system, then users will hesitate as they attempt to assign meaning to the color changes. A set of color-coding standards should be written down for the benefit of every designer.

- *Be alert to common expectations about color codes*   The designer needs to speak to users to determine what color codes are applied in the task domain. From automobile-driving experience, red is commonly considered to indicate stop or danger, yellow is a warning, and green is go. In investment circles, red is a financial loss and black is a gain. For chemical engineers, red is hot and blue is cold. For map makers, blue means water, green means forests, and yellow means deserts. These multiple conventions can cause problems for designers. A designer

might consider using red to signal that an engine is warmed up and ready, but a user might understand the red coding as an indication of danger. A red light is often used to indicate power ON for electrical equipment, but some users are made anxious by this decision since red has a strong association with danger or stopping. When appropriate, indicate the color-code interpretations on the display or in a help panel.

- *Be alert to problems with color pairings*   If saturated (pure) red and blue appear on a display at the same time, it may be difficult for users to absorb the information. Red and blue are on the opposite ends of the spectrum, and the muscles surrounding the human eye will be strained by attempts to produce a sharp focus for both colors simultaneously. The blue will appear to recede and the red will appear to come forward. Blue text on a red background would present an especially difficult challenge for users to read. Similarly, other combinations will appear to be garish and difficult to read—for example, yellow on purple, magenta on green. Too little contrast also is a problem: Imagine yellow letters on a white background or brown letters on a black background. On each color monitor, the color appears differently, and careful tests with various text and background colors are necessary. Pace (1984) tested 24 color combinations using 36 undergraduate subjects. He found that error rates ranged from approximately one to four errors per 1000 characters read. Black on blue and blue on white were two color schemes associated with low error rates in both tasks, and magenta on green and green on white were two color schemes associated with high error rates. Tests with other monitors and tasks are necessary to reach a general conclusion about the most effective color pairs.

- *Use color changes to indicate status changes*   If an automobile speedometer had a digital readout of the driving speed, it might be helpful to change from green numbers below the maximum speed limit to red above the maximum speed limit to act as a warning. Similarly, in an oil refinery, pressure indicators might change color as the value went above or below acceptable limits. In this way, color acts as an attention-getting method. This technique is potentially valuable when there are hundreds of values displayed continuously.

- *Use color in graphic displays for greater information density*   In graphs with multiple plots, color can be helpful in showing which line seg-ments form the full graph. The usual strategies for differentiating lines in black-on-white graphs—such as dotted lines, thicker lines, and dashed lines—are not as effective as is using separate colors for each line. Architectural plans benefit from color coding of electrical, tele-phone, hot-water, cold-water, and natural-gas lines. Similarly, maps can have greater information density when color coding is used.

The complexity of using color was demonstrated in studies of decision-making tasks, rather than of simple location of information or recall, with management-information systems (Benbasat et al., 1986). Although color coding was found to be beneficial and preferred, there was an interaction with personality factors. Further intricate relationships were found in a comparison of monochrome versus color-coded pie charts, bar charts, line graphs, and data tables, in which color coding sped performance in all but the line graphs (Hoadley, 1990). Hoadley concludes that "uncritical addition of color may not be uniformly beneficial. Color is a subtle variable that can significantly enhance the decision maker's ability to extract information."

Box 11.4

Guidelines that highlight the complex potential benefits and dangers of using color coding.

Guidelines for using color
- Use color conservatively: Limit the number and amount of colors.
- Recognize the power of color to speed or slow tasks.
- Ensure that color coding should supports the task.
- Make color coding appear with minimal user effort.
- Keep color coding under user control.
- Design for monochrome first.
- Use color to help in formatting.
- Be consistent in color coding.
- Be alert to common expectations about color codes.
- Use color changes to indicate status changes.
- Use color in graphic displays for greater information density.

Benefits of using color
- Various colors are soothing or striking to the eye.
- Color can improve an uninteresting display.
- Color facilitates subtle discriminations in complex displays.
- A color code can emphasize the logical organization of information.
- Certain colors can draw attention to warnings.
- Color coding can evoke more emotional reactions of joy, excitement, fear, or anger.

Dangers of using color
- Color pairings may cause problems.
- Color fidelity may degrade on other hardware.
- Printing or conversion to other media may be a problem.

Color displays are becoming nearly universal, even in laptops, and designers usually make heavy use color in system designs. There are undoubtedly benefits of increased user satisfaction and often increased performance; however, there are real dangers in misusing color. Care should be taken to make appropriate designs and to conduct thorough evaluations (Box 11.4).

11.6 Practitioner's Summary

The wording of system messages may have an effect on performance and attitudes, especially for novices whose anxiety and lack of knowledge put them at a disadvantage. Designers might make improvements by merely using more specific diagnostic messages, offering constructive guidance rather than focusing on failures, employing user-centered phrasing, choosing a suitable physical format, and avoiding vague terminology or numeric codes.

When giving instructions, focus on the user and the user's tasks. Avoid anthropomorphic phrasing and use the *you* form to guide the novice user. Avoid judging the user. Simple statements of status are more succinct and usually are more effective.

Pay careful attention to display design, and develop a local set of guidelines for all designers. Use spacing, indentation, columnar formats, and field labels to organize the display for users. Denser displays, but fewer of them, may be advantageous. Color can improve some displays and can lead to more rapid task performance with higher satisfaction; but improper use of color can mislead and slow users.

Organizations can benefit from careful study of display-design guidelines documents and from the creation of their own set of guidelines tailored to local needs (see Section 3.2.1). This document should also include a list of local terminology and abbreviations. Consistency and thorough testing are critical.

11.7 Researcher's Agenda

Experimental testing could refine the proposed error-message guidelines proposed here, and could identify the sources of user anxiety or confusion. Message placement, highlighting techniques, and multiple-level message strategies are candidates for exploration. Improved analysis of sequences of user actions to provide more effective messages automatically would be useful.

There is a great need for testing to validate data-display and color-design guidelines. Basic understanding and cognitive models of visual perception

of displays would be a dramatic contribution. Do users follow a scanning pattern from the top left? Do users whose natural language reads from right to left or users from different cultures scan displays differently? Does use of whitespace around or boxing of items facilitate comprehension and speed interpretation? When is a single dense display preferable to two sparse displays? How does color coding reorganize the pattern of scanning?

World Wide Web Resources   WWW

Usage guidelines for color are nicely done on the World Wide Web with some empirical results, but the most informative and enjoyable experience is simply browsing through the lively and colorful websites. Styles and fashions come and go quickly, so save the examples you like best.

http://www.aw.com/DTUI

References

Benbasat, I., Dexter, A. S., and Todd, P., The influence of color and graphical information presentation in a managerial decision simulation, *Human–Computer Interaction*, 2 (1986), 65–92.

Brennan, Susan E. and Ohaeri, Justina O., Effects of messages style on users' attributions towards agents, *Proc. ACM CHI '94 Human Factors in Computing Systems: Conference Companion*, ACM, New York (1994), 281–282.

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

Burns, Michael J., Warren, Dianne L., and Rudisill, Marianne, Formatting space-related displays to optimize expert and nonexpert user performance, *Proc. ACM SIGCHI '86 Human Factors in Computing Systems*, ACM, New York (1986), 274–280.

Casner, Stephen M., A task-analytic approach to the automated design of information graphic presentations, *ACM Transactions on Graphics*, 10, 2 (April 1991), 111–151.

Dean, M., How a computer should talk to people, *IBM Systems Journal*, 21, 4 (1982), 424–453.

Donner, Kimberly A., McKay, Tim, O'Brien, Kevin M., and Rudisill, Marianne, Display format and highlighting validity effects on search performance using complex visual displays, *Proc. Human Factors Society—Thirty-Fifth Annual Meeting*, Santa Monica, CA (1991), 374–378.

Foley, James D., van Dam, Andries, Feiner, Steven K., and Hughes, John F., *Computer Graphics: Principles and Practice* (Second Edition), Addison-Wesley, Reading, MA (1990).

Friedman, Batya, "It's the computer's fault"—Reasoning about computers as moral agents, *Proc. ACM CHI '95 Human Factors in Computing Systems: Conference Companion*, ACM, New York (1995), 226–227.

Galitz, Wilbert O., *It's Time to Clean Your Windows: Designing GUIs that Work*, John Wiley and Sons, New York (1994).

Gardiner, Jeremy, *Digital Photo Illustration*, Van Nostrand Reinhold, New York (1994).

Hoadley, Ellen D., Investigating the effects of color, *Communications of the ACM*, 33, 2 (February 1990), 120–139.

Isa, Barbara S., Boyle, James M., Neal, Alan S., and Simons, Roger M., A methodology for objectively evaluating error messages, *Proc. ACM CHI '83 Human Factors in Computing Systems*, ACM, New York (1983), 68–71.

Kiesler, Sara, Sproull, Lee, and Waters, Keith, A prisoner's dilemma experiment on cooperation with people and human-like computers, *Journal of Personality and Social Psychology*, 70, 1 (1996), 47–65.

Laurel, Brenda, Interface agents: Metaphors with character. In Laurel, Brenda (Editor), *The Art of Human–Computer Interface Design*, Addison-Wesley, Reading, MA (1990), 355–365.

Lohse, Jerry. A cognitive model for perception and understanding of graphs, *Proc. ACM CHI '91 Human Factors in Computing Systems*, ACM, New York (1991), 137–144.

Mackinlay, Jock, Automating the design of graphical presentations of relational information, *ACM Transactions on Graphics*, 5, 2 (1986), 110–141.

Maes, Pattie, Artificial life meets entertainment: Lifelike autonomous agents, *Communications of the ACM*, 38, 11 (November 1995), 108–114.

Marcus, Aaron, *Graphic Design for Electronic Documents and User Interfaces*, ACM Press, New York (1992).

Martin, Dianne, ENIAC: Press conference that shook the world, *IEEE Technology and Society Magazine*, 14, 4 (Winter 1995/96), 3–10.

Mosteller, W., Job entry control language errors, *Proceedings of SHARE 57*, SHARE, Chicago (1981), 149–155.

Mullet, Kevin and Sano, Darrell, *Designing Visual Interfaces: Communication Oriented Techniques*, Sunsoft Press, Englewood Cliffs, NJ (1995).

Mumford, Lewis, *Technics and Civilization*, Harcourt Brace and World, New York (1934), 31–36

Nass, Clifford, Steuer, Jonathan, and Tauber, Ellen R., Computers are social actors, *Proc. ACM CHI '94 Human Factors in Computing Systems*, ACM, New York (1994), 72–78

Nass, Clifford, Lombard, Matthew, Henriksen, Lisa, and Steuer, Jonathan, Anthropocentrism and computers, *Behaviour & Information Technology*, 14, 4 (1995), 229–238.

Pace, Bruce J., Color combinations and contrast reversals on visual display units, *Proceedings of the Human Factors Society Twenty-Eighth Annual Meeting*, Santa Monica, CA (1984), 326–330.

Perlman, Gary and Swan, II, J. Edward, Color versus texture coding to improve visual search performance, *Proc. Human Factors Society—Thirty-Seventh Annual Meeting*, Santa Monica, CA (1993), 343–347.

Quintanar, Leo R., Crowell, Charles R., and Pryor, John B., Human–computer interaction: A preliminary social psychological analysis, *Behavior Research Methods and Instrumentation*, 14, 2 (1982), 210–220.

Resnik, P. V. and Lammers, H. B., The influence of self-esteem on cognitive response to machine-like versus human-like computer feedback, *Journal of Social Psychology*, 125, (1986), 761–769.

Roth, Steven F., Kolojejchick, John, Mattis, Joe, and Goldstein, Jade, Interactive graphic design using automatic presentation knowledge, *Proc. ACM CHI '94 Human Factors in Computing Systems*, ACM, New York (1994), 112–117.

Salomon, Gitta, New Uses for Color, In Laurel, Brenda (Editor), *The Art of Human–Computer Interface Design*, Addison-Wesley, Reading, MA (1990), 269–278.

Sears, Andrew, Layout appropriateness: Guiding user interface design with simple task descriptions, *IEEE Transactions on Software Engineering*, 19, 7 (1993), 707–719.

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

Shneiderman, Ben, Looking for the bright side of agents, *ACM Interactions*, 2, 1 (January 1995), 13–15.

Shubin, Hal, Falck, Deborah, and Johansen, Ati Gropius, Exploring color in interface design, *ACM interactions III.4* (August 1996), 36–48.

Smith, Sid L. and Mosier, Jane N., *Guidelines for Designing User Interface Software*, Report ESD-TR-86-278, MITRE, Bedford, MA (August 1986).

Snyder, Harry L., Decker, Jennie J., Lloyd, Charles J. C., and Dye, Craig, Effect of image polarity on VDT task performance, *Proc. Human Factors Society—Thirty-Fourth Annual Meeting*, Santa Monica, CA (1990), 1447–1451.

Springer, Carla J., Retrieval of information from complex alphanumeric displays: Screen formatting variables' effect on target identification time. In Salvendy, Gavriel (Editor), *Cognitive Engineering in the Design of Human–Computer Interaction and Expert Systems*, Elsevier, Amsterdam, The Netherlands (1987), 375–382.

Staggers, Nancy, Impact of screen density on clinical nurses' computer task performance and subjective screen satisfaction, *International Journal of Man–Machine Studies*, 39, 5 (November 1993), 775–792.

Takeuchi, Akikazu and Naito, Taketo, Situated facial displays: Towards social interaction, *Proc. ACM CHI '95 Human Factors in Computing Systems*, ACM, New York (1995), 450–455.

Teitelbaum, Richard C., and Granda, Richard F., The effects of positional constancy on searching menus for information, *Proc. ACM CHI '83 Human Factors in Computing Systems*, ACM, New York (1983), 150–153.

Thorell, L. G., and Smith, W. J., *Using Computer Color Effectively*, Prentice-Hall, Englewood Cliffs, NJ (1990).

Travis, David S., *Effective Color Displays: Theory and Practice*, Academic Press, New York (1991).

Treisman, Anne, Perceptual grouping and attention in visual search for features and for objects, *Journal of Experimental Psychology: Human Perception and Performance*, 8, 2 (1982), 194–214.

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

Tullis, T. S., An evaluation of alphanumeric, graphic and color information displays, *Human Factors*, 23, (1981), 541–550.

Tullis, T. S., Screen design. In Helander, Martin (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988a), 377–411.

Tullis, T. S., A system for evaluating screen formats: Research and application. In Hartson, H. Rex, and Hix, Hartson (Editors), *Advances in Human–Computer Interaction* (Volume 2), Ablex, Norwood, NJ (1988b), 214–286.

Vincow, Michelle A. and Wickens, Christopher, Spatial layout of displayed information: Three steps toward developing quantitative models, *Proc. Human Factors Society—Thirty-Seventh Annual Meeting*, Santa Monica, CA (1993), 348–352.

Walker, Janet H., Sproull, Lee, and Subramani, R., Using a human face in an interface, *Proc. ACM CHI '94 Human Factors in Computing Systems*, ACM, New York (1994), 85–91.

Weitzman, Donald O., Color coding re-viewed, *Proc. Human Factors Society—Twenty-ninth Annual Meeting*, Santa Monica, CA (1985), 1079–1083.
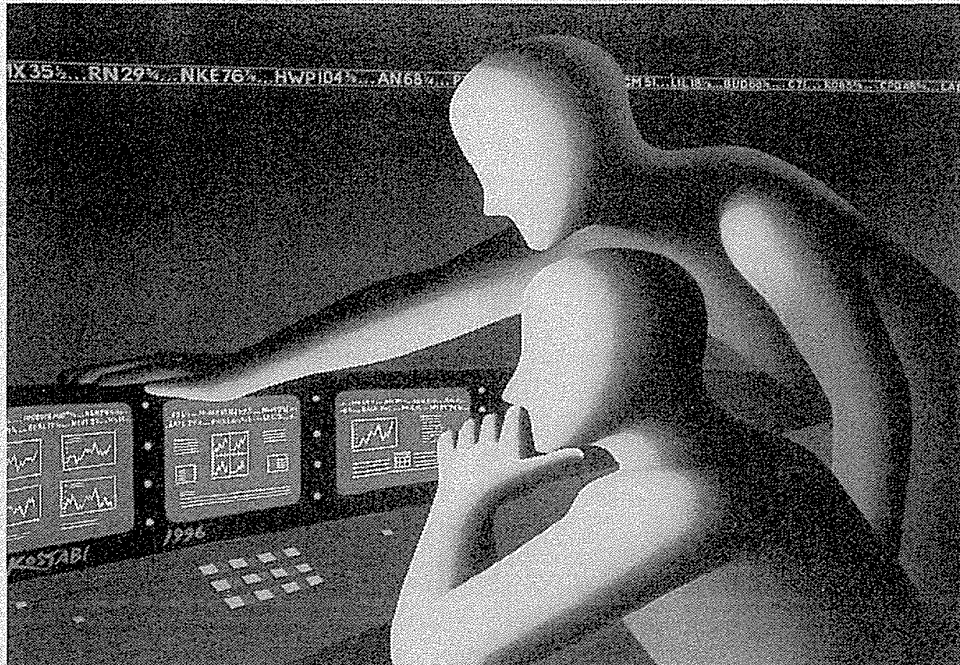
Mark Kostabi, *Computer Virus*, 1992

Printed Manuals, Online Help, and Tutorials

What is really important in education is . . . that the mind is matured, that energy is aroused.

Soren Kierkegaard, *Either/Or, Volume II*

Chapter 12

12.1 Introduction
12.2 Reading from Paper versus from Displays
12.3 Preparation of Printed Manuals
12.4 Preparation of Online Facilities
12.5 Practitioner's Summary
12.6 Researcher's Agenda
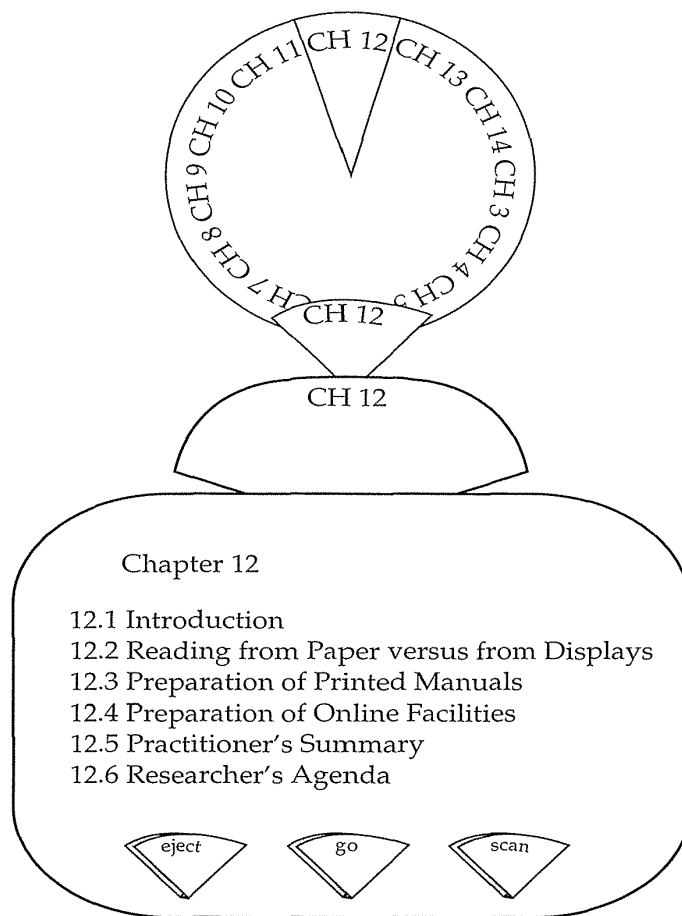
12.1 Introduction

All users of interactive computer systems require training. Many users learn from another person who knows the system, but training materials are often necessary. Traditional printed manuals are sometimes poorly written, but this medium can be effective and convenient if prepared properly (Price, 1984; Brockmann, 1990). Online help, manuals, and tutorials that use the same interactive system to provide training, reference, and reminders about specific features and syntax have become expected components of most systems. In fact, as display devices appear in cars, cameras, VCRs, and elsewhere, ubiquitous help should be the norm.

Learning anything new is a challenge. Although challenge can be joyous and satisfying, when it comes to learning about computer systems, many people experience anxiety, frustration, and disappointment. Much of the dif-

ficulty flows directly from the poor design of the menus, displays, or instructions that lead to error conditions, or simply from the inability of users to know what to do next.

Even though increasing attention is being paid to improving user-interface design, the complexity of online systems grows. There will always be a need for supplemental materials that aid users, in both paper and online form. Some of the many forms of paper user manuals are:

- *Brief getting-started notes* to enable eager first-time users to try out features
- *Introductory tutorial* to explain common features
- *Thorough tutorial* that covers typical and advanced tasks
- *Quick reference card* with a concise presentation of the syntax
- *Conversion manual* that teaches the features of the current system to users who are knowledgeable about some other system
- *Detailed reference manual* with all features covered

There is also a variety of online materials:

- *Online user manual*   This simple conversion of the traditional user manual to electronic form may make the text more readily available, but more difficult to read and absorb
- *Online help facility*   The most common form of online help is a list of article titles (possibly searchable for keywords) and an index of terms that lead to articles.
- *Online tutorial*   This potentially appealing and innovative approach uses the electronic medium to teach the novice user by showing simulations of the working system, by displaying attractive animations, and by engaging the user in interactive sessions.
- *Online demonstration*   Potential users who want an overview of the software can benefit from an online demonstration that gives them a guided tour through the software.

Duffy and colleagues (1992) classify paper and online materials by user's goals:

User's Goal	Medium of Delivery	
	Paper	*Online*
I want to *buy* it	sales brochure fact sheet	demonstration program
I want to *learn* it	tutorial manual	guided tour
I want to *use* it	user's manual	online help online document

Other forms of instruction or information acquisition include classroom instruction, personal training and assistance, telephone consultation, videotapes, and audio tapes. These forms are not discussed here, but many of the same instructional design principles apply.

12.2 Reading from Paper versus from Displays

The technology of printing text on paper has been evolving for more than 500 years. The paper surface and color, the typeface, character width, letter sharpness, text contrast with the paper, width of the text column, size of margins, spacing between lines, and even room lighting all have been explored in efforts to produce the most appealing and readable format.

In the last 40 years, the cathode ray tube *(CRT)*, often called the *visual display unit (VDU) or tube (VDT),* has emerged as an alternate medium for presenting text, but researchers have only begun the long process of optimization (Cakir et al., 1980; Grandjean and Vigliani, 1982; Heines, 1984; Helander, 1987; Hansen and Haas, 1988; Oborne and Holton, 1988; Creed and Newstead, 1988; Horton, 1990) to meet user needs. Serious concerns about CRT radiation or other health hazards have lessened as manufacturers, labor unions, and government agencies have funded major research in this area. One advantage of the increasingly popular liquid crystal displays (LCDs) is the lessened concerns about radiation.

The widespread reports about visual fatigue and stress have been confirmed, but these conditions respond well to rest, frequent breaks, and task diversity. But even before users are aware of visual fatigue or stress, their capacity to work with displays may be below their capacity to work with printed materials.

Approximately 10 studies during the 1980s found 15- to 30-percent slower task times for comprehension or proofreading of text on computer displays, compared to on paper. The potential disadvantages of reading from displays include these:

- *Fonts* may be poor, especially on low resolution displays. The dots composing the letters may be so large that each is visible, making the user expend effort to recognize the character. Monospace (fixed width) fonts, lack of appropriate kerning (for example, adjustments to bring "V" and "A" closer together), inappropriate interletter and interline spacing, and inappropriate colors may all complicate recognition.

- *Low contrast* between the characters and the background, and *fuzzy character* boundaries also can cause trouble.

- *Emitted light* from displays may be more difficult to read by than reflected light from paper; glare may be greater, *flicker* can be a problem, and the *curved display surface* may be troubling.

- *Small displays* require frequent *page turning*; issuing the page-turning commands is disruptive, and the page turns are unsettling, especially if they are slow and visually distracting.

- *Reading distance* can be greater than for paper, displays are *fixed* in place, and display *placement* may be too high for comfortable reading (optometrists suggest reading be done with the eyes in a downward-looking direction); the "near quintad" are the five ways eyes adjust to seeing close items (Grant, 1990): *accommodation* (lens-shape change), *convergence* (looking toward the center), *meiosis* (pupillary contraction), *excyclotorsion* (rotation), and *depression of gaze* (looking down).

- *Layout and formatting* can be problems, such as improper margins, inappropriate line width (35 to 55 characters is recommended), or awkward justification (left justification and ragged right are recommended).

- *Reduced hand and body motion* with displays as compared to paper, and *reduce rigid posture* for displays, as both may be fatiguing.

- *Unfamiliarity of displays* and the *anxiety* that the image may disappear can increase *stress*.

The fascinating history of this issue goes back at least to Hansen and associates (1978) who found that seven students who were asked to take examinations on paper and on PLATO terminals took almost twice as long online. Much of the increased time could be attributed to system delays, poor software design, and slower output rates, but the authors could not thus account for 37 percent of the longer time on PLATO. They conjecture that this additional time could be attributed to uncertainty about how to control the medium, what the system would do, and what the system had done.

Wright and Lickorish (1983) studied proofreading of 134-line texts that contained 39 errors (typographical errors, spelling errors, missing words, and repeated words). Thirty-two subjects read from an Apple II using an 80-column display on a 12-inch black-and-white display screen or from hardcopy generated by a dot-matrix printer. There was a modest, but significant, increase in detected errors with the printed text. There was also a 30- to 40-percent advantage in speed with the printed text.

Gould and Grischkowsky (1984) studied proofreading for typographic errors on displays and on output from a computer-controlled photocomposer. Both the displays and the hardcopy texts had 23 lines per page, with about nine words per line. Twenty-four subjects spent 8 hours reading in each format. The reading rate was significantly faster on hardcopy (200 words per minute) than on the screens (155 words per minute). Accuracy was slightly, but reliably, higher on hardcopy. The subjective ratings of readability were similar for both forms. A later series of studies with improved displays led to much smaller differences and even to the elimination of differences (Gould et al., 1987a; 1987b).

More recent results demonstrate no difference between reading text on displays versus paper when researchers control for enough of the variables. Oborne and Holton (1988) believe that earlier studies may have been flawed by lack of control and comparing low-resolution displays to high-quality print. In their comprehension studies using a within-subjects design with approximately 380-word passages, there were no statistically significant differences between displays and photographs of displays. They controlled for position, distance to retina, line length, layout, and illumination. Jorna's study (1991) solidly demonstrates that, when the resolution of the display matches that of the hardcopy, there is no difference in reading speed or perceived image quality. Since computer displays do not yet have the resolution of paper, it is still easier to read from paper.

These empirical studies isolated the issues and led to a clear message for designers: High-resolution displays are recommended if users are to read lengthy texts online. Related studies clarify that short response times, fast display rates, black text on white background, and page-sized displays are important considerations if displayed text is meant to replace paper documents.

12.3   Preparation of Printed Manuals

Traditionally, training and reference materials for computer systems were printed manuals. Writing these manuals was often left to the most junior member of the development team as a 5-percent effort at the end of the project. As a result, the manuals were often poorly written, were not suited to the background of the users, were delayed or incomplete, and were tested inadequately.

There is a growing awareness that users are not like designers, that system developers might not be good writers, that it takes time and skill to write an effective manual, that testing and revisions must be done before widespread dissemination, and that system success is closely coupled to documentation quality.

In an early experiment, Foss, Rosson, and Smith (1982) modified a standard text-editor manual. The standard manual presented all the details about a command; the modified manual offered a progressive, or *spiral*, approach to the material by presenting subsets of the concepts. The standard manual used an abstract formal notation to describe the syntax of the commands; the modified manual showed numerous examples. Finally, the standard manual used terse technical prose; the modified manual included readable explanations with fewer technical terms.

During the experiment, subjects took 15 to 30 minutes to study the manuals, and were then asked to complete nine complex text-editing or creation tasks within a 3-hour period. On all five dependent measures, the subjects with the modified manual demonstrated superior performance (Table 12.1).

Table 12.1

Results of a study comparing standard manual with a modified manual (spiral approach, numerous examples, more readable explanations) (Foss et al., 1982).

	Standard Manual	Modified Manual
Tasks Completed	7.4	8.8
Average Minutes per Task	26.6	16.0
Average Edit Errors per Task	1.4	.3
Average Commands per Task	23.6	13.0
Average Requests for Verbal Help	5.5	2.6

The results make a strong case for the effect of the manual on the success of the user with the system.

The iterative process of refining a text-editor manual and evaluating its effectiveness is described by Sullivan and Chapanis (1983). They rewrote a manual for a widely used text editor, conducted a walk-through test with colleagues, and performed a more elaborate test with five temporary secretaries. Subjective and objective metrics showed substantial benefits from the rewriting. Allwood and Kalen (1997) realized similar improvements by keeping sentences short, by avoiding jargon, by using a new paragraph for each command, and by emphasizing tasks.

The benefits of well-designed manuals include shorter learning times, better user performance, increased user satisfaction and fewer calls for support (Spencer and Yates, 1995).

12.3.1 Use of the OAI Model to design manuals

The objects–actions interface (OAI) model offers insight to the learning process, and thus provides guidance to instructional-materials designers. If the user has only partial knowledge of the task objects and actions (Fig. 12.1), then training in the task is the first step. For a task such as letter writing, users must learn about address blocks, salutations, content, and signatures. Once users know the hierarchy of objects from the high-level down to the atomic and recognize the range of their high-level intentions down to their specific action steps (Fig. 12.2), they are equipped to learn about the interface representations. The instructional materials should start from familiar objects and actions in the letter-writing task, link these concepts to the high-level interface objects and actions (Fig. 12.3), and then show the syntax needed to accomplish each task. Knowledgeable users who understand the task and interface (see Fig. 2.2) can move on to expert levels of usage with shortcuts that speed performance.
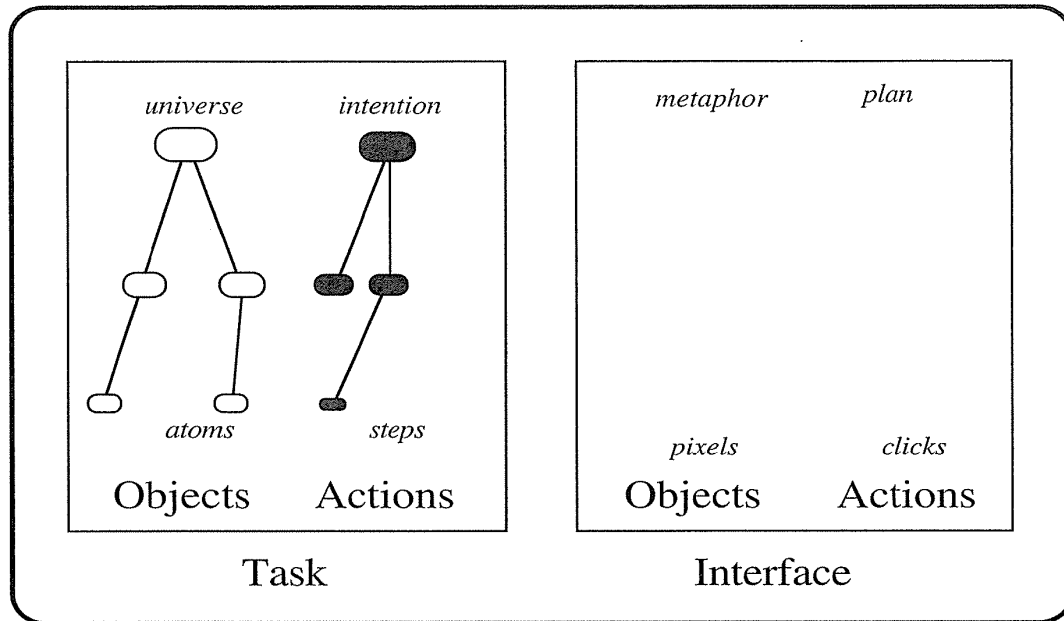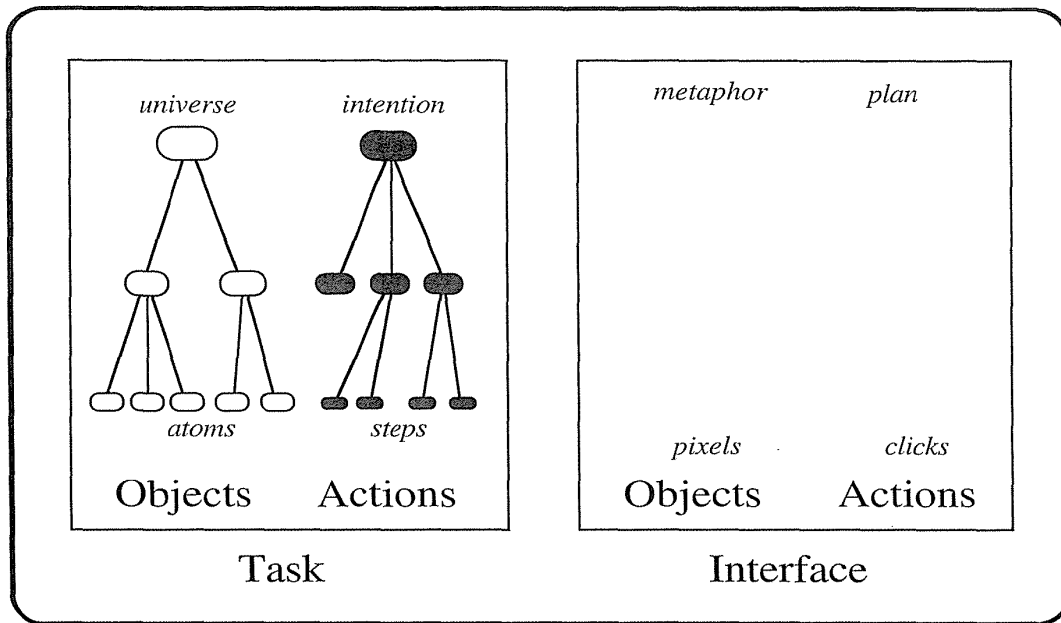
Figure 12.1

A representation of users who know some of task objects and actions, but know nothing about the interface. A deeper knowledge of task objects and actions will give them a framework for learning about the interface.

Some users are knowledgeable about letter writing and word processing, but must learn a new word processor. They need a presentation that shows the relationship between the metaphors and plans they know and the new ones. Increasingly, the metaphors and plans are shared among word processors, but the dialog boxes, clicks, and keystrokes may vary.

Some users have learned the task and interface objects and actions, but cannot recall details of how to convert their plans into detailed actions.

These three scenarios demonstrate three popular forms of printed materials: the *introductory tutorial,* the *conversion manual,* and the *quick reference* (cheat sheet).

The OAI model can also help researchers to map the current levels of knowledge in learning systems. For example, a user who is learning about database-management systems for Congressional voting patterns might have some knowledge about the database and its manipulation, the query-language concepts, and the syntax needed. This user would benefit from seeing typical queries that would demonstrate the syntax and serve as templates for other queries. In fact, complete *sample sessions* are extremely helpful in giving a portrait of the system features and interaction style (Fig. 12.4). Many users will work through these sessions to verify their under-

Figure 12.2

A representation of users who know the task adequately, but do not know the interface. Educational materials for this community should explain the interface objects and actions, starting with plans.

standing, to gain a sense of competence in using the system, and to see whether the system and the manual match.

Another helpful guide to using a system is an overall *flow diagram* of activity (Fig. 12.5). Such visual overviews provide a map that orients users to the transitions from one activity to another. Similarly, if the system uses a complex model of data objects, an overview diagram may help users to appreciate the details.

12.3.2 Organization and writing style

Designing instructional materials is a challenging endeavor. The author must be knowledgeable about the technical content; sensitive to the background, reading level, and intellectual ability of the reader; and skilled in writing lucid prose. Assuming that the author has acquired the technical content, the primary job in creating a manual is to understand the readers and the tasks that they must perform.

A precise statement of the *educational objectives* (Mager, 1962) is an invaluable guide to the author and the reader. The sequencing of the instructional
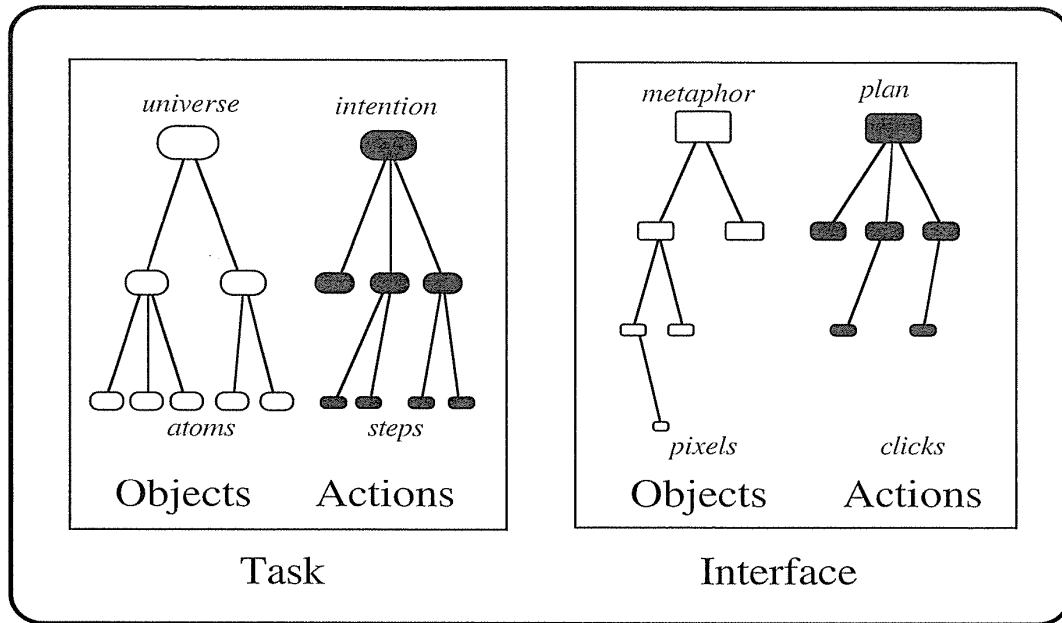
Figure 12.3

A representation of users who are knowledgeable about the task and high-level
interface aspects, and need to learn only the specific visual representations and syn-
tactic details. For example, someone who knows about writing scientific articles and
is familiar with at least one word processor will find it relatively easy to acquire the
low-level objects and actions in another word processor.

contents is governed by the reader's current knowledge and ultimate objec-
tives. Precise rules are hard to identify, but the author should attempt to pre-
sent concepts in a logical sequence with increasing order of difficulty, to
ensure that each concept is used in subsequent sections, to avoid forward ref-
erences, and to construct sections that contain approximately equal amounts
of new material. In addition to these structural requirements, the manual
should have sufficient examples and complete sample sessions.

Within a section that presents a concept, the author should begin with
the reason for the concept, describe the concept in task-domain semantic
terms, then show the computer-related semantic concepts, and, finally, offer
the syntax.

The choice of words and their phrasing is as important as the overall
structure. A poorly written sentence mars a well-designed manual, just as an
incorrect note mars a beautifully composed sonata. The classic book on writ-
ing, *The Elements of Style* (Strunk and White, 1979) is a valuable resource.
Style guides for organizations represent worthy attempts at ensuring consis-

SEARCH: I am looking for an image of Goleta Beach.

Steps:

1. MAP BROWSER: The user selects zoom 10X IN for the Zoom Factor, and clicks with the mouse on point near Goleta in Southern California.

2. MAP BROWSER: The Zoom Factor is changed to 5X IN. The Map Database is changed from World to USA, so that the next map redraw will display additional features; then, the user again clicks on a point near Goleta.

3. MAP BROWSER: The user now selects GAZETTEER to aid in the location of Goleta Beach through search by feature name.

4. GAZETTEER SEARCH FORM: Limit search to items CONTAINED within for the spatial search method and Current Map Region are preselected by system. The user inputs `Goleta` into the Sort Name field and selects Show Terms.

5. GAZETTEER SUGGESTIONS: A page is displayed to inform the user that the domain of this field is too large to display. The user is then given the option to use a fuzzy text search engine to help her to identify valid terms. Conquest search is selected, and the user is taken to a page to suggest values for that field (Sort Name).

6. GAZETTEER SUGGESTIONS: If a term was entered on the Gazetteer Search Form page, this term is seeded into the selection box and the user selects Suggest. Otherwise, the user provides a term to be used to suggest values for Sort Name.

7. GAZETTEER SUGGESTIONS: The terms GOLETA BEACH, GOLETA POINT, GOLETA PIER, GOLETA COVE, AND GOLETA are selected, and ADD is selected to add these items to the search selection list for the Gazetteer.

Figure 12.4

A sample walkthrough is often a convenient way to explain system usage. Many users try the sample and then make variations to suit their needs. This sample shows the first seven of 21 steps to retrieve a map image for the Alexandria Digital Library. (University of California at Santa Barbara, CA.)

tency and high quality. Of course, no set of guidelines can turn a mediocre writer into a great writer. Writing is a highly creative act; effective writers are national treasures.

Writing style should match the users' reading ability (Roemer and Chapanis, 1982). After a tutorial was written at the fifth-, tenth-, and fifteenth-grade levels, 54 subjects were divided into groups with low, middle, and high reading ability. Higher reading ability led to significant reductions in the completion time and number of errors, and to higher scores on a concepts test. Increased complexity of the writing style did not lead to significant differences on the performance variables, but subjective preferences significantly
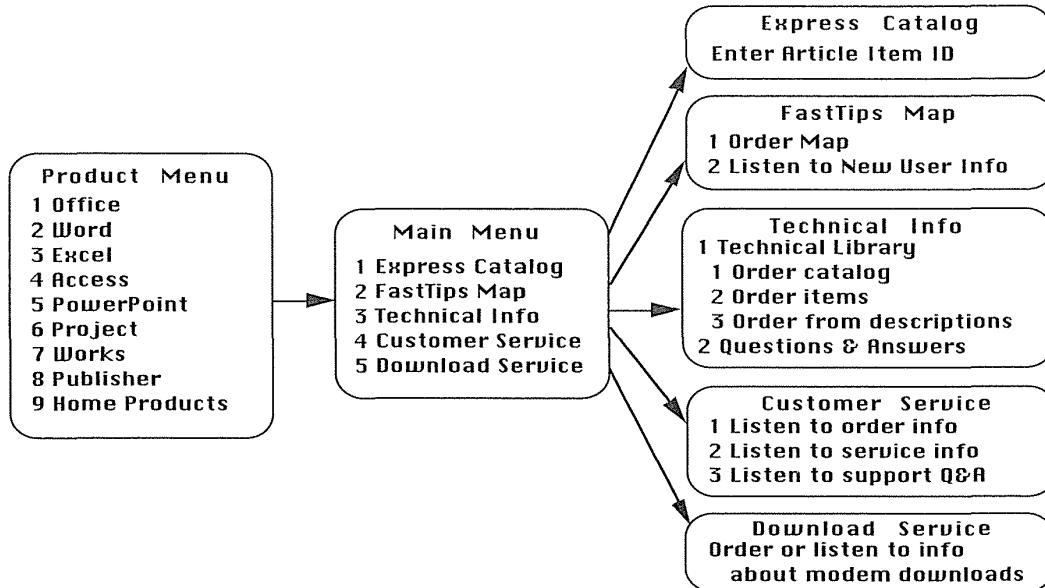
Figure 12.5

A transition diagram can be a helpful aid for users. This FastTips map, adapted from the Microsoft Support Network, gives telephone callers an overview and therefore capability to move rapidly within the support service. Users identify relevant articles, which are faxed to them. (Used with permission of Microsoft Corp., Redmond, WA.)

favored the fifth-grade version. Subjects could overcome the complex writing style, but the authors conclude that "the most sensible approach in designing computer dialogue is to use the simplest language."

Thinking-aloud studies (see Section 4.3) of subjects who were learning word processors revealed the enormous difficulties that most novices have and the strategies that they adopt to overcome those difficulties (Carroll and Mack, 1984). Learners are actively engaged in trying to make the system work, to read portions of the manual, to understand the screen displays, to explore the function of keys, and to overcome the many problems that they encounter. Learners apparently prefer trying out actions on the computer, rather than reading lengthy manuals. They want to perform meaningful, familiar tasks immediately, and to see the results for themselves. They apply real-world knowledge, experience with other computer systems, and frequent guesswork, unlike the fanciful image of the new user patiently reading through and absorbing the contents of a manual.

These observations led to the design of *minimal manuals* that drastically cut verbiage, encourage active involvement with hands-on experiences as soon as

possible, and promote *guided exploration* of system features (Carroll, 1984; 1990). The key principles have been refined over time (van der Meij and Carroll, 1995):

- Choose an action-oriented approach
- Anchor the tool in the task domain
- Support error recognition and recovery
- Support reading to do, study, and locate

Results of field trials and of dozens of empirical studies demonstrate that with improved manuals, the learning time can be reduced substantially and user satisfaction increased (van der Meij and Lazonder, 1993).

Visual aspects are helpful to readers, especially with highly visual direct-manipulation and GUIs. Showing numerous well-chosen screen prints that demonstrate typical uses enables users to develop an understanding and a *predictive model* of the system. Often, users will mimic the examples in the manual during their first trials of the software. Figures containing complex data structures, transition diagrams, and menu maps (Parton et al., 1985) can improve performance dramatically by giving users access to fundamental structures created by designers.

Of course, every good manual should have a *table of contents* and an *index*. *Glossaries* can be helpful for clarifying technical terms. *Appendices* with error messages are recommended.

Whether to give *credit* to authors and designers is a lively and frequently debated issue. Advocates, including me, encourage giving credit in the manuals to honor good work, to encourage contributor responsibility for doing an excellent job, and to build the users' trust. Responsibility and trust are increased because the contributors were willing to have their names listed publicly. Having names in the manual makes software fit in with other creative human endeavors, such as books, films, and music, in which contributors are acknowledged, even if there are dozens of them. Opponents say that it is difficult to identify each contribution or that unwelcome telephone calls might be received by contributors.

12.3.3 Nonanthropomorphic descriptions

The metaphors used in describing computer systems can influence the user's reactions. Some writers are attracted to an anthropomorphic style that suggests that the computer is close to human in its powers. This suggestion can anger some users; more likely, it is seen as cute the first time, silly the second, and annoyingly distracting the third time (see Section 11.3).

Many designers prefer to focus attention on the users and on the tasks that the users must accomplish. In introductory sections of user manuals and online help, use of the second-person singular pronoun ("you") seems

appropriate. Then, in later sections, simple descriptive sentences place the emphasis on the user's tasks.

In a transportation-network system, the user might have to establish the input conditions on the screen and then to invoke the program to perform an analysis:

> **Poor:** The expert system will discover the solution when the F1 key is pressed.
> **Better:** You can get the solution by pressing F1.
> **Better:** To solve, press F1.

The first description emphasizes the computer's role, the second focuses on the user and might be used in an introduction to the system. In later sections, the briefer third version is less distracting from the task.

In discussing computers, writers might be well advised to avoid such verbs as these:

> **Poor:** know, think, understand, have memory

In their place, use more mechanical terms, such as

> **Better:** process, print, compute, sort, store, search, retrieve

When describing what a user does with a computer, avoid such verbs as

> **Poor:** ask, tell, speak to, communicate with

In their place, use such terms as

> **Better:** use, direct, operate, program, control

Still better is to eliminate the reference to the computer, and to concentrate instead on what the user is doing, such as writing, solving a problem, finding an answer, learning a concept, or adding up a list of numbers:

> **Poor:** The computer can teach you Spanish words.
> **Better:** You can use the computer to learn Spanish words.

Make the user the subject of the sentence:

> **Poor:** The computer will give you a printed list of employees.
> **Poor:** Ask the computer to print a list of employees.
> **Better:** You can get the computer to print a list of employees.
> **Better:** You can print a list of employees.

The final sentence puts the emphasis on the user and eliminates the computer.

Poor: The computer needs to have the disk in the disk drive to boot the system.

Better: Put the disk labeled A2 in the disk drive before starting the computer.

Better: To begin writing, put the Word Processor disk in the drive.

The final form emphasizes the function or activity that the user is about to perform.

Poor: The computer knows how to do arithmetic.

Better: You can use the computer to do arithmetic.

Focus on the user's initiative, process, goals, and accomplishments.

12.3.4   Development process

Recognizing the difference between a good and a bad manual is necessary to producing a successful manual on time and within a reasonable budget. *Production of a manual,* like any project, must be managed properly, staffed with suitable personnel, and monitored with appropriate milestones (Box 12.1).

Getting started early is invaluable. If the manual-writing process begins before the implementation, then there is adequate time for review, testing, and refinement. Furthermore, the user's manual can act as a more complete and comprehensible alternative to the formal specification for the software. Implementers may miss or misunderstand some of the design requirements when reading a formal specification; a well-written user manual may clarify the design. The manual writer becomes an effective critic, reviewer, or question asker who can stimulate the implementation team. Early development of the manual enables pilot testing of the software's learnability even before the system is built. In the months before the software is completed, the manual may be the best way to convey the designers' intentions to potential customers and users, as well as to system implementers and project managers.

Ample lead time in the development of the manual allows for reviews and suggestions by designers, other technical writers, potential customers, intended users, copy editors, graphic artists, lawyers, marketing personnel, instructors, telephone consultants, and product testers (Brockmann, 1990).

Beyond informal reviews by people with different backgrounds, there are other strategies for evaluating the manual. Checklists of features have been developed by many organizations based on experience with previous manuals. Automated metrics of reading level or difficulty are available to help

Box 12.1

User-manual guidelines based on practice and empirical studies.

User Manual Guidelines
- Product
  - Let user's tasks guide organization (outside-in).
  - Let user's learning process shape sequencing.
  - Present task concepts before interface objects and actions.
  - Keep writing style clean and simple.
  - Show numerous examples.
  - Offer meaningful and complete sample sessions.
  - Draw transition or menu-tree diagrams.
  - Try advance organizers and summaries.
  - Provide table of contents, index, and glossary.
  - Include list of error messages.
  - Give credits to all project participants.
- Process
  - Seek professional writers and copy writers.
  - Prepare user manuals early (before implementation).
  - Review drafts thoroughly.
  - Field test early editions.
  - Provide a feedback mechanism for readers.
  - Revise to reflect changes regularly.

designers isolate complex sections of text. Computer-based style evaluations and spelling checkers are useful tools in refining any document.

Informal walkthroughs with users are usually an enlightening experience for software designers and manual writers. Potential users are asked to read through the manual and to describe aloud what they are seeing and learning. More controlled experiments with groups of users may help authors to make design decisions about the manual. In such studies, subjects are assigned tasks; their time to completion, error rates, and subjective satisfaction are the dependent variables.

Field trials with moderate numbers of users constitute a further process for identifying problems with the user manual and the software. Field trials can range from 1/2 hour with a half-dozen people to several months with thousands of users. One effective and simple strategy is for field-trial users

to mark up the manual while they are using it. They can thus rapidly indicate typos, misleading information, and confusing sections.

Software and the accompanying manuals are rarely completed. Rather, they go into a continuous process of evolutionary refinement. Each version eliminates known errors, adds refinements, and extends the functionality. If the users can communicate with the manual writers, then there is a greater chance of rapid improvement. Most manuals offer a tear-out sheet for users to indicate comments for the manual writers. This device can be effective, but other routes should also be explored: electronic mail, interviews with users, debriefing of consultants and instructors, written surveys, group discussions, and further controlled experiments or field studies.

Brockmann (1990) offers a nine-step process for writing user documentation:

1. Develop the document specifications:
    - Use task orientation
    - Use minimalist design
    - Handle diverse audiences
    - State the purpose
    - Organize information and develop visualizations
    - Consider layout and color
2. Prototype
3. Draft
4. Edit
5. Review
6. Field test
7. Publish
8. Perform postproject review
9. Maintain

12.4   Preparation of Online Facilities

There is a great attraction to making technical manuals available on the computer. The positive reasons for doing so are these:

- Information is available whenever the computer is available. There is no need to locate the correct manual—which activity could cause a minor disruption if the proper manual is close by, or a major disruption if the manual must be retrieved from another building or person. The harsh reality is that many users lose their manuals or do not keep the manuals current with new versions of the software.

- Users do not need to allocate physical work space to opening up manuals. Paper manuals can be clumsy to use and can clutter a workspace.
- Information can be electronically updated rapidly and at low cost. Electronic dissemination of revisions ensures that out-of-date material cannot be retrieved inadvertently.
- Specific information necessary for a task can be located rapidly if the online manual offers electronic indexing or text searching. Searching for one page in hundreds can usually be done more quickly on a computer than with printed material.
- Authors can use graphics, sound, color, and animations that may be helpful in explaining complex actions and creating an engaging experience for users.

However, these positive attributes can be compromised by several potentially serious negative side effects:

- Displays may not be as readable as printed materials (see Section 12.2).
- Each display may contain substantially less information than a sheet of paper, and the rate of paging is slow compared to the rate of paging through a manual. The display resolution is lower than that for paper, which is especially important when pictures or graphics are used.
- The user interface of help systems may be novel and confusing to novices. By contrast, most people are thoroughly familiar with the "user interface" of paper manuals. The extra mental effort required for navigating through many screens may interfere with concentration and learning.
- Splitting the display between work and help or tutorial windows reduces the space for work displays. If users must switch to a separate application, then the burden on the user's short-term memory can be large. Users lose their context of work and have difficulty remembering what they read in the online manual. Multiple displays or windows provide a potential resolution for this problem.

Even recent studies (Hertzum and Forkjaer, 1996) have shown that paper manuals can yield faster learning. Still, the online environment opens the door to a variety of helpful facilities (Roesler and McLellan, 1995) that might not be practical in printed form. Relles and Price (1981) offer this list:

- Successively more detailed explanations of a displayed error message
- Successively more detailed explanations of a displayed question or prompt
- Successive examples of correct input or valid commands
- Explanation or definition of a specified term
- A description of the format of a specified command

- A list of allowable commands
- A display of specified sections of documentation
- A description of the current values of various system parameters
- Instruction on the use of the system
- News of interest to users of the system
- A list of available user aids

Houghton (1984) reviews online help facilities and points out the great difficulty in helping the novice user to get started, as well as helping the expert user who needs one specific piece of information. Kearsley (1988) offers examples, empirical data about online help systems, and these guidelines:

- Make the help system easy to access and easy to return from.
- Make helps as specific as possible.
- Collect data to determine what helps are needed.
- Give users as much control as possible over the help system.
- Supply different helps for different types of users.
- Make help messages accurate and complete.
- Do not use helps to compensate for poor interface design.

Results demonstrating the efficacy of online help facilities come from a study with 72 novice users of a text editor (Cohill and Williges, 1982). A control group receiving no online help facilities was compared with groups in eight experimental conditions formed from all combinations of initiation (user versus computer causes the help session to begin), presentation (printed manual versus online), and selection of topics (user versus computer selects which material is displayed). The control group with no online facilities performed significantly less well than did the experimental groups (Table 12.2). Of the eight experimental groups, the best performance was achieved by the user-initiated, user-selected, and printed-manual group.

Magers (1983) revised an online help facility to offer context-sensitive help instead of keyword-indexed help, wrote tutorial screens in addition to the reference material, reduced computer jargon, used examples instead of a mathematical notation, provided an online dictionary of command synonyms, and wrote task-oriented rather than computer-oriented help screens. Thirty computer novices were split into two groups; half received the original and half the modified help facility. The subjects with the revised help achieved a task score of 90.6, compared with 43.0 for the other subjects. Time was reduced from 75.6 to 52.0 minutes with the improved help facility. Subjective-satisfaction scores also strongly favored the revised help facility.

In spite of improvements, many users wish to avoid paper or online manuals, and prefer to learn system features by exploration (Rieman, 1996).

Table 12.2

Results from a study comparing nine styles of online help facilities (Cohill and Williges, 1982).

Help Configurations

Initiation	Presentation	Selection	Time in Subtask	Errors per Subtask	Commands per Subtask	Subtasks Completed
User	Manual	User	293.1	0.4	8.4	5.0
User	Manual	System	442.2	2.0	17.7	4.9
User	Online	User	350.9	1.1	13.5	5.0
User	Online	System	382.2	1.8	17.6	4.9
System	Manual	User	367.9	1.3	13.1	4.8
System	Manual	System	399.1	0.9	13.8	4.9
System	Online	User	425.9	2.8	15.1	5.0
System	Online	System	351.5	1.2	13.7	4.9
Control: No HELP Available			679.1	5.0	20.2	3.4

Nonetheless, for most contemporary software, failure to provide online manuals or help will be seen as a deficiency by most reviewers and users. The form and content of the online help facility make a profound difference. Good writing, task orientation, context sensitivity, and appropriate examples contribute to improved online manuals and help.

12.4.1   Online manuals

Developers of traditional paper manuals are often proud of their work, and may be tempted to load the text automatically to make it available online. This course is attractive, but the results are likely to be less than optimal. Page layouts for paper may not be convertible to a useful online or web formats, and dealing with the figures automatically is risky. The automatic conversion to online or web text is most attractive if the users have a display large enough to show a full page of text; then, precise images of the printed text can be scanned in with text, figures, photographs, page numbers, and so on. A close match between printed and online manuals can be useful, but if the quality of the displayed image is significantly lower than that of the printed version, users may prefer the paper.

Online manuals can be enhanced by availability of string search, multiple indices, tables of contents, tables of figures, electronic bookmarks, annotation, hypertext traversal, and automatic history keeping (Fig. 12.6). The designers will be most effective if they can redesign the manuals to fit the electronic
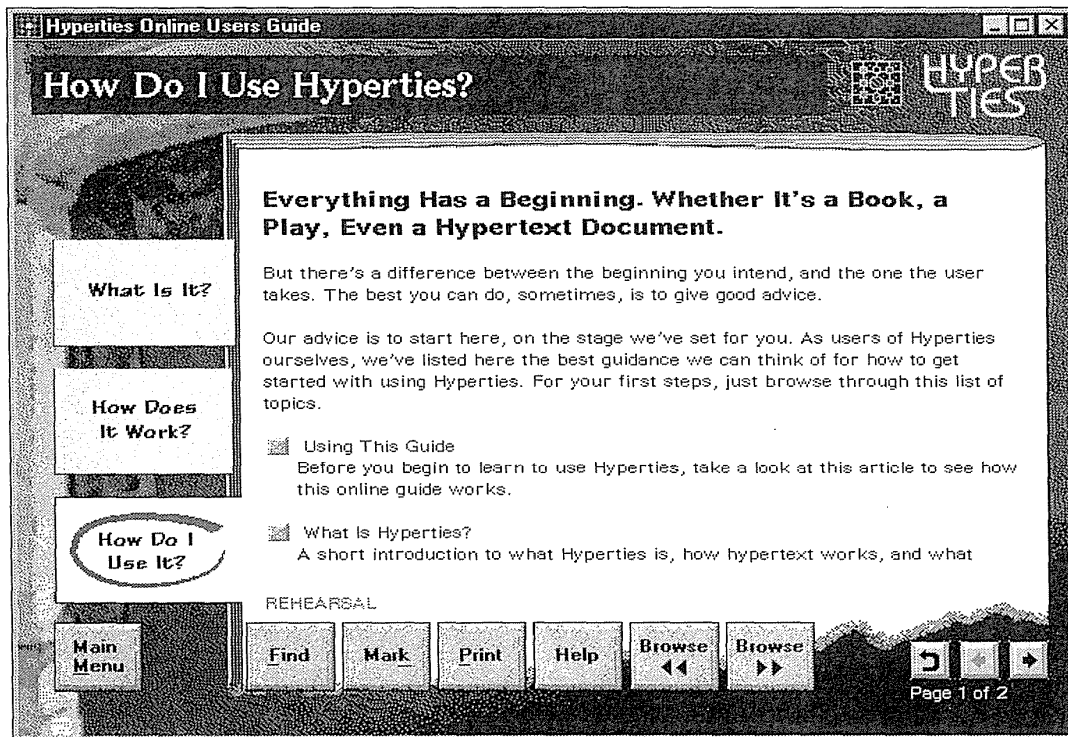
Figure 12.6

Online Manual for Hyperties hypertext electronic-publishing system, using three topic categories related to the user's goals described by Duffy and colleagues (1992). (Used with permission of Cognetics Corp., Princeton Junction, NJ.)

medium and to take advantage of multiple windows, text highlighting, color, sound, animation, and especially string search with relevance feedback.

Several workstation manufacturers have attempted to put their user manuals online. Symbolics was an early success, with 4000 pages online in such a convenient browser that many purchasers never removed the shrink-wrap plastic from their printed manuals (Walker, 1987). The growing availability plus low production and shipping costs of CD-ROMs has encouraged hardware suppliers to produce browsers for online manuals that are exact images of the printed manuals. Apple put its six-volume *Inside Macintosh* series for developers onto a single CD-ROM with scanned images and hypertext links. This HyperCard product took only 1 month of intense effort to develop (Bechtel, 1990). Another Apple (1993) innovation was to create a CD-ROM guide for interface designers with more than a hundred animations of poor, good, and better designs.

A vital feature for online manuals is a properly designed table of contents that can remain visible to the side of the page of text. Selection of a chapter or other entry in the table of contents should immediately produce the appropriate page on the display. Expanding or contracting tables of contents (Egan et al., 1989) or multiple panes to show several levels at once are beneficial (see Fig. 13.10) (Chimera and Shneiderman, 1994).

A more primitive approach to online manuals is the Unix man facility, which has textual descriptions and associated options for each command. Users must know the command names to find the material, but a clever and simple system called apropos helps substantially. The apropos file contains the name of each Unix command with a carefully written one-line description. Users can type apropos sort to get this listing, and then can display the manual pages:

sortm	**sort messages**
comm	select, reject lines common to two
	sorted files
look	find lines in a sorted file
qsort	quick sort
scandir, alphasort	scan a directory
sort	sort or merge files
sortbib	sort bibliographic database
tsort	topological sort

This approach can work some of the time, but seems to be more effective for experienced than for novice users.

Online help that offers concise descriptions of the interface syntax and semantics is probably most effective for intermittent knowledgeable users, but is likely to be difficult for novices who have more need for tutorial training. The traditional approach is to have the user type or select a help-menu item, and to display six or 60 or 600 alphabetically arranged command or menu names for which there is a paragraph or more of help that the user can retrieve by typing help followed by the command name. This method can also work, but it is often frustrating for those users who are not sure of the correct name for the task they wish to accomplish; for example, the name might be search, query, select, browse, find, reveal, display, info, or view. They may see several familiar terms but not know which one to select to accomplish their task. Worse still, there may not be a single command that accomplishes the task, and there is usually little information about how to assemble commands to perform tasks, such as converting graphics into a different format.

Designers can improve *keyword lists* by clustering keywords into meaningful categories, and indicating a starter set of commands for novices. Users

may also be able to set the level of detail and the kind of information (for example, descriptions with or without options, examples, complete syntax) that they obtain about each command.

Two other useful lists might be of *keystrokes* or *menu items*. Each might have an accompanying feature description, such as the first few lines from the help list of the early keyboard-oriented WordPerfect:

Key	Feature	Key Name
Ctrl-F5	Add Password	Text In/out,2
Shft-F7	Additional Printers	Print,S
Shft-F8	Advance Up, Down or Line	Format,4
Ctrl-PgUp	Advanced Macros	Macro Commands
Ctrl-F10	Advanced Macros, Help on	Macro Definition
Shft-F8	Align/Decimal Character	Format,4

Many designers recognized that the keyword lists are overwhelming and that users probably want to have local information about the task that they are performing. They developed *context-sensitive help*, in which users get different messages from the help processor, depending on where they are in the software. Help requested from inside a dialog box produces a window with information about that dialog box, preferably in a nearby pop-up window.

Another approach to context-sensitive help in form-fillin or menu systems is to have the user position the cursor and then to press F1 or a help key to produce information about the item on which the cursor is resting. A subtle variation is to use a mouse to click on a help button or a ? to turn the cursor into a question mark. Then, the cursor is dropped on a field, icon, or menu item, and a pop-up window describes that item. In an accelerated version of this technique, the user simply drags the cursor over items, causing a small windows to pop up with explanations of those items. This strategy is the idea behind Apple's balloon help (Fig. 12.7).

A variant of balloon help is to turn on all the balloons at once, so that the user can see all the explanations simultaneously. A mode switch might change from the balloons to a set of marks that indicate which parts of the display are clickable, double-clickable, draggable, and so on.

Searching through the full text of online manuals is increasingly rapid, and the user-interface strategies are being refined steadily. An expanding and contracting table of contents was combined with string search and relevance feedback indicating number of "hits" on the table-of-contents listing (Egan et al., 1989). A series of three empirical studies showed the effects of several improvements and the advantage over print versions of the same document. The electronic version was advantageous, especially when the
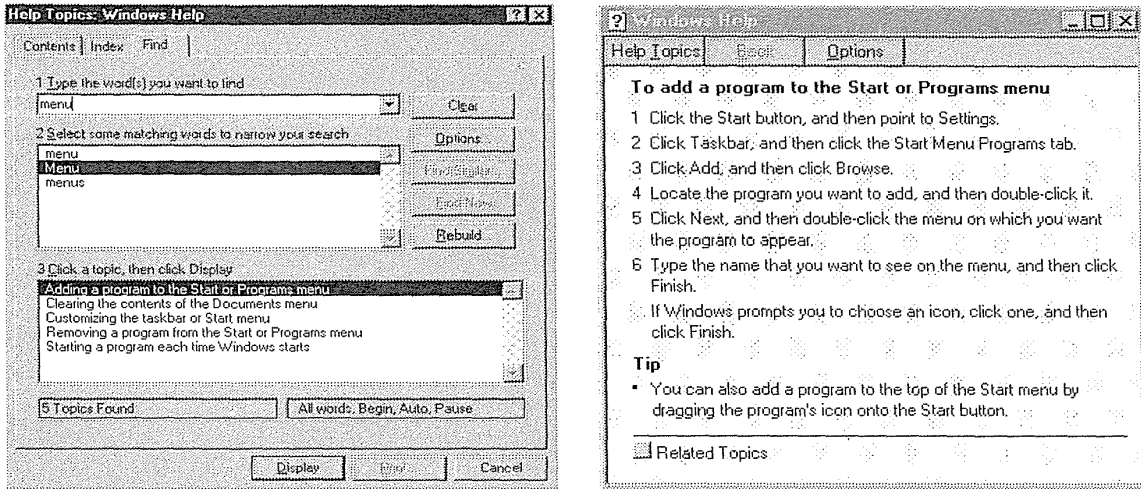
Figure 12.7

Apple's balloon help pops up a single balloon with an explanation of the item on which the cursor is resting on. Users can turn off the balloon help. (Used with permission of Apple Computer, Cupertino, CA.)

search questions contained words that were in the document headings or text. Browsing strategies were found to be most effective in a study of 87 computer-science students, but search by keywords proved to be a useful complement (Hertzum and Frokjaer, 1996).

The Microsoft Windows 95 help facility offers users access to large numbers of hyperlinked articles. Users begin by selecting a topic word; then, they can scroll a list of relevant article titles to find the most appropriate one. The help articles are task oriented with step-by-step instructions, but the large number of articles and the complexity of information often makes it hard for novices to get what they need. Some level structuring of the help articles to limit the novices to seeing novice topics would be an improvement. Popular features are the Wizards and CueCards that guide users through sequences of actions to accomplish a task such as preparing a macro.

The online help in Microsoft Windows 95 offers four ways of finding relevant articles, called *topics*. Users can browse a meaningfully organized table of contents that lists the topics hierarchically. They can also browse an alphabetical list of terms and topics, or find a topic by typing the first few letters of a keyword (Fig. 12.8). This huge list is searched rapidly, and variant

Figure 12.8

Microsoft Windows 95 offers elaborate online help, including an index, bookmarks, definition buttons, and other features to assist navigation. (Used with permission of Microsoft Corp., Redmond, WA.)

forms of capitalization are shown for the exact words as they appear in the topics. Finally, the Answer Wizard approach allows users to type a request using natural-language statements; the program then selects the relevant keywords and offers a list of topics organized into three categories. For example, typing "Tell me how to print addresses on envelopes" produces:

How Do I

Print an address on an envelope
Change the size of an envelope
Print envelopes by merging an address list
...

Tell Me About

Field codes: UserAddress field
Form letters, envelopes, and mailing labels
Making a list of names and addresses for a mail merge
...

Programming and Language Reference

Mail Merge statements and functions
Tool statements and functions

This example query produces an effective result as the first topic, but other topics are not appropriate. Results will vary depending on the situation.

12.4.2   Online tutorials, demonstrations, and animations

First-time users of a software package need an interactive tutorial environment in which the computer instructs the user to carry out commands right on the system. One introductory tutorial for the Lotus 1-2-3 package displays the exact keystrokes the user must type, and then carries out the commands. The user can type the exact keystrokes or just keep pressing the space bar to speed through the demonstration. Adobe's PhotoDeluxe includes an online tutorial that leads users through the multiple steps needed for graphical image manipulation (Fig. 12.9). Some users find this guided approach attractive; others are put off by the restrictive sequencing that prevents errors and exploration.

Online tutorials can be effective because the user (Al-Awar et al., 1981)

- Does not have to keep shifting attention between the terminal and the instructional material
- Practices the skills needed to use the system
- Can work alone at an individual pace and without the embarrassment of mistakes made before a human instructor or fellow students

The opportunity for carrying out practice tasks as part of an online tutorial is one of the latter's greatest strengths. Getting users to be active is one of the key tenets of the minimal-manual approach, and it applies especially well to
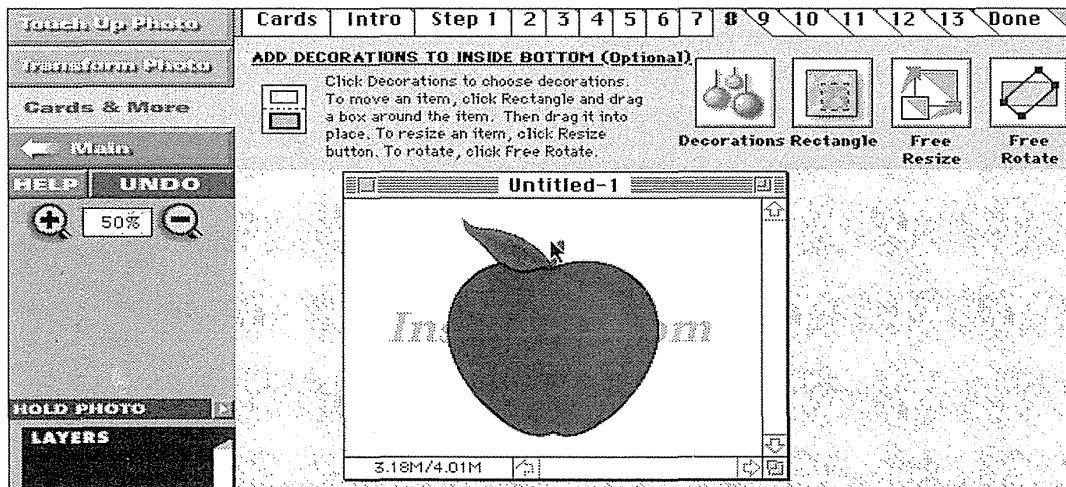
Figure 12.9

This online tutorial for Adobe PhotoDeluxe leads users through the multiple steps needed for graphical image manipulation. It shows three levels of menus simultaneously, giving users flexibility in moving rapidly among tasks. (Used with permission of Adobe Systems, Seattle, WA.)

online tutorials. Subjects who carried out online exercises learned better than did those who used only guided exploration (Wiedenbeck et al., 1995).

Creators of interactive tutorials must address the usual questions of instructional design and also the novelty of the computer environment. A library of common tasks for users to practice is a great help. Sample documents for word processors, slides for presentation software, and maps for geographic-information systems help users to experience the application. Repeated testing and refinement is highly recommended for tutorials.

Demonstration (demo) disks have become a modern high-tech art form. Someday soon, someone should start a museum of demo disks to preserve these innovative, flashy, and slick byproducts of the computer era. Demo disks are designed to attract potential users of software or hardware by showing off system features using the best animations, color graphics, sound, and information presentation that advertising agencies can produce. The technical requirement that the demo fit on a single diskette makes development challenging, but this requirement is giving way as demos are put on CD-ROMs or on the World Wide Web. Animated text (words zooming, flipping, or spinning), varied transitions (fades, wipes, mosaics, or dissolves), sound effects, bright graphics, and finally an address or telephone number to use for ordering the software are typically included. The user-interface requirements are to capture and maintain the users' interest, while conveying information and building a positive product image. Automatic pacing or manual control satisfies hands-off or hands-on users, respectively. Sessions should be alterable to suit the user who wants a three-minute introduction and the user who wants a one-hour in-depth treatment. Additional control to allow users to stop, replay, or skip parts adds to the acceptability.

Animations as part of online help are increasingly common as hardware improves and competition increases (Apple, 1993). A simple and ingenious approach is to animate the action icons in a display to give a quick demonstration of usage (Baecker et al., 1991). An artist created brief animations for the 18 icons in the HyperCard tool menu (paint brush, lasso, eraser, and so on) that ran within each icon's 20- by 22-pixel box. A usability test was conducted to refine the designs and to demonstrate their effectiveness: "In every case in which static icons were not understood, the dynamic icons were successful in conveying the purpose of the tool." Another approach to animated help showed sequences of menu or icon selections that performed a complex task, such as moving a block of text (Sukaviriya and Foley, 1990). This project produced a working system that generated the animations automatically from task descriptions. The benefits of animations for learners of user interfaces are still unclear, although users do enjoy this presentation style (Palmiter and Elkerton, 1991; Payne et al., 1992; Harrison, 1995).

12.4.3   Helpful guides

Sometimes, friendly guides, such as the marketing manager for the software, or a famous personality related to the content, or a cartoon character for children, can lead users through a body of knowledge. A pioneering effort was the GUIDES 3.0 project, in which a Native American chief, a settler wife, and a cavalry man appear as small photographs on the display to guide readers through the materials by offering their points of view on the settling of the American West (Oren et al., 1990). When selected, the guides tell their stories through video sequences from laser disk. In addition, a modern woman is available in TV format to help guide the readers through using the system. This approach does not anthropomorphize the computer, but rather makes the computer a medium of communication, much like a book enables the author to speak to readers by way of the printed page.

Introductions to online services such as CompuServe or America Online, web sites such as for the Library of Congress, and Bill Gates's CD-ROM book *The Road Ahead* (1995) welcome new users and offer guidance about which features to begin using. Audio tours of art galleries have become popular at many museums. An informed and engaging curator such as J. Carter Brown can lead visitors through the National Gallery of Art in Washington, D.C., but users can control the pace and replay sections. The well-designed CD-ROM *A Passion for Art* has several authoritative guides explaining the software, history, and impressionist art in the Barnes collection (Corbis, 1995). A still photo of the speaker is accompanied by spoken text to guide the users through the software and the collection.

Natural-language dialog was proposed for interactive learning about an operating-system command language, but this strategy has not proved to be effective (Shapiro and Kwasny, 1975; Wilensky et al., 1984). An optimistic discussion of advice-giving expert systems (Carroll and McKendree, 1987) laments the lack of behavioral research and focuses on questions such as these: In what ways do people voluntarily restrict their use of natural language when interacting with a recognition facility? Can user models that incorporate learning transitions and trajectories (as well as end states) be developed? A simulated "intelligent help" system was tested with eight users doing business tasks, such as printing a mailing list (Carroll and Aaronson, 1988). The researchers prepared messages for expected error conditions, but they found that "people are incredibly creative in generating errors and misconceptions, and incredibly fast." The results, even with a simulated system, were mixed; the authors concluded that "development of intelligent help systems faces serious usability challenges." In a Smalltalk programming environment, cartoonlike gurus appear onscreen and offer audio commentaries with animated demonstrations of the GUI (Alpert et al., 1995). The designers consider many of the problems of anthropomorphic help, such as user initiation, pacing, and user control of remediation; unfortunately, however, no empirical evidence of efficacy is offered.

The newer and apparently more effective approach is to have a help network, in which email is used to support question asking and responses (Eveland, 1994; Ackerman, 1994; Ackerman and Palen, 1996). Electronic-mail help questions can be sent to a designated help desk or staff person, or to a general list within an organization. Responses can be received in seconds or, more typically, minutes or hours, but users must publicly expose their lack of knowledge and risk getting incorrect advice. In one simple example a broadcast message produced the answer in 42 seconds:

```
Time: 18:57:10 Date: Fri Oct 29, 1993
From: <azir>
after i change a list to a group, how long before
I can use it?

Time: 18:57:52 Date: Fri Oct 29, 1993
From: starlight on a moonless night <clee>
you can use it immediately
```

Increasingly the communal broadcast approach is appealing because of low cost, and respondents have a sense of satisfaction from being able to help and demonstrate their abilities. The social nature of computing networks encourages question asking. Recording questions and answers into files of *frequently asked questions (FAQs)* enables newcomers to browse typical problems discussed in the past. For example, this question and brief answer come from the Netscape Mail Server FAQ:

```
Can the Mail Server deliver mail using UUCP?

No, the Mail Server doesn't support UUCP at present. We
recommend the use of more current standards, such as PPP
and SLIP.
```

12.5   Practitioner's Summary

Paper manuals and online help can determine the success or failure of a software product. Sufficient personnel, money, and time should be assigned to these support materials. User manuals and online help should be developed before the implementation to help the development team to define the interface and to allow adequate time for testing. Both manuals and online help segments should be tailored to specific user communities and to accomplishment of specific goals (offer task instruction or describe interface objects and actions). Instructional examples should be realistic, encourage prompt action, use consistent terminology, and support error recognition and recovery. Online manuals and help are increasingly attractive, as screen resolution,

size, and speed increase, but designers should minimize extra commands, preserve the context of work, and avoid forcing memorization of information. Online guides or tutorials can lend a human touch, if they contain presentations by real humans. Help networks provide a powerful low-cost support mechanism.

12.6   Researcher's Agenda

The main advantage of online materials is the potential for rapid retrieval and traversal of large databases, but little is known about how to offer this advantage conveniently without overwhelming the user. Layered approaches, in which users can set their level of expertise, seem helpful but are untested. The cognitive model of turning pages in a book is too simple, but if elaborate hyperlinked networks are used, disorientation is a danger. Users' navigation among online help segments should be recorded and studied, so that we can gain a better understanding of what help segments are effective. Multiple windows help users by allowing the users to see the problem and the online help or tutorial at the same time, but better automatic layout strategies are needed. Cognitive models of how different classes of people learn to use computer systems need refinement.

World Wide Web Resources | WWW |

The World Wide Web contains many online manuals and tutorials that can be studied. There are also guidelines for writing better manuals and some empirical studies.

http://www.aw.com/DTUI

References

Ackerman, Mark S., Augmenting the organizational memory: A field study of Answer Garden, *Proc. Conference on Computer Supported Cooperative Work '94*, ACM, New York (1994), 243–252.

Ackerman, Mark S. and Palen, Leysia, The Zephyr help instance: Promoting ongoing activity in a CSCW system, *Proc. CHI '96 Human Factors in Computer Systems*, ACM, New York (1996), 268–275.

Al-Awar, J., Chapanis, A., and Ford, W. R., Tutorials for the first-time computer user, *IEEE Transactions on Professional Communication*, PC-24, (1981), 30–37.

Allwood, C. M. and Kalen, T., Evaluating and improving the usability of a user manual, *Behaviour & Information Technology*, 16, 1 (January–February 1997), 43–57.

Alpert, Sherman R., Singley, Mark K., and Carroll, John M., Multiple multimodal mentors: Delivering computer-based instruction via specialized anthropomorphic advisors, *Behaviour & Information Technology*, 14, 2 (1995), 69–79.

Apple Computer, *Making it Macintosh*, Cupertino, CA (1993), CD-ROM animated guide.

Baecker, Ronald, Small, Ian, and Mander, Richard, Bringing icons to life, *Proc. CHI '91 Human Factors in Computer Systems*, ACM, New York (1991), 1–6.

Bechtel, Brian, Inside Macintosh as hypertext. In Rizk, A., Streitz, N., and Andre, J. (Editors), *Hypertext: Concepts, Systems and Applications*, Cambridge University Press, Cambridge, U.K. (1990), 312–323.

Brockmann, R. John, *Writing Better Computer User Documentation: From Paper to Hypertext: Version 2.0*, John Wiley and Sons, New York (1990).

Cakir, A., Hart, D. J., and Stewart, T. F. M., *Visual Display Terminals: A Manual Covering Ergonomics, Workplace Design, Health and Safety, Task Organization*, John Wiley and Sons, New York (1980).

Carroll, John M., Minimalist training, *Datamation*, 30, 18 (1984), 125–136.

Carroll, John M., *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*, MIT Press, Cambridge, MA (1990).

Carroll, John M. and Aaronson, Amy P., Learning by doing with simulated intelligent help, *Communications of the ACM*, 31, 9 (September 1988), 1064–1079.

Carroll, John M. and Mack, R. L., Learning to use a word processor: By doing, by thinking, and by knowing. In Thomas, J. C., and Schneider, M. (Editors), *Human Factors in Computing Systems*, Ablex, Norwood, NJ (1984), 13–51.

Carroll, John M. and McKendree, Jean, Interface design issues for advice-giving expert systems, *Communications of the ACM*, 30, 1 (January 1987), 14–31.

Chimera, R. and Shneiderman, B., Evaluating three user interfaces for browsing tables of contents, *ACM Transactions on Information Systems*, 12, 4 (October 1994), 383–406.

Cohill, A. M. and Williges, Robert C., Computer-augmented retrieval of HELP information for novice users, *Proc. Human Factors Society—Twenty-Sixth Annual Meeting* (1982), 79–82.

Corbis Publishing, *A Passion for Art*, Bellevue, WA (1995).

Creed, A., Dennis, I., and Newstead, S., Effects of display format on proof-reading on VDUs, *Behaviour & Information Technology*, 7, 4 (1988), 467–478.

Duffy, Thomas, Palmer, James, and Mehlenbacher, Brad, *Online Help Systems: Theory and Practice*, Ablex, Norwood, NJ (1992).

Egan, Dennis E., Remde, Joel R., Gomez, Louis M., Landauer, Thomas K., Eberhardt, Jennifer, and Lochbum, Carol C., Formative design-evaluation of SuperBook, *ACM Transactions on Information Systems*, 7, 1 (January 1989), 30–57.

Eveland, J. D., Blanchard, Anita, Brown, William, and Mattocks, Jennifer, The role of "help networks" in facilitating use of CSCW tools, *Proc. Conference on Computer Supported Cooperative Work '94*, ACM, New York (1994), 265–274.

Foss, D., Rosson, M. B., and Smith, P., Reducing manual labor: An experimental analysis of learning aids for a text editor, *Proc. Human Factors in Computer Systems*, ACM, Washington, D.C. (March 1982).

Gates, Bill, *The Road Ahead*, Viking Penguin, New York (1995).

Gould, John, and Grischkowsky, Nancy, Doing the same work with hardcopy and with cathode ray tube (CRT) terminals, *Human Factors*, 26 (1984), 323–337.

Gould, J., Alfaro, L., Barnes, V., Finn, R., Grischkowsky, N., and Minuto, A., Reading is slower from CRT displays than from paper: Attempts to isolate a single-variable explanation, *Human Factors*, 29, 3 (1987a), 269–299.

Gould, J., Alfaro, L., Finn, R., Haupt, B., and Minuto, A., Reading from CRT displays can be as fast as reading from paper, *Human Factors*, 29, 5 (1987b), 497–517.

Grandjean, E. and Vigliani, E. (Editors), *Ergonomic Aspects of Visual Display Terminals*, Taylor and Francis, London (1982).

Grant, Allan, Homo quintadus, computers and ROOMS (repetitive ocular orthopedic motion stress), *Optometry and Vision Science*, 67, 4 (1990), 297–305.

Hansen, Wilfred J., Doring, Richard, and Whitlock, Lawrence R., Why an examination was slower on-line than on paper, *International Journal of Man–Machine Studies*, 10, (1978), 507–519.

Hansen, Wilfred J. and Haas, Christine, Reading and writing with computers: A framework for explaining differences in performance, *Communications of the ACM*, 31, 9 (1988), 1080–1089.

Harrison, Susan M., A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphic user interface, *Proc. CHI '95 Conference: Human Factors in Computing Systems*, ACM, New York (1995), 82–89.

Heines, Jesse M., *Screen Design Strategies for Computer-Assisted Instruction*, Digital Press, Bedford, MA (1984).

Helander, Martin G., Design of visual displays. In Salvendy, Gavriel (Editor), *Handbook of Human Factors*, John Wiley and Sons, New York (1987), 507–548.

Hertzum, Morten and Frokjaer, Erik, Browsing and querying in online documentation: A study of user interfaces and the interaction process, *ACM Transactions on Computer–Human Interaction*, 3, 2 (June 1996), 136–161.

Horton, William K., *Designing and Writing Online Documentation: Help Files to Hypertext*, John Wiley and Sons, New York (1990).

Houghton, Raymond C., Online help systems: A conspectus, *Communications of the ACM*, 27, 2 (February 1984), 126–133.

Jorna, Gerard C., Image quality determines differences in reading performance and perceived image quality with CRT and hard-copy displays, *Proc. Human Factors Society—Thirty-Fifth Annual Meeting*, Human Factors Society, Santa Monica, CA (1991), 1432–1436.

Kearsley, Greg, *Online Help Systems: Design and Implementation*, Ablex, Norwood, NJ (1988).

Mager, Robert F., *Preparing Instructional Objectives*, Fearon, Palo Alto, CA (1962).

Magers, Celeste S., An experimental evaluation of on-line HELP for non-programmers, *Proc. CHI '83 Conference: Human Factors in Computing Systems*, ACM, New York (1983), 277–281.

Oborne, David J. and Holton, Doreen, Reading from screen versus paper: There is no difference, *International Journal of Man–Machine Studies*, 28, (1988), 1–9.

Oren, Tim, Salomon, Gitta, Kreitman, Kristee, and Don, Abbe, Guides: Characterizing the interface. In Laurel, Brenda (Editor), *The Art of Human–Computer Interface Design*, Addison Wesley, Reading, MA (1990), 367–381.

Palmiter, Susan and Elkerton, Jay, An evaluation of animated demonstrations for learning computer-based tasks, *Proc. CHI '91 Conference: Human Factors in Computing Systems*, ACM, New York (1991), 257–263.

Parton, Diana, Huffman, Keith, Pridgen, Patty, Norman, Kent, and Shneiderman, Ben, Learning a menu selection tree: Training methods compared, *Behaviour and Information Technology*, 4, 2 (1985), 81–91.

Payne, S. J., Chesworth, L., and Hill, E., Animated demonstrations for exploratory learners, *Interacting with Computers*, 4, (1992), 3–22.

Price, Jonathan, and staff, *How to Write a Computer Manual*, Benjamin/Cummings, Addison-Wesley, Reading, MA (1984).

Relles, Nathan and Price, Lynne A., A user interface for online assistance, *Proc. Fifth International Conference on Software Engineering*, IEEE, Silver Spring, MD (1981).

Rieman, John, A field study of exploratory learning strategies, *ACM Trans. on Computer–Human Interaction*, 3, 3 (September 1996), 189–218.

Roemer, Joan M. and Chapanis, Alphonse, Learning performance and attitudes as a function of the reading grade level of a computer-presented tutorial, *Proc. Human Factors in Computer Systems*, ACM, Washington, D.C. (1982), 239–244.

Roesler, A. W. and McLellan, S. G., What help do users need? Taxonomies for on-line information needs and access methods, *Proc. CHI '95 Conference: Human Factors in Computing Systems*, ACM, New York (1995), 437–441.

Shapiro, Stuart C. and Kwasny, Stanley C., Interactive consulting via natural language, *Communications of the ACM*, 18, 8 (August 1975), 459–462.

Spencer, C. J. and Yates, D. K., A good user's guide means fewer support calls and lower support costs. *Technical Communication*, 42, 1 (1995), 52.

Strunk, William, Jr. and White, E. B., *The Elements of Style* (Third Edition), Macmillan, New York (1979).

Sukaviriya, Piyawadee "Noi" and Foley, James D., Coupling a UI framework with automatic generation of context-sensitive animated help, *Proc. User Interface Software and Technology*, 3, ACM, New York (1990), 152–166.

Sullivan, Marc A. and Chapanis, Alphonse, Human factoring a text editor manual, *Behaviour and Information Technology*, 2, 2 (1983), 113–125.

van der Meij, Hans and Carroll, John M., Principles and heuristics in designing minimalist instruction, *Technical Communication* (Second Quarter 1995), 243–261.

van der Meij, Hans and Lazonder, Ard W., Assessment of the minimalist approach to computer user documentation, *Interacting with Computers*, 5, 4 (1993), 355–370.

Walker, Janet, Issues and strategies for online documentation, *IEEE Transactions on Professional Communication PC*, 30 (1987), 235–248.

Wiedenbeck, Susan, Zila, Patti L., and McConnell, Daniel S., End-user training: An empirical study comparing on-line practice methods, *Proc. CHI '95 Conference: Human Factors in Computing Systems*, ACM, New York (1995), 74–81.

Wilensky, R., Arens, Y., and Chin, D., Talking to UNIX in English: An overview of UC, *Communications of the ACM*, 27, 6 (June 1984), 574–593.

Wright, P. and Lickorish, A., Proof-reading texts on screen and paper, *Behaviour and Information Technology*, 2, 3 (1983), 227–235.
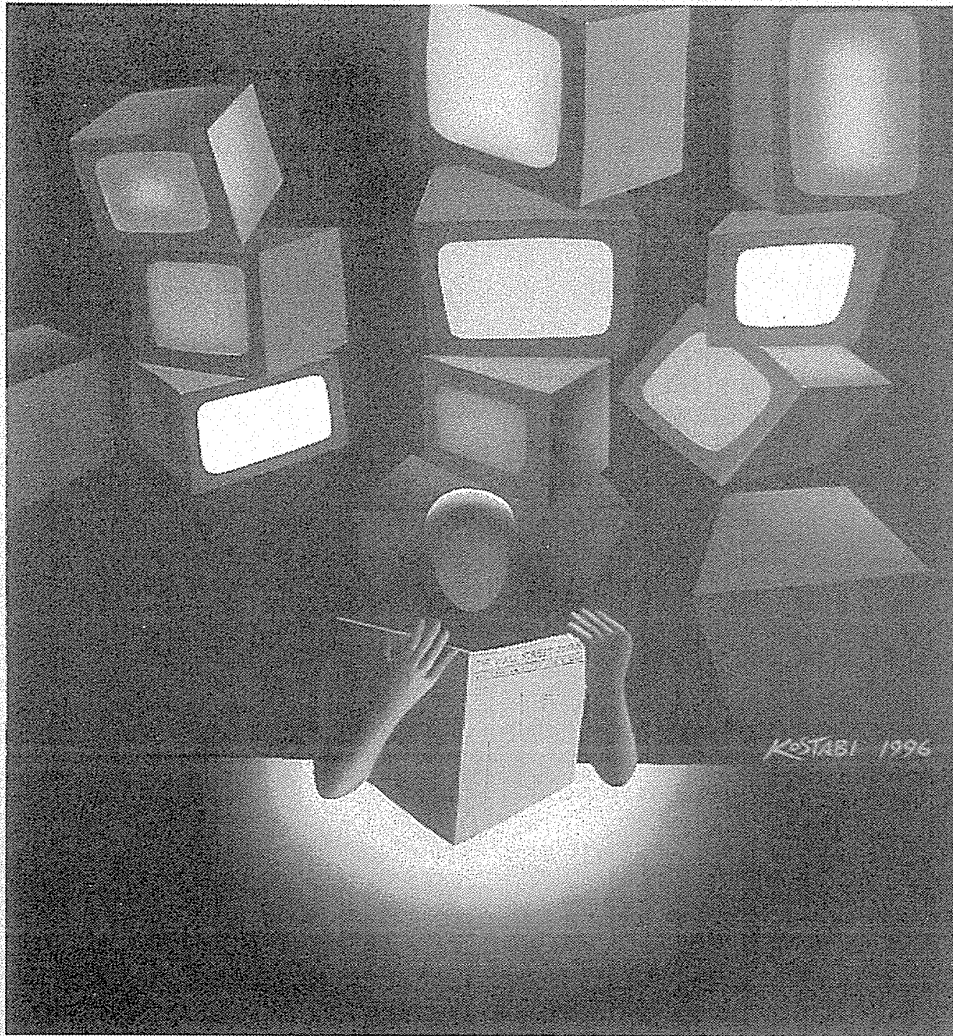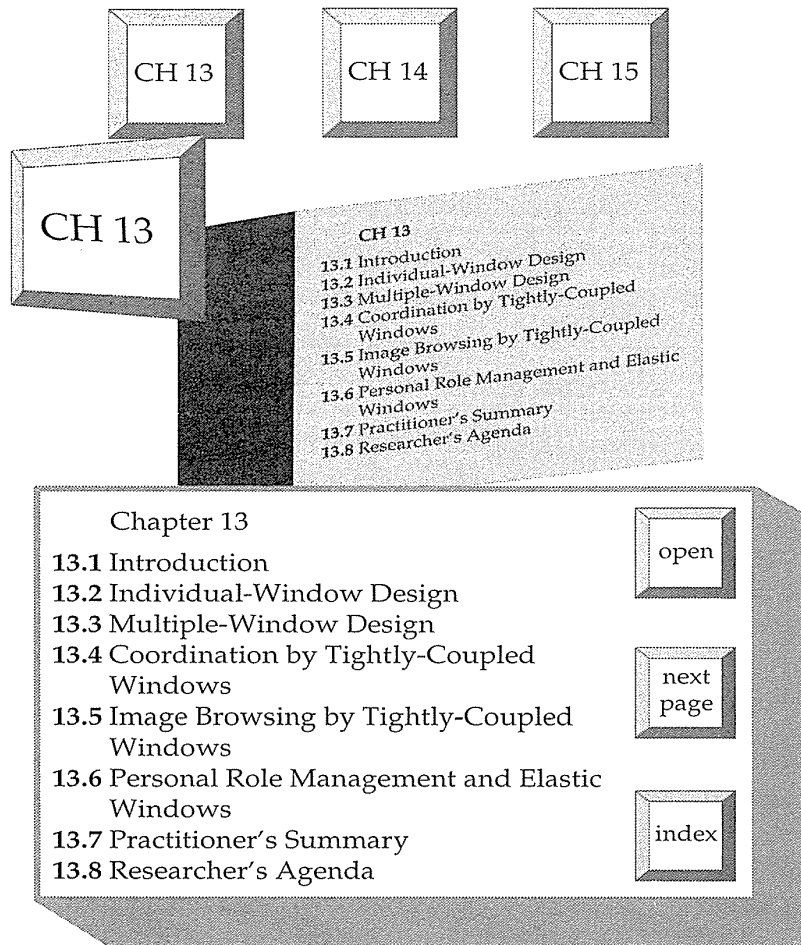
Mark Kostabi, *Market Research*, 1996

Multiple-Window Strategies

Through even the smallest window the eye can reach the most distant horizon.

A. Bergman, *Visual Realities,* **1992**

CH 13   CH 14   CH 15

CH 13

open

next
page

index

13.1 Introduction

The output of early computers was printed by Teletype on an ever-growing scroll of paper. As designers switched to displays, the need to go back was sometimes supported by electronic scrolling of the session. This technique is useful, but designers became aware of similar situations in which users have to jump around to related text or graphics. Programmers have to jump from procedural code to data declarations, or from procedure invocations to procedure definitions. Authors of scientific papers jump from writing the text to adding a bibliographic reference to reviewing empirical data to creating figures to reading previous papers. Airline reservationists jump from working on a client itinerary to reviewing schedules to choosing seat assignments.

The general problem for many computer users is the need to consult multiple sources rapidly, while minimally disrupting their concentration on their task. With large desk- or wall-sized displays, many related documents can be displayed simultaneously, but visibility and eye–head movement might be a problem. With small displays, windows are usually too small to provide adequate information or context. In the middle ground, with 20–70 cm displays (approximately 640 × 480 to 2048 × 2048 pixels), it becomes a design challenge to offer users sufficient information and flexibility to accomplish their tasks while reducing window housekeeping actions, distracting clutter, and eye–head movement. The animation characteristics, three-dimensional appearance, and graphic design play key roles in efficacy and acceptance (Gait, 1985; Kobara, 1991; Marcus, 1992).

If users' tasks are well understood and regular, then there is a good chance that an effective *multiple-window display* strategy can be developed. The airline reservationist might start a client-itinerary window, review flight segments from a schedule window, and drag selected flight segments to the itinerary window. Windows labeled Seat Selection or Food Preferences might appear as needed, and then the charge-card information window would appear to complete the transaction. When the sequence is varied and unpredictable, users will need to have more control of the layout and will need more training.

Window housekeeping is an activity related to the interface and not directly related to the user's task. If window-housekeeping actions can be reduced, then users can complete their tasks more rapidly. In an empirical test with eight experienced users, the windowed version of a system produced longer task-completion times than did the nonwindowed (full-screen) environment (Bury et al., 1985). Multiple smaller windows led to more time arranging information on the display and more scrolling activity to bring necessary information into view. However, after the time to arrange the display was eliminated, the task-solution times were shorter for the windowed environment. Fewer errors were made in the windowed environment. These results suggest that there are advantages to using windows, but these advantages may be compromised unless effective window arrangement is provided.

On small displays with poor resolution, opportunities for using multiple windows are limited because users are annoyed by frequent horizontal and vertical scrolling. With medium-resolution displays and careful design, multiple windows can be practical and esthetically pleasing. Window-border decorations can be made to be informative and useful. On larger, high-resolution displays, windows become still more attractive, but the manipulation of windows can remain as a distraction from the user's task. Opening windows, moving them around, changing their size, or closing them are the most common operations supported (Card et al., 1984; Myers, 1988).

The visual nature of window use has led many designers to apply a direct-manipulation strategy (see Chapter 6) to window actions. To stretch,

move, and scroll a window, users can point at appropriate icons on the window border and simply click on the mouse button (Billingsley, 1988; Kobara, 1991; Marcus 1992). Since the dynamics of windows have a strong effect on user perceptions, the animations for transitions (zooming boxes, sequencing of repainting when a window is opened or closed, blinking outlines, or highlighting during dragging) must be designed carefully.

It is hard to trace the first explicit description of windows (Hopgood et al., 1985), although several sources credit Doug Engelbart with the invention of the mouse, windows, outlining, collaborative work, and hypertext as part of his pioneering NLS system during the mid-1960s (Engelbart, 1988). Movable, overlapping windows appeared in the Smalltalk graphical environment (Fig. 13.1) as it evolved in the 1970s at Xerox PARC, with contributions from Alan Kay, Larry Tesler (1981), Daniel Ingalls, and others. In 1981, the highly graphical Xerox Star (Fig. 13.2) (Smith et al., 1982; Johnson et al., 1990) allowed up to six nonoverlapping windows (with limited size control and movement, but with no dragging of windows or icons) to cover the desktop, plus multiple property sheets to overlay temporarily parts of the windows or desktop. Soon after, the Apple Lisa and, in 1984, the Apple Macintosh (see Fig. 1.1) made popular their style of GUI with overlapping windows (Apple, 1987).
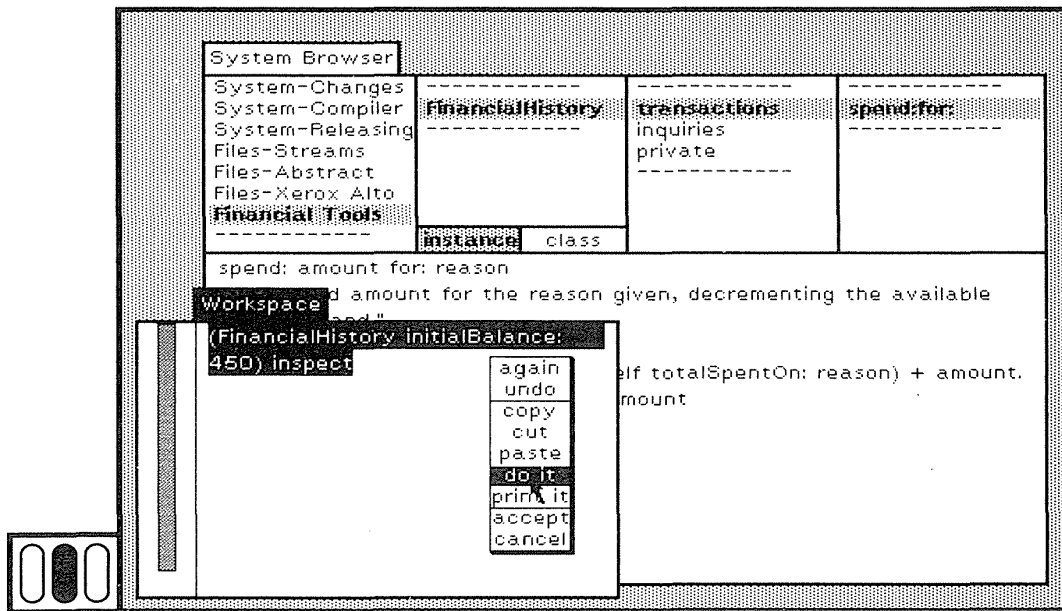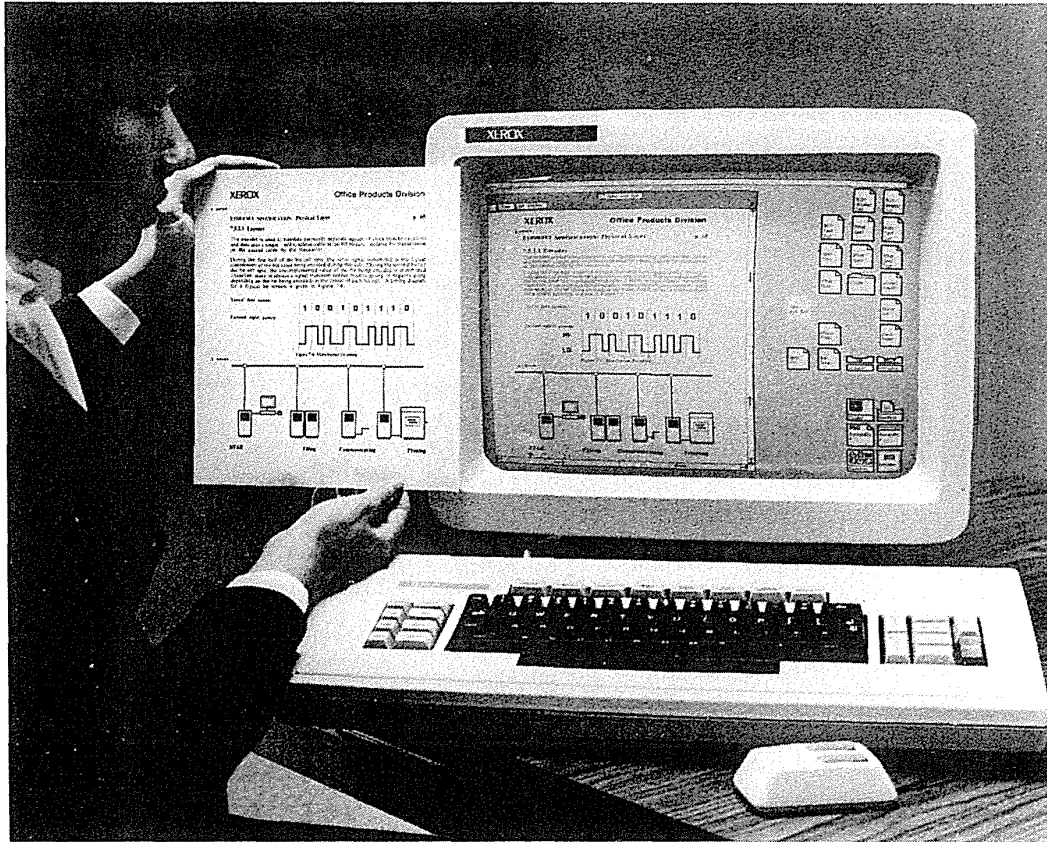

Figure 13.1

Many versions of Smalltalk were created in the 1970s, but the user interface is remembered for overlapping windows, a multipane hierarchical browser, window titles that stick out like tabs, pop-up menus for window actions, and an unorthodox scroll bar. (Courtesy of Parc Place Systems, Mountain View, CA.)
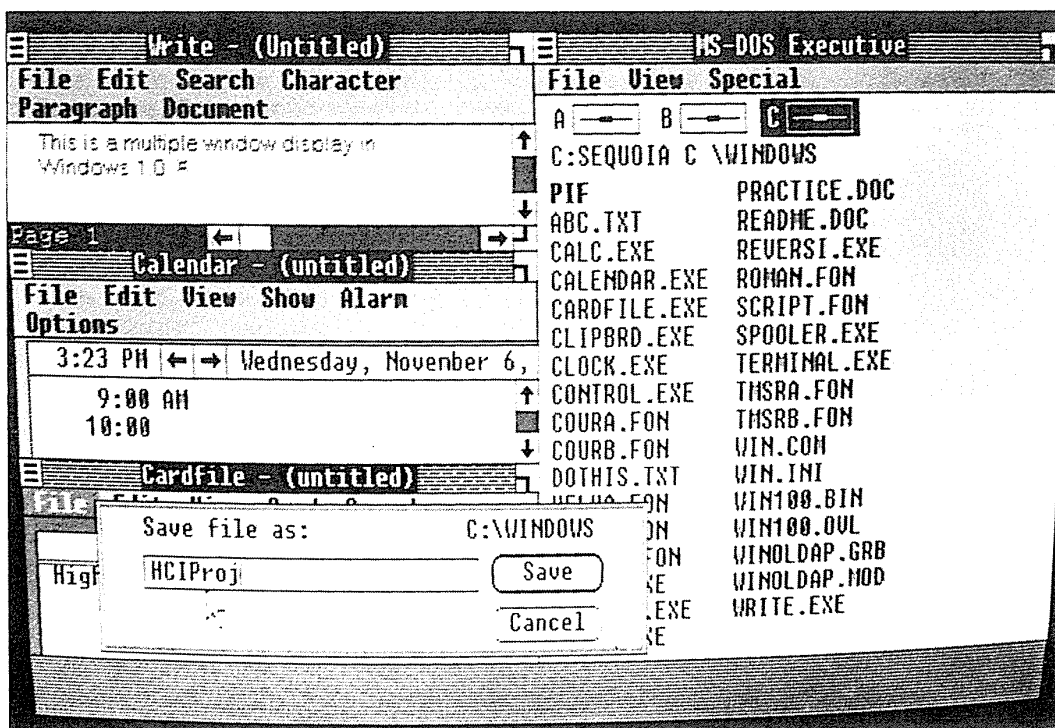
Figure 13.2

The Xerox 8010 Star. The Star played a leading role in popularizing high-resolution, WYSIWYG editing for document preparation in office environments. The Star system pioneered the use of the display screen as an electronic desktop, as well as commercially introducing the mouse, symbolic icons, multiple display windows, and a bitmapped display that showed graphics in detail. (Courtesy of Xerox Corp., Rochester, N.Y.)

Microsoft followed with the graphical MS Windows 1.0 (tiled windows) (Fig. 13.3), 2.03 (Fig. 13.4), 3.0 (Fig. 13.5), and Windows 95 (Color Plate A1) for IBM PCs, while IBM offered OS/2.

The notion of collections of windows assembled into *rooms* is an important step forward in matching window strategies to users' tasks (Henderson and Card, 1986; Card and Henderson, 1987). Users can open and leave in one room a set of windows for reading electronic mail; another room might have a set of windows for composing an article or a program. Rooms can be seen as a form of window macro that enables users to specify actions on several windows at a time. Hewlett-Packard's HP-VUE implements the rooms idea

Figure 13.3

Microsoft Windows 1.0, which permitted variable size/place/number and space-filling tiling only. Dialog boxes could appear on top of the tiled windows. (Reprinted with permission from Microsoft Corporation, Redmond, WA.)

as a set of workspaces that users can visit. Sun Microsystems offers virtual workspaces with an overview window to support navigation among clusters of windows. An effective historic overview of windowing strategies is available in videotape form (Myers, 1990).

Much progress has been made, but there is still an opportunity to reduce dramatically the housekeeping chores with individual windows and to provide task-related multiple-window coordination. Innovative features, inventive borders or color combinations, individual tailoring, programmable actions, and cultural variations should be expected.

13.2   Individual-Window Design

The *MS Windows 3.0 User's Guide* (1990) identifies a window as "a rectangular area that contains a software application, or a document file. Windows can be opened and closed, resized and moved. You can open several of them
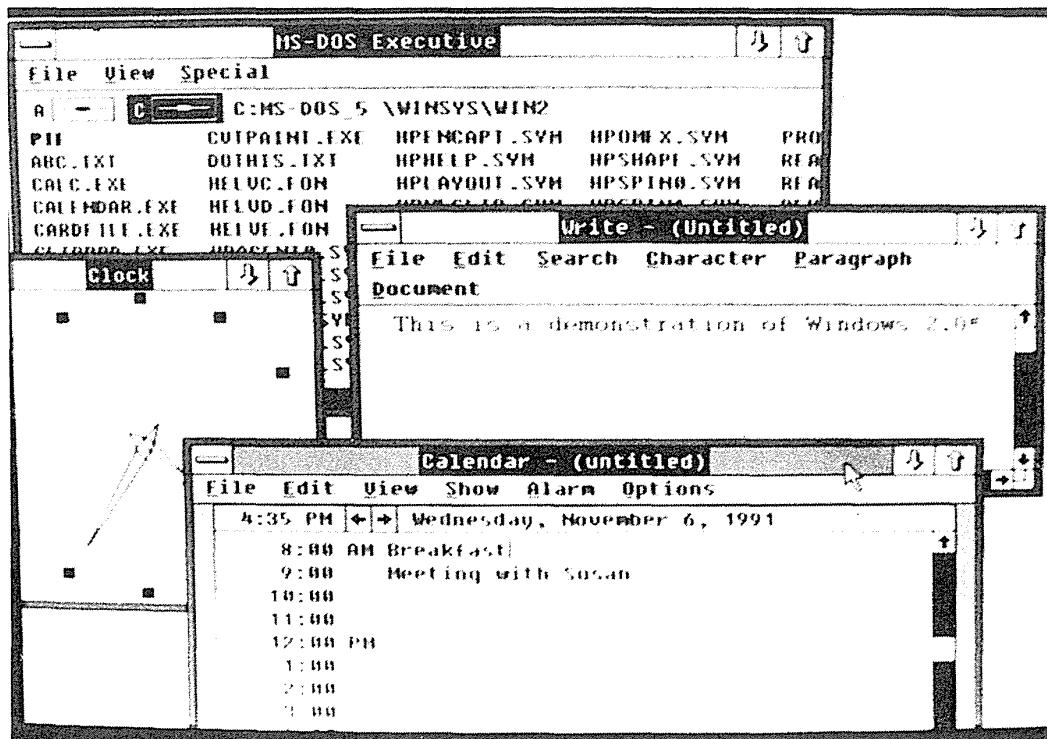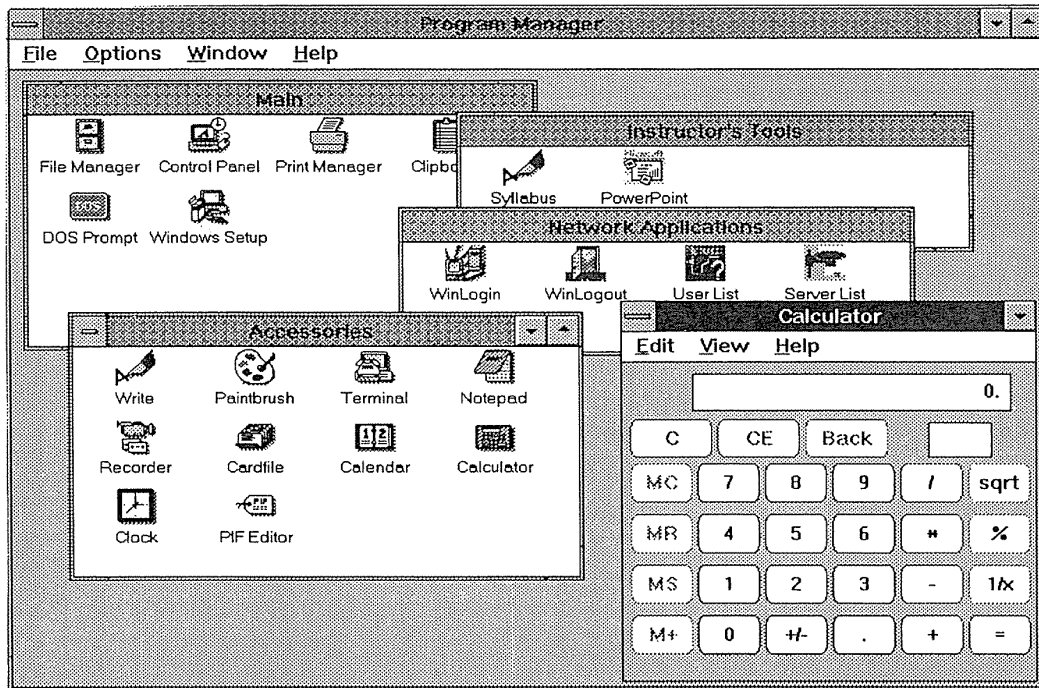
Figure 13.4

Microsoft Windows 2.0, which permitted arbitrary overlapping windows. (Reprinted with permission from Microsoft Corporation, Redmond, WA.)

on the desktop at the same time and you can shrink windows to icons or enlarge them to fill the entire desktop." Although some people might disagree with aspects of this definition (for example, windows are not always rectangular, and other window actions are possible), it is useful. Window interface objects include:

- *Titles*   Most windows have an identifying title at the top center, top left, or bottom center, or on a tab that extends from the window (Fig. 13.6). Tabbed window titles can be helpful in locating a window on a cluttered desktop. To save window space, designers may create some windows with no titles. Title bars may change shading or color to show which window is currently active (the active window is the one that receives keystrokes from the keyboard). When a window is closed, it may be represented as an icon, and it may show a title to its right, below it and centered, or below it and left justified. Other approaches show the titles in a pop-up menu list or as a tab sticking out of a pile of windows.
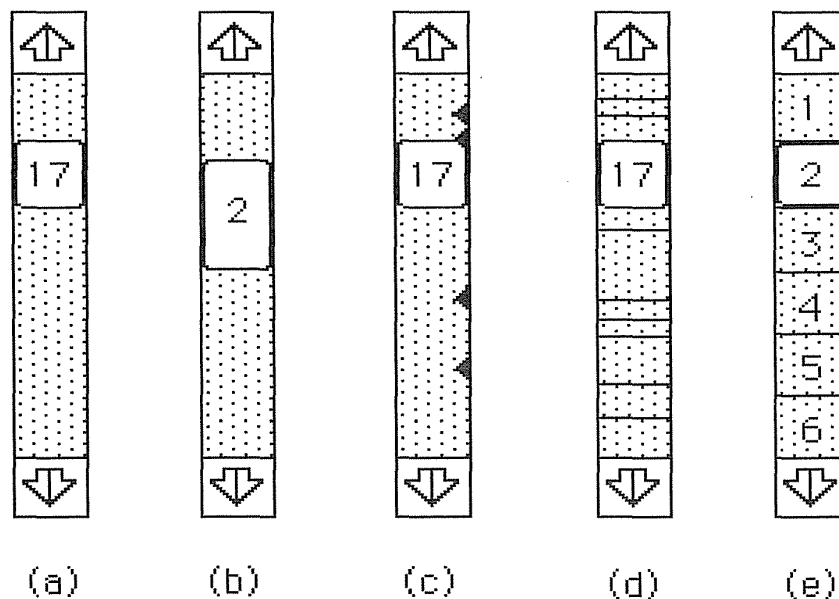
Figure 13.5

Microsoft Windows 3.0 and 3.1x, which had higher-resolution displays and overlapping windows, and became enormously popular. (Reprinted with permission from Microsoft Corporation, Redmond, WA.)
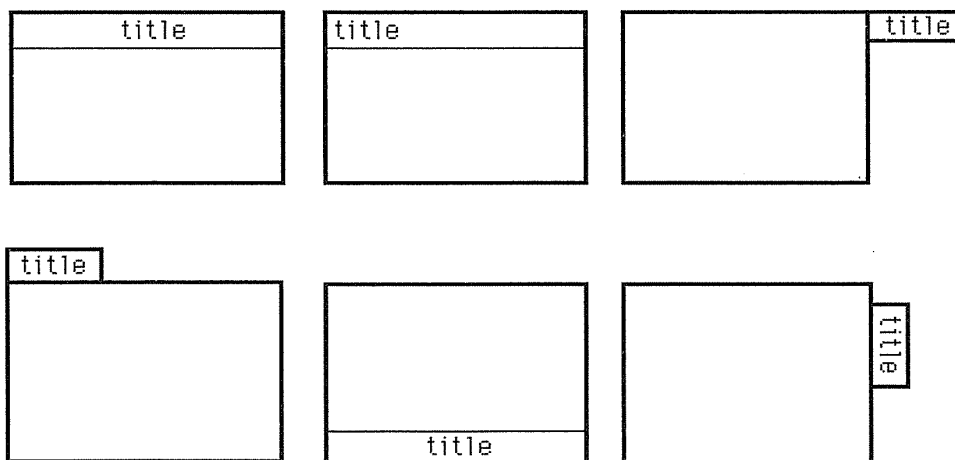
Figure 13.6

Titles may be inside the window in various locations, or protruding from it in various positions.

- *Borders or frames*   The window border or frame may be one or more pixels thick to accommodate selection for resizing or to distinguish windows from the background. Several systems use three-dimensional lighting models and may show a shadow below each window. Three-dimensional buttons and icons on the borders have become popular. This three-dimensional effect is attractive to many users (although some users find it distracting), but it may also use precious pixels that could be devoted to window contents. Border thickness or color changes can be used to highlight the currently active window.

- *Scroll bars*   Since a window may be small compared to its contents, some method for moving the window over the contents or moving the contents under the window is needed. The basic operation of a scroll bar is to move up or down and left or right, but many variations have been implemented. Small and large motions must be supported, incremental and destination actions are appreciated, and feedback is necessary to help users formulate their plans (Fig. 13.7a). Most scroll bars have some form of up and down arrows on which the user can click to produce a small motion, such as a single-line scroll. An important feature is to permit smooth scrolling when the up or down arrow is selected continuously (for example, when the user holds down the

(a)   (b)   (c)   (d)   (e)

Figure 13.7

Scroll bars showing existing and proposed features. (a) Page number in scroll box. (b) Proportional scroll box. (c) Selectable position markers. (d) Value bar showing sections. (c) Page bar with discrete positions.

mouse button). This strategy is preferable to repeated mouse clicks, which distract the user from reading the contents. Scrolling by a full window or page turning is often supported by clicks above or below the scroll box. Users can get to a specific destination in a document, such as the end, by dragging the scroll box to the desired destination.

Feedback in scroll bars is important for ensuring confidence and correct operations. An important approach uses proportional scroll boxes that indicate what portion of the document is visible currently. Another appreciated feature displays a page number inside the scroll box, so that, as users drag the scroll box, they can see where they are in a document (Fig. 13.7b). If the scroll bar is used on a list, such as alphabetically organized document names, then the scroll box can show the first letter of the names.

There is room for improvement of scroll bars. For example, it might be nice to mark a particular position and to see a small triangle on the scroll bar (Fig. 13.7c). Then, merely clicking on the triangle would cause the scroll bar and the window's contents to jump to that location. Scroll bars can contain document-visualization tools (see Chapter 15) using techniques such as *value bars* to show document section boundaries (Fig. 13.7d) (Chimera, 1992) and *page bars* to facilitate discrete page turning (Fig. 13.7e).

Scroll-bar arrows usually indicate the direction for movement of the window, but they could also represent the direction of movement of the contents. An early study (Bury et al., 1982) showed that a majority of users thought that a down arrow meant that the window moved down to show later portions of a document, but many users are still confused and make errors. Recovery of errors is rapid, but creative designers might still pursue visual cues to give users a better indication of which motion will occur.

Window interface actions include:

- *Open action*   A window can be opened from its icon or text-menu list onto the display by a typed command, a menu selection, a voice command, or a double click. The icons can be of varying size (.5 to 3 cm square) and labels can be placed in varying locations.

Feedback during opening can vary from simply having the window appear after the full display repaints, to blanking or blackening of the window destination followed by appearance of the window border and then the contents. Visually appealing animations are possible, such as zoom boxes (animated series of growing boxes emanating from the icon

growing into the window), zoom lines (streaks, dots, or other representations of light going to the corners of the window), window-shade opening (the window appears to pull down like a window shade), or three-dimensional flips or spins from the icon to the full window (Silicon Graphics Iris). Open actions may have accompanying sounds.

- *Open place and size*   A key determinant of the usability of window systems is the choice of where the window opens. Most window systems support the most-recently-used place and size approach, which has a better chance to satisfy user needs than a fixed position. Often, the most effective solution is to open the new window close to the current focus (icon, menu item, field, and so on) to limit eye motion, but far enough away to avoid obscuring the current focus. For example, if a control-panel icon is selected, then the control panel could appear just below the icon. Similarly, if a fillin field for a form is selected with a help action, the help window should appear to the side but should not obscure the field in question.

- *Close action*   Windows may have a small icon (typically in the upper-left or upper-right corner) to close them and show animation in a way that is symmetric with the open action. Feedback during closing varies from none (which can be a problem for users who do not know where the icon is located) to zoom boxes (animated series of shrinking boxes moving toward where the window rests on the desktop as an icon), or three-dimensional flips or spins of the window as it shrinks to an icon. Most systems close windows smoothly and rapidly; in slower systems, however, awkward sequences of display painting can be unsettling—for example, if the window is cleared in strips that break up the window frame or the contents.

- *Resize action*   There is great diversity in approaches to resizing a window. The Macintosh permits sizing only from a size box on the lower-right corner, whereas MS Windows, OSF/Motif, OS/2, and many other systems permit sizing by all four corners and by each of the four sides. Some systems will resize adjacent windows automatically. An interesting question is whether or not resizing the window causes reformatting of text (reflowing of documents or changing fontsize), graphics (size changes to ensure that the full object is seen no matter what the window size) or icon layouts (icons are moved to ensure that they remain visible in a smaller window). Another way in which window systems vary is on size limits. Some systems allow windows to be made extremely small (for example, 1×2 centimeters); other systems require windows to be larger than a certain minimum (for example, 4×6 centimeters, or big enough to show the contents). Most window systems have window-size upper bounds that are as large as the display.

- *Move action*   There is also great diversity in approaches to moving a window. The Xerox STAR and MS Windows 1.0 had a Move menu item; users selected it and then clicked on the destination, but the results were sometimes surprising because of the complex layout strategies. The Macintosh designers use the entire title bar as a handle and have the users drag an outline of the window until satisfied with the placement. A variety of visual feedback has been created. Display rates have become faster, and some systems now support displays of the full window as it is dragged. Some systems require that the full window be visible on the display, whereas others permit portions of a window to be off the display in three (typically left, right, and bottom), or in all four, directions. Some systems constrain a window (the child) to be contained within another window (the parent).

- *Bring forward or activation*   When overlapping windows are used, users need a mechanism to bring forward and make active a window that is totally or partially obscured by other windows. Approaches to bringing a window forward include clicking on a menu list of open windows to select one, selecting an action such as Top from a pop-up menu associated with each window, clicking in any part of a window (or a restricted part, such as the title bar), or simply moving the cursor into a window. Variations include the possibility that all windows are active with text flowing into the window that contains the cursor, and that a window is made active but is not brought to the top. Activation may be shown by changes to the border (color or thickness changes), title (color or stripes are added), or text background (brightness increases). An important part of activation is the smoothness and sequence of the painting process. Although some systems are so rapid that the entire window and its contents appear virtually instantaneously (in less than 100 milliseconds), many systems may clear the area first, then paint the border, and finally fill the window top to bottom. Awkward painting strategies—such as painting different-colored parts of a frame one at a time, or filling the window from the bottom to the top—can be unsettling.

These basic window-interface objects and actions are shared by many systems, but there are numerous variations and extensions. An interesting extension is to use spoken commands and speech-recognition technology (see Section 9.4) to control window actions. An initial version of an X Window implementation, called Xspeak, was tested by four users to explore useful features (Schmandt et al., 1990). The authors concluded that "navigation in a window can be handled with speech input," but, in the current implementation they "found the use of voice in navigation an incomplete substitute for the mouse."

13.3 Multiple-Window Design

The challenge of providing access to multiple sources of information has stimulated many solutions:

- *Multiple monitors* Stock-market traders or process-control operators sometimes use multiple physical monitors because that is the only way to see all the information needed with the hardware available. Experience suggests that a smaller number of higher-resolution monitors with multiple windows is superior in most cases, because the distraction of eye motion across the gaps between monitors slows work.

- *Rapid display flipping* Another alternative is rapid alternation or flipping among displays, automatically or by user control. This strategy can be helpful, but it too has been shown to place greater burdens on users to recognize where they are, to know the commands, to formulate a plan to reach the desired display, and to execute the plan. User-controlled rapid display flipping can be usefully applied with knowledgeable users to supplement windowing. Automatic page flipping can be useful in public-access information systems, but airport designers have recognized that, if budget and space permit, having a bank of six or eight displays for departures and another six or eight displays for arrivals is superior to using an automatic page-flipping strategy on one display.

- *Split displays* Many early word processors enabled users to split their display to show two (or more) parts of a document, or two (or more) documents. Split displays were available in early text-oriented systems such as emacs, WordPerfect (two windows), MS Word on IBM-PCs (eight windows), and many other word processors. Splits could be made horizontally (to create two full-width windows) or vertically (to allow side-by-side comparisons—but this arrangement was effective for only those files with narrow lists). A split display is a simplified approach to multiple windows, and offers fewer features than window systems.

- *Space-filling tiling with fixed number, size, and place* Simple display splits are often described as being *tiled*, since the display space is often covered completely with rectangular sections resembling the ceramic tiles on a floor. Tiling usually is meant to convey space filling and no overlapping, but there are many variations. Making a simple dichotomy between tiling and overlapping ignores many interesting variations. The simplest case is a fixed number of fixed-sized and fixed-placement tiles—for example, two, four, six, or eight rectangles filling the display—with the possible exception of some control or icon region.

- *Space-filling tiling with variable size, place, and number*   A common strategy is to start with a single large window and, when a second window is opened, to cut the first one in half horizontally or vertically to make space for the second. Microsoft Windows 1.0 used this tiling strategy with splits occurring when a document or application icon was dragged to the horizontal or vertical borders to cause a vertical or horizontal splits, respectively. Moving a window was possible, but the results were often surprising to users. Similarly, closing a window in a variable-number space-filling tiled layout could produce unexpected results, such as other windows growing to use the space, windows moving, or simply blank space appearing.

- *Non–space-filling tiling*   Variations emerged that did not require the entire display to be covered. The Xerox Star allowed blank space on the right half of the two-column display. In the original Xerox Star, the first window to open would be a full-page size on the left side of the display (see Fig. 13.2); then, the second window would cause the left side of the display to be shared by upper and lower halves. The third window would result in each window getting one-third of the left side of the display. The fourth through sixth (a maximum of six was allowed) windows would fill the right side of the display. This strategy avoided narrow windows that require annoying and frequent horizontal scrolling.

- *Piles of tiles*   Variations on the basic tiling strategy include the piles-of-tiles strategy, in which windows are stacked on top of one another, so that tiles can be popped to reveal previously used windows. Typically, piles of tiles are of fixed size and fixed position to simplify usage. Subsequent windows are placed on the least recently used pile, with tabs protruding to allow selection.

- *Window zooming*   Since users often need to expand a window temporarily, some systems offer a convenient feature to enlarge a window—even to full display size—and then to shrink the window back to the previous size. This technique, often used in slide-presentation programs that show thumbnails of the slide show, allows users to zoom in on a single slide (Fig. 13.8).

- *Arbitrary overlaps*   There is a great appeal to seeing multiple windows on a display with the appearance of partial and arbitrary overlaps. With this now popular strategy, windows can be moved to any point on the display, and portions of the window may be off the display, clipped by the display boundary. This approach has been called *two-and-one-half dimensional programming* to characterize the appearance of multiple windows overlapping as though they were floating one above the other. This strategy was used in Smalltalk, the Apple Lisa, the Apple Macintosh, MS Windows 2.03, and many later window systems. Arbitrary overlapping windows can be advantageous if independent tasks are
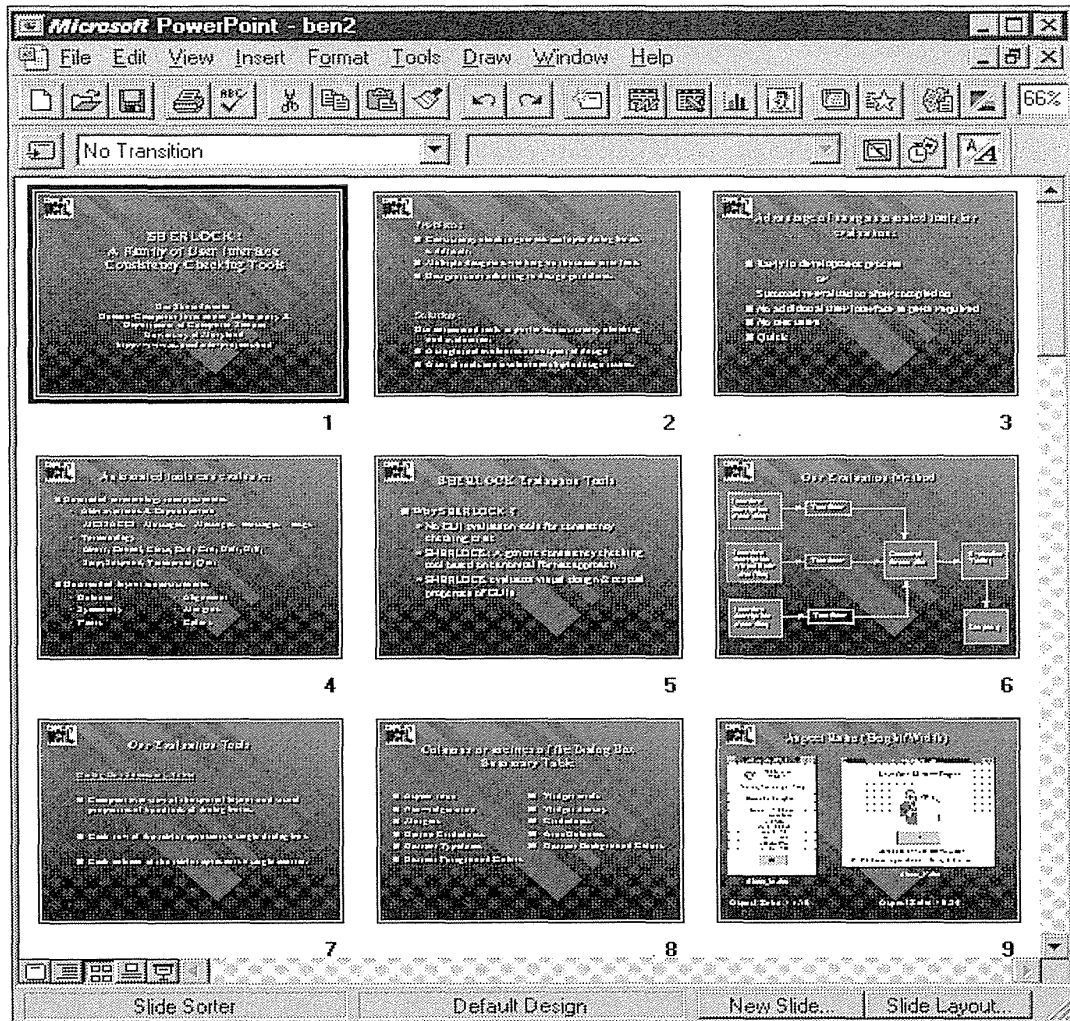
Figure 13.8

Microsoft PowerPoint display showing an overview of a slide show by thumbnail views that the user can select to produce full-sized images. (Reprinted with permission from Microsoft Corporation, Redmond, WA.)

being carried out. For example, in the middle of using a word processor, a user can decide to send a piece of electronic mail, to use a calculator, or to consult a personal schedule. The user can pop up a new window, take care of the task, and return to the main task without losing context or restarting the work. However, overlapping windows have the potential to obscure relevant material and to increase the housekeeping load.

- *Cascades*   Designers have applied the familiar deck-of-cards metaphor by positioning a sequence of windows from the upper left down to the

lower right (or from the lower left up to the upper right). Successive windows are offset below (or above) and to the right to allow each window title to remain visible. Some systems automatically lay out successively opened windows in the cascade. Other systems allow users to select a cascade action from a menu, which places the currently open windows in a cascade, but newly opened windows appear elsewhere on the display. Tombaugh et al. (1987) demonstrated that, with sufficient practice, users could answer questions more rapidly from a multiple-window cascade representing chapters of a book than they could from a paged single window.

There are certainly other approaches to designing multiple windows, and a comprehensive guiding theory is still needed. A theory might emerge from computational geometry or from a more task-related model, such as the working-set model (Card, 1989).

Empirical studies are needed to clarify the issues and to establish methods for measuring performance and ability in windowing environments. Enthusiasts of the greater flexibility of arbitrary overlaps may have been disappointed by an empirical comparison with a tiled approach of the early Xerox Star (Bly and Rosenberg, 1986). Tasks requiring little window manipulation were carried out more quickly with the tiled strategy, but other tasks were carried more quickly with the overlapping strategy by some users. Overlapping window manipulation appeared to require the users to have experience before they could use it effectively. Overall, the authors suggest that, for users with a higher level of expertise, tiled windows may be better.

We implemented program browsing with a multiwindow hypertext environment using a tiled approach using automatic window placement plus zooming (Seabrook and Shneiderman, 1989). Typical program-exploration tasks were performed more rapidly than with a standard single-window editor.

13.4   Coordination by Tightly-Coupled Windows

Designers may break through to the next generation of window managers by developing coordinated windows, in which windows appear, change contents, and close as a direct result of user actions in the task domain (Norman et al., 1986; Shneiderman et al., 1986). For example, in medical insurance-claims processing, when the agent retrieves information about a client, such fields as the address, telephone numbers, and membership numbers should

appear on the display. Simultaneously, and with no additional commands, the medical history might appear in a second window, and the record of previous claims might appear in a third window. A fourth window might contain a form for the agent to complete to indicate payment or exceptions. Scrolling the medical-history window might produce a synchronized scroll of the previous claims window to show related information. When the claim is completed, all window contents are saved and the windows are closed. Such sequences of actions can be established by designers, or by users with end-user programming tools.

Coordination is a task concept that describes how information objects change based on user actions. *Tight coupling* among windows is the interface concept that supports coordination. A careful study of user tasks can lead to task-specific coordinations based on sequences of actions; there are also certain generic coordinations that might be supported by interface developers:

- *Synchronized scrolling*   A simple coordination is synchronized scrolling, in which the scroll bar of one window is tightly coupled to another scroll bar, and action on one scroll bar causes the other to scroll the associated window contents. This technique is useful for comparing two versions of a program or document. Synchronization might be on a line-for-line basis, on a proportional basis, or keyed to matching tokens in the two windows. Another way to do synchronization might be as an option on an open action that would specify two windows to open side by side with a single scroll bar between them.

- *Hierarchical browsing*   Tightly coupled windows can be used to support hierarchical browsing (Fig. 13.9). If one window contains the table of contents of a document, selection of a chapter title by a pointing device should lead to display, in an adjoining window, of the chapter contents (Fig. 13.10). Hierarchical browsing was nicely integrated into the Windows 95 Explorer to allow users to browse hierarchical directories, and is increasingly used in web-sites.

- *Direct selection*   Another tight-coupling idea is direct selection, in which pointing at an icon, a word in the text, or a variable name in a program pops up an adjoining window with the details of the icon, word definition, or the variable declaration (Fig. 13.11). Macintosh balloon help and Windows tool-tips are applications of direct selection, and users should be able to define such coordinations easily.

- *Two-dimensional browsing*   A two-dimensional cousin of hierarchical browsing shows an overview of a map, graphic, or photograph in one window, and the details in a second window. Users can move a field-

Figure 13.10

Multiple coordinated windows that create a convenient browsing environment. Clicking in the top window automatically fills the lower windows. (Chimera and Shneiderman, 1991.)
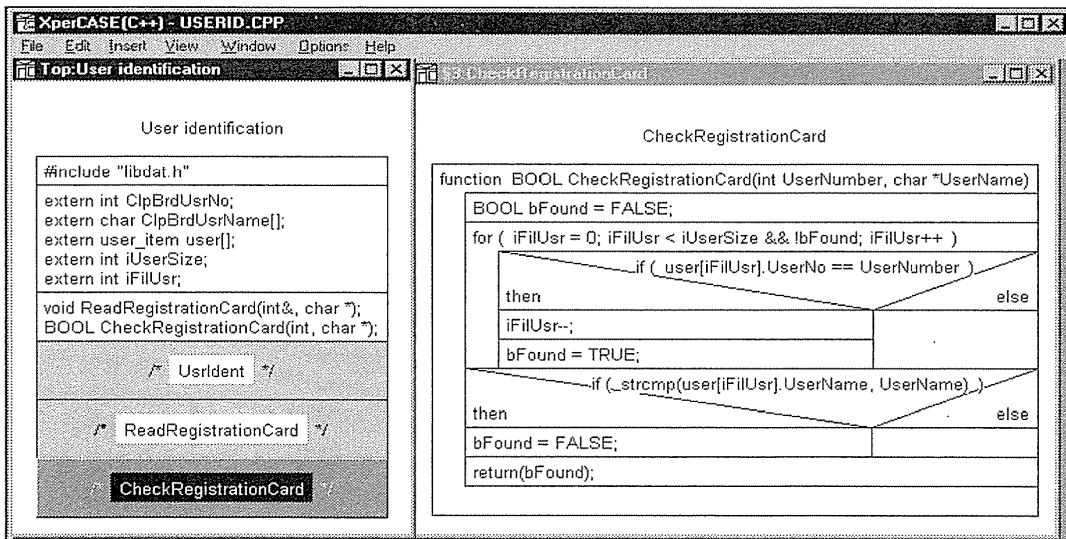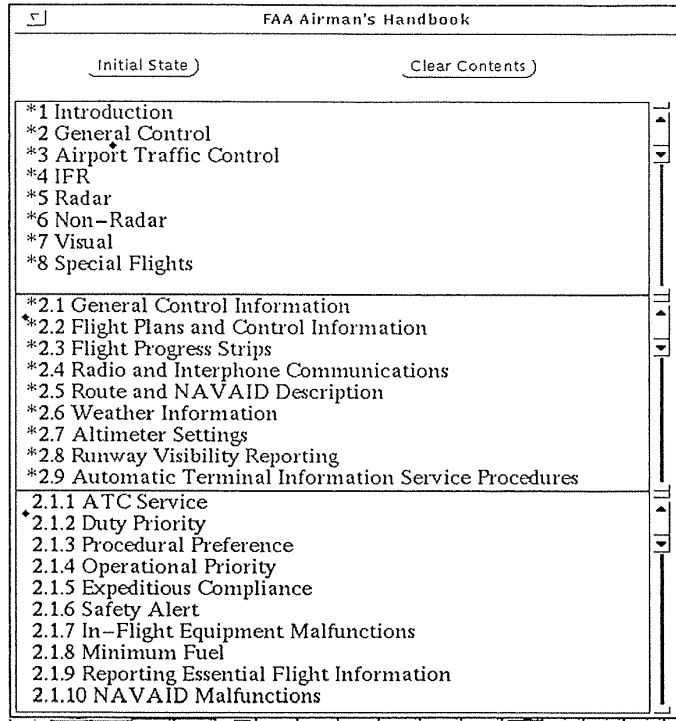

Figure 13.9

Hierarchical browsing in the XperCASE tool (now called EasyCASE with EasyCODE. The specification is on the left. As users click on components (CheckRegistrationCard), the detail view in a Nassi–Shneiderman Chart appears on the right. (Used with permission of Siemens AG Austria.)

```
Full thirty times hath Phoebus' cart gone round
Neptune's salt wash and Tellus' orbed ground,
And thirty dozen moons with borrowed sheen
About the world have time twelve thirties been,
Since love our hearts, and Hymen did our hands,
Unite communal in most sacred bands.
```

```
Full thirty times hatl Phoebus' cart gone round
Neptune's salt wash ( the chariot of  ground,
And thirty dozen moo the sun god     d sheen
About the world have            rties been,
Since love our hearts, and Hymen did our hands,
Unite communal in most sacred bands.
```
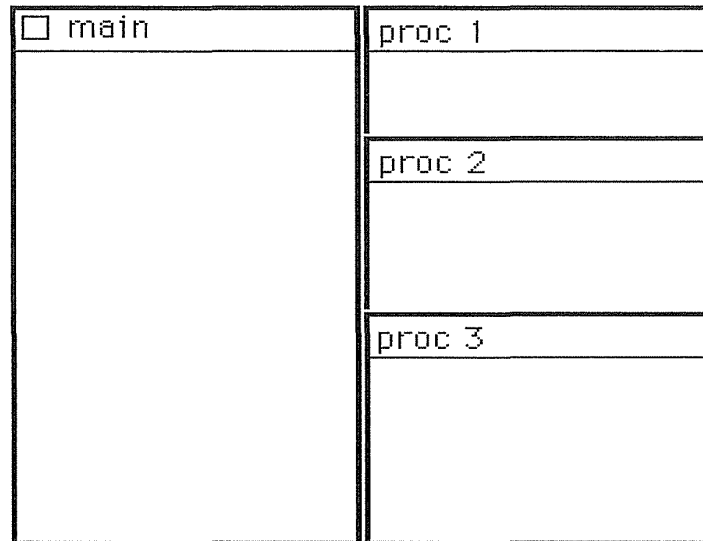
Figure 13.11

Direct selection of the phrase Phoebus' cart in this sample of Shakespearian text produces an immediate, in-place explanation. This technique is effective in showing data definitions for program variables.

of-view box in the overview to adjust the detail-view content (see Section 13.5).

- *Dependent-windows opening*   An option on opening a window might be to open dependent windows in a nearby and convenient location. For example, when users are browsing a program, when they open a main procedure, the dependent set of procedures could open up (Fig. 13.12).

- *Dependent-windows closing*   An option on closing a window would be to close all the dependent windows. This option might be applied to closing dialog, message, and help windows with a single action. For example, in filling in a form, users might have seen a dialog box with a choice of preferences. That dialog box might have led the user to activate a pop-up or error-message window, which in turn might have led to an invocation of the help window. After the user indicates the desired choice in the dialog box, it would be convenient for a double click on the dialog box close icon (or to select from a menu) to close all three or four windows (Fig. 13.13).

- *Save or open window state*   A natural extension of saving a document or a set of preferences is to save the current state of the display, with all the windows and their contents. This feature might be implemented by a simple addition of a Save screen as... menu item to the

Figure 13.12

Dependent windows. When such windows open, several other windows may open automatically. In this example the main procedure of a program has been opened, and the dependent procedures 1, 2, and 3 have been opened and placed at a convenient location. Connecting lines, shading, or decoration on the frame might indicate the parent and child relationships.

File menu of actions. This action would create a new icon representing the current state; it could be opened to reproduce that state. This feature is a simple version of the rooms approach (Henderson and Card, 1986).

13.5   Image Browsing by Tightly-Coupled Windows

When the user is browsing large images from medical, geographic information, or graphic-design systems, a pair of scroll bars is adequate when the image size is less than three to five times the screen size. With larger images, many designers have found that a tight coupling between an overview and a detail view provides many advantages over a single zoom-
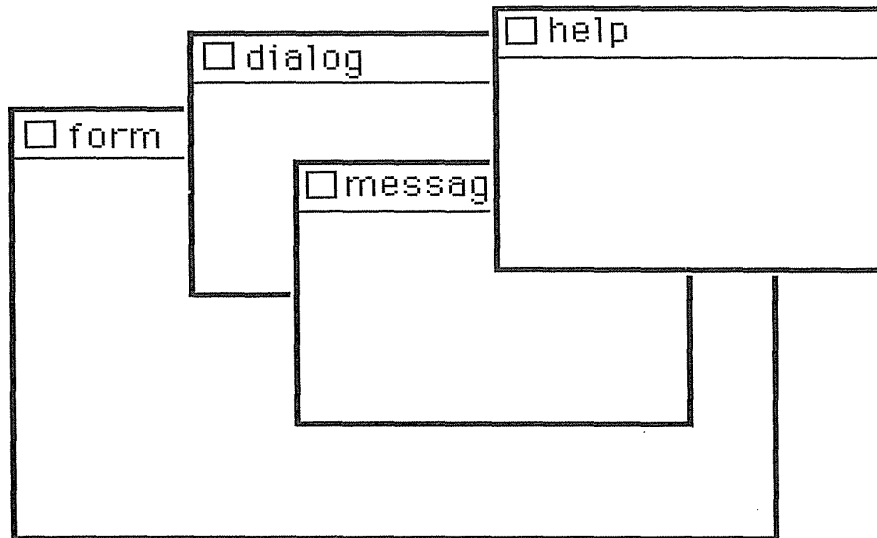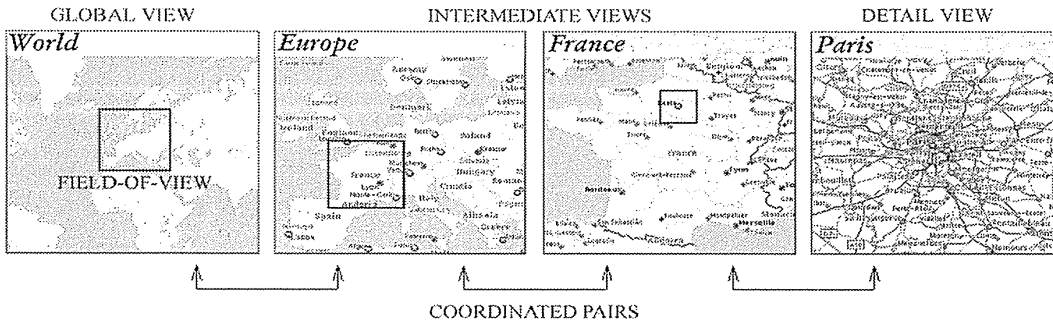
Figure 13.13

Dependent windows. When such windows close, other windows may close too. Here, all four windows will be closed automatically when the parent window, form, is closed. Lines, shading, or border decorations may indicate families of windows, with special marks to indicate parents and children.

ing or several independent windows. A field-of-view box on the overview can be moved to update the detail view. Similarly, if users pan in the detail view, the field-of-view box should move in the overview. Well-designed tightly coupled windows have matching aspect ratios in the field-of-view box and the detail view, and changes to the shape of either produces a corresponding change in the other.

The magnification from the overview to the detail view is called the *zoom factor*. When the zoom factors are between 5 and 30, the tightly coupled overview and detail view pair are effective; for larger zoom factors, however, an additional intermediate view is needed. For example, if an overview shows a map of the France, then a detail view showing the Paris region is effective. However, if the overview were of the entire world, then intermediate views of Europe and France would preserve orientation (Fig. 13.14).

Side-by-side placement of overview and detail views is the most common layout, since it allows users to see the big picture and the details at the same time. Some systems provide a single view, either smooth zooming to move in on a selected point (Bederson and Hollan, 1994), or simply replacing the

GLOBAL VIEW   INTERMEDIATE VIEWS   DETAIL VIEW

COORDINATED PAIRS

Figure 13.14

Global and intermediate views, which provide overviews for the detail view of Paris. Movements of the field-of-view boxes change the content in the detail view. (Plaisant et al., 1994.)

overview with the detail view. This zoom-and-replace approach is simple to implement and gives the maximal screen space for each view, but it denies the users the chance to see the overview and detail view at the same time. A variation is to have the detail view overlap the overview, even though it may obscure key items.

Attempts to provide detail views (focus) and overviews (context) without obscuring anything have motivated interest in *fisheye views* (Sarkar and Brown 1994; Bartram et al., 1995) (Fig. 13.15). The focus area (or areas) is magnified to show detail, while preserving the context, all in a single display. This approach is visually appealing, but the changing distortion may be disorienting, and the zoom factor in published examples never exceeds 5.

The design for image browsers should be governed by the users' tasks, which can be classified as follows (Plaisant et al., 1995):

- *Image generation*   Paint or construct a large image or diagram.
- *Open-ended exploration*   Browse to gain an understanding of the map or image.
- *Diagnostic*   Scan for flaws in an entire circuit diagram, medical image or newspaper layout.
- *Navigation*   Have knowledge of overview, but need to pursue details along a highway or vein.
- *Monitoring*   Watch the overview and, when problem occurs, zoom in on details.
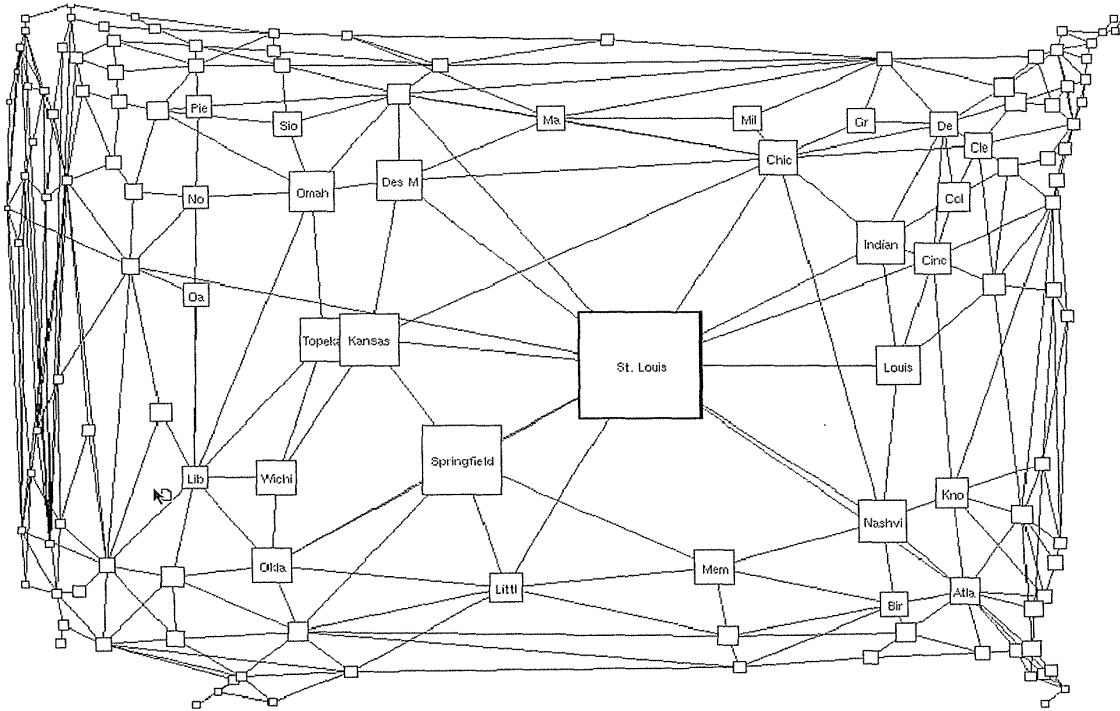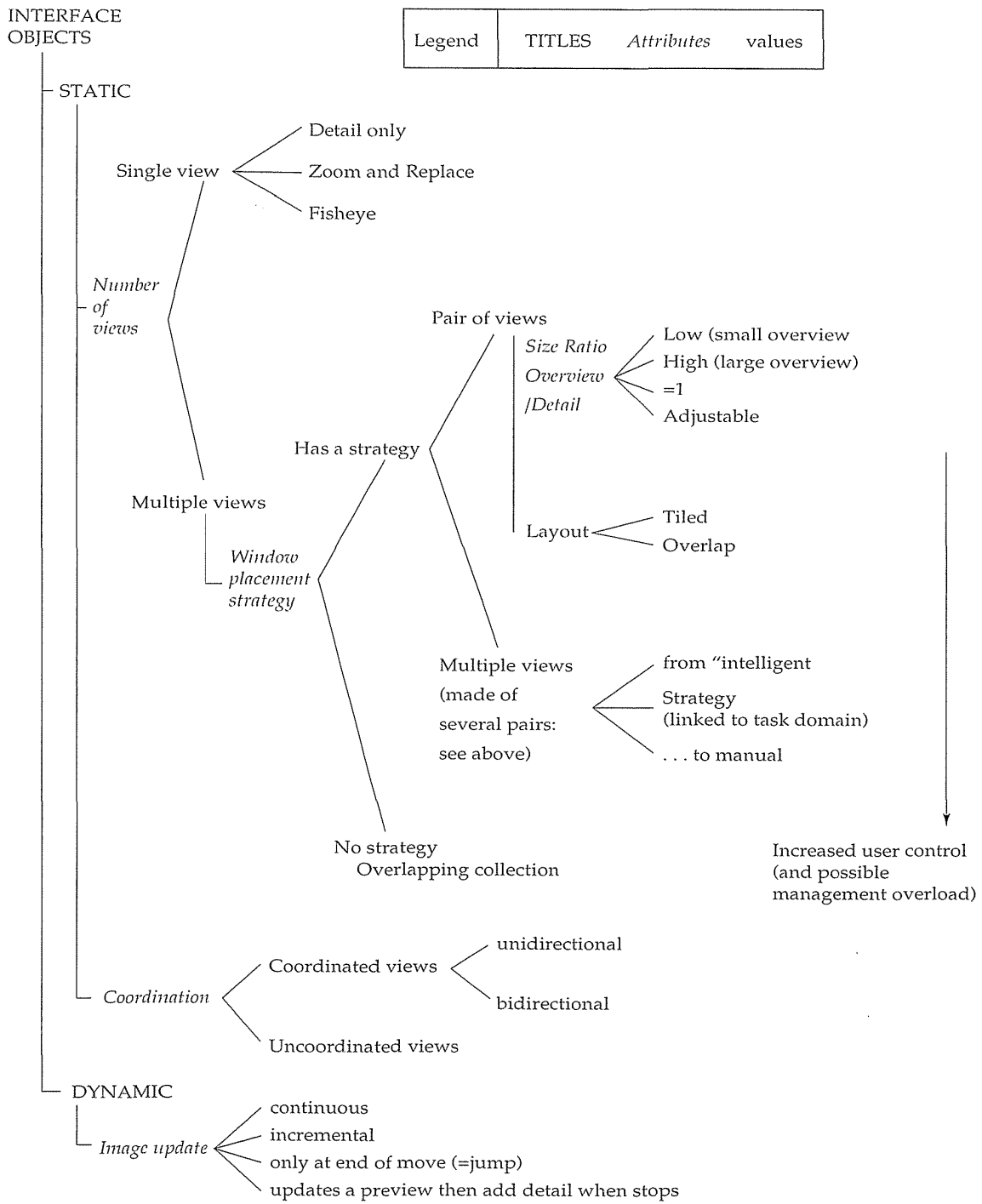
Figure 13.15

Fisheye view of U.S. cities, with the focus on St. Louis. The context is preserved, although the distortions can be disorienting. (Sarkar and Brown, 1994.) (Used with permission of Marc Brown, DEC Systems Research Center, Palo Alto, CA.)

Within these high-level tasks, users carry out many low-level actions, such as moving the focus (jumping from city to city on a map), comparing (viewing two harbors at the same time to compare their facilities, or viewing matching regions in X-ray images of left and right lungs), traversing (following a vein to look for blockages), or marking locations to return to them at a later time.

A taxonomy of browser-interface objects (Fig. 13.16) and actions (Fig. 13.17) reveals the rich possibilities that will be sorted out in the coming years as certain strategies gain commercial dominance. Tight couplings between windows and other interface widgets—especially sliders and data-entry fields—are proving to be helpful in many tasks (see Chapter 15).

INTERFACE
OBJECTS

Legend	TITLES	*Attributes*	values

- STATIC

Single view
- Detail only
- Zoom and Replace
- Fisheye

Number of views

Multiple views

Window placement strategy

Has a strategy

Pair of views

Size Ratio Overview /Detail
- Low (small overview)
- High (large overview)
- =1
- Adjustable

Layout
- Tiled
- Overlap

Multiple views
(made of
several pairs:
see above)

Strategy
- from "intelligent
- Strategy (linked to task domain)
- . . . to manual

No strategy
Overlapping collection

Increased user control
(and possible
management overload)

Coordination

Coordinated views
- unidirectional
- bidirectional

Uncoordinated views

- DYNAMIC

Image update
- continuous
- incremental
- only at end of move (=jump)
- updates a preview then add detail when stops

Figure 13.16

Taxonomy of browser strategies, focusing on interface objects. (Plaisant et al., 1994.)

INTERFACE
ACTIONS

Legend	TITLES	*Attributes*	values

MANUAL

Zoom
- None (detail only)
- Yes
  - *Zoom location spec*
    - at cursor
    - field of view drawn from
      - one corner to diagonal
      - center to corner
  - *Zoom factor*
    - fixed (steps)
    - specified
      - by value
      - implicitly by size of field of view
  - *Destination window*
    - fixed
    - user's selection
  - *Zoom out*
    - None
    - By undoing (implicit zoom-out)
    - *Explicit zoom out*
      - Step
      - Specified

Pan
- none (zoom out then in required)
- scroll
  - scrollbars on detail view
  - by moving the field of view on a synchronized overview
- sticky hand
- arrow key

AUTOMATED
- Synchronization of coordinated views
- SAVE POINTS
  - Jump to point by
    - Selection from a list
    - Click on icons placed on:
      - overview
      - detail view
        (= short cuts)
- AUTOMATIC EXPLORATION
  - Following of specified trail
    Interruption and sidetracting (y/n)
  - Scan of a specified area
  - Scan of complete image
- SEARCH
  - *Text* (y/n)
  - Graphic (y/n)
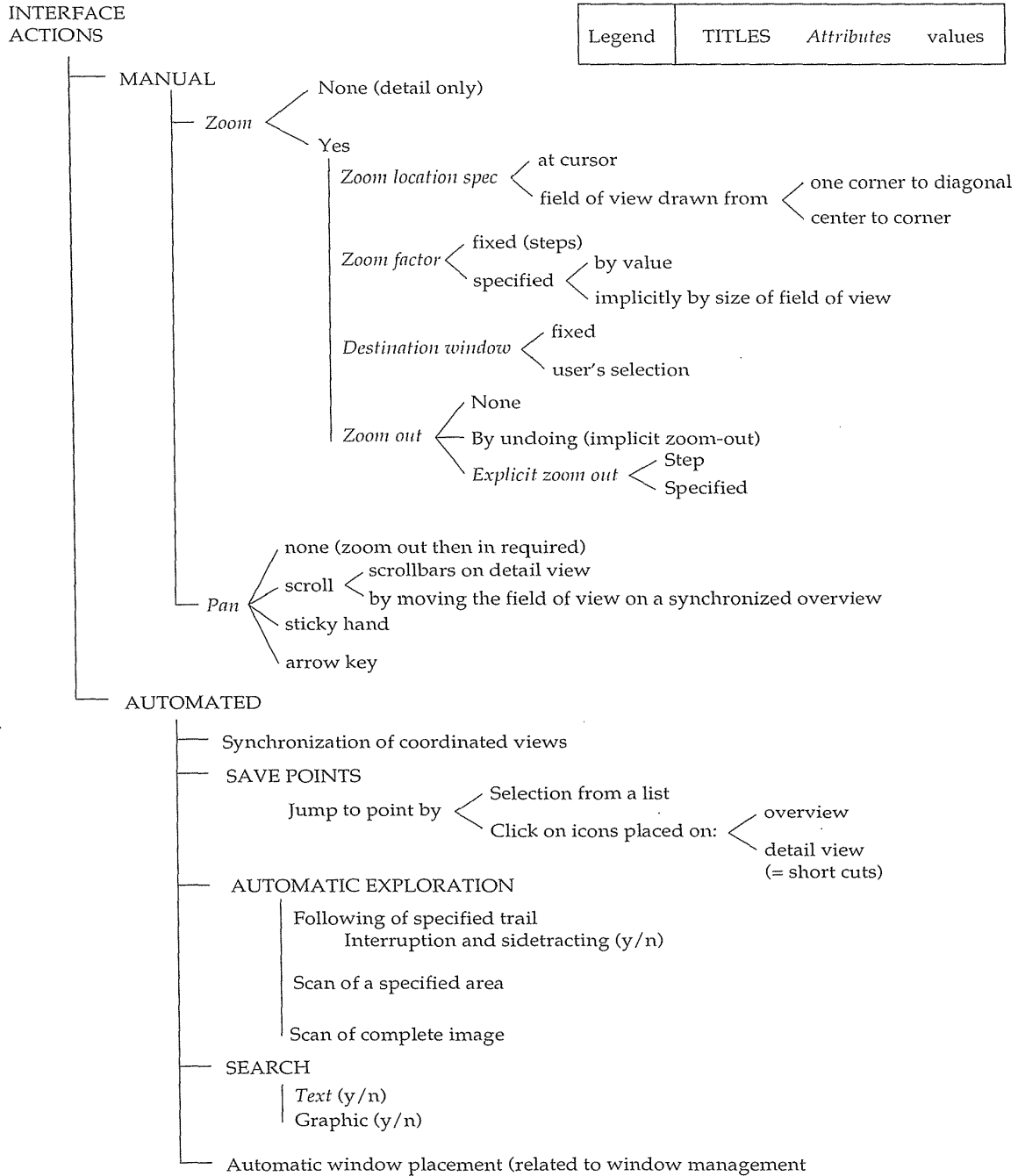- Automatic window placement (related to window management

Figure 13.17

Taxonomy of browser strategies, focusing on interface actions. (Plaisant et al., 1994.)

13.6   Personal Role Management and Elastic Windows

Window coordination by tight coupling will facilitate handling of larger images and tasks that have been too complex to deal with in the past. However, there are even more potent opportunities to improve window management radically. The current GUIs offer a desktop with applications represented as icons and documents organized into folders.

Novel approaches emphasize a *docucentric* design (Microsoft's Object Linking and Embedding or Apple's OpenDoc Architecture), in which documents become more important and applications fade into the background. The enriched documents contain multiple object types, such as text, drawings, photos, spreadsheets, and sound, with links across documents to share common objects. Actions that earlier had required opening a new window for an application—such as spell checking, thesaurus referencing, or document faxing—are now integrated into the unified docucentric interface.

Although these are useful steps away from the underlying technology and more in harmony with the users' perceptions of their work, larger steps are needed to reach the next generation of user interfaces. A natural progression is toward a *role-centered* design, which emphasizes the users' tasks rather than the documents. This design concept is in harmony with the current movement toward computer-supported cooperative work and groupware (Chapter 14). These tools are aimed at coordination of several people performing a common task with a common schedule. By contrast, a role-centered design could substantially improve support for individuals in managing their multiple roles in an organization. Each role brings these individuals in contact with different people for carrying out a hierarchy of tasks following an independent schedule. A *personal role manager* (PRM), instead of a window manager, could improve performance and reduce distraction while the user is working in a given role, and could facilitate shifting of attention from one role to another (Shneiderman and Plaisant, 1994; Plaisant and Shneiderman, 1995).

In a personal role manager, each role has a vision statement (a document that describes responsibilities, quotas, and goals) that is established by the user or manager. The explicitness of the vision can simplify the training and integration of new personnel into the organization, and also can facilitate the temporary covering of responsibilities among employees (during vacations or parental leave, for example).

For example, a professor may have roles such as a teacher of courses, advisor to graduate students, member of the recruiting committee, principal investigator of grants, author of technical reports, and liaison to industry. In the teacher role, the professor's vision statement might include the intention to apply electronic mail to facilitate a large undergraduate course. Files

might include homework assignments, bibliography, course outline, and so on. The task hierarchy might begin with tasks such as choosing a textbook and end with administering the final exam. The subtasks for administering the final exam might include preparing the exam, photocopying the exam, reserving a room, proctoring the exam, and grading the exam. The set of people includes the students, teaching assistants, bookstore manager, registrar, and colleagues teaching other sections of the course. The schedule would begin with deadlines for submitting the book order to the bookstore manager and end with turning in the final grades to the registrar.

The personal role manager was stimulated by experiences in managing multiple projects with many participants, plus observations and interviews with 15 experienced users to understand their needs. Although there are scheduling, time management, address book, document-management packages available, the coordination of these tools is often underemphasized. The personal role manager would simplify and accelerate the performance of common coordination tasks, in the same way that GUIs simplify file-management tasks.

The key to PRM is organizing information according to the roles that an individual has in an organization. When users are working in a role, they have most relevant information visually available. These visual cues remind them of their goals, related individuals, required tasks, and scheduled events. The initial layout of roles may be established by a manager for a new employee, but then the employee can adjust, combine, or split roles as the demands change.

Screen management is one of the key functions of the personal role manager. All roles should be visible, but the current focus of attention could occupy most of the screen. As users shift attention to a second role, the current one shrinks and the second one grows to fill the screen. Users could simultaneously enlarge two roles if there were interactions between them. The task objects in a personal role manager are these:

- *Vision statement* Each role has a vision statement that reminds users of their goals. As a professor, my teaching role might have a vision statement about my desire to "increase class participation by collaborative methods, improve teamwork on term projects by requiring regular management meetings, prepare careful notes to facilitate future teaching of the same course, and coordinate with my teaching assistants by weekly meetings and electronic-mail discussions."

- *Set of people* When acting in a given role, users interact with a set of people that is a subset of the people in an organization's telephone book. Making the role-relevant group of people continuously visible (for example, with names or small photos on the border of the large

screen) has at least two benefits. First, the images will act as cues to remind the user of the need to inform, makes request of, or communicate with that individual (similar to seeing someone in the hallway, and thus triggering communication to coordinate work). Second, the images act as active menus to initiate telephone, fax, or electronic-mail communication. For example, a document can be dragged and dropped onto an image, triggering electronic mail plus a log of the action. Providing direct access to these people without the need of a directory search speeds performance and reduces cognitive load.

- *Task hierarchy*   Tasks are hierarchically organized into subtasks via an outlining tool. The professor role may have a task for each of several courses, or the principal-investigator role may have tasks for multiple grants. Each course has multiple subtasks, such as writing the syllabus, ordering textbooks, giving exams, and preparing final grades. The task hierarchy acts as a to-do list, and is linked to the schedule calendar to remind users of upcoming deadlines.

- *Schedule*   Each role has an associated schedule that is a component of a user's master schedule. When viewing a role, the user initially sees only the role-related schedule. For example, when the user is viewing the professor role, the semester schedule is visible; when the principal-investigator role is on view, the 2-year grant schedule is visible. Schedules can be combined to reveal a master schedule that allows users to allocate time and to ensure that travel, vacations, and required meetings are blocked off on every schedule.

The requirements for a personal role manager include these:

- Support a *unified framework* for information organization according to users' roles.
- Provide a *visual, spatial layout* that matches tasks.
- Support *multiwindow actions* for fast arrangement of information.
- Support *information access* with partial knowledge of an information item's nominal, spatial, temporal, and visual attributes, and relationships to other pieces of information.
- Allow *fast switching* and *resumption* among *roles*.
- Free users' *cognitive resources* to work on *task-domain actions*, rather than making users concentrate on interface-domain actions.
- Use *screen space* efficiently and productively for tasks.

The Rooms model for multiple-window workspaces provides improved support for these requirements, but does not provide sufficiently smooth transi-

tion among roles. *Elastic windows* organize windows in a space-filling, hierarchically nested, tiled layout that was designed to support the personal role manager's requirements (Kandogan and Shneiderman, 1997). Actions, such as an `open` or `close`, can be applied to a group of windows to open six related electronic-mail messages or 10 resumes of job applicants. In a study with 12 sophisticated users, multiwindow operations were shown to enable faster task switching and structuring of the work environment, by opening, closing, or changing the size of 10 to 20 windows at a time.

Figure 13.18 depicts an example mapping of different roles of a student onto a hierarchical window organization. This student takes two courses this semester: Software Engineering and Computer Networks. Project materials and partners; homework assignments; and correspondence with the professor, TAs, and classmates for each course are organized in a hierarchical fashion. This student has a number of other roles, in which he manages home duties, job responsibilities, and the planning of a birthday party. Partners,
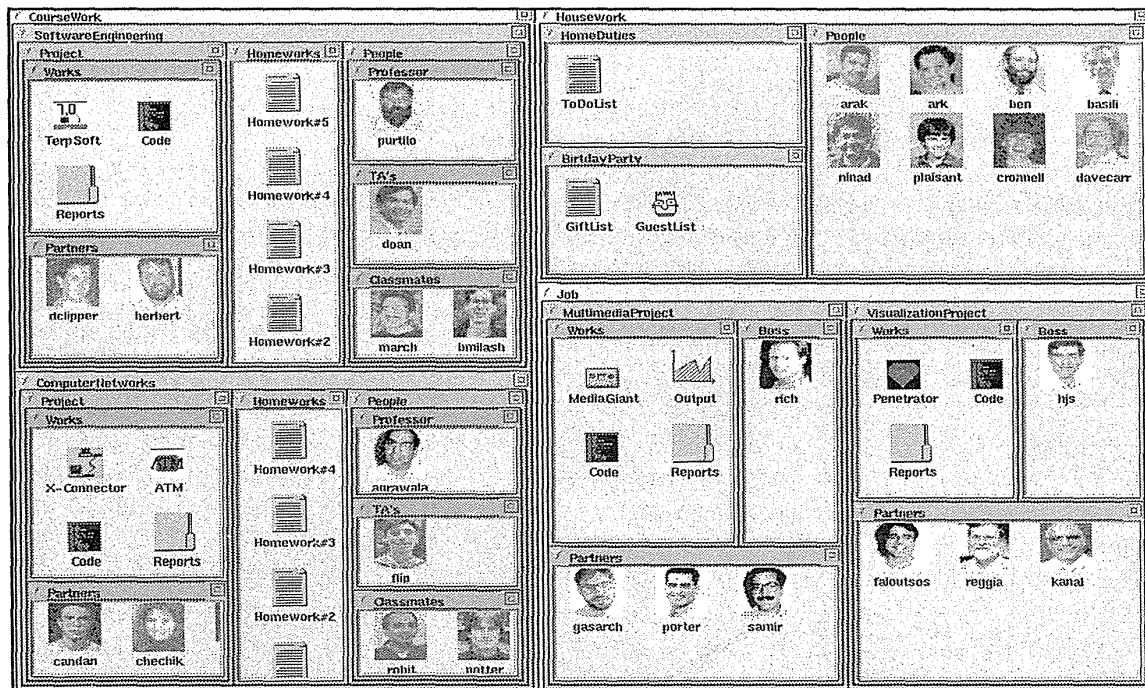

Figure 13.18

Role-manager prototype for a student doing coursework (in Software Engineering and Computer Networks), housework, and a job (with a Multimedia Project and a Visualization Project). Users can resize, open, close, or move groups of windows. (Kandogan and Shneiderman, 1997.)

schedules, tools, and documents pertaining to each of these roles are mapped hierarchically into different windows. The interface layout provides an overview of the roles, enables direct access on demand to details of any role, and can be custom-tailored for a specific task.

In elastic windows, users can change the layout according to tasks quickly, by applying operations on groups of windows (for example, *container* windows that surround their *member* windows). Elastic-window resizing, which the system does by enlarging a window or a group of windows while reducing the size of other windows in the group, enables users to focus easily on information in different windows. Also, a subhierarchy of windows can be collapsed into a single visual primitive (for example, icon or textual item). Then, for example, all windows containing the code for the software-engineering class project can be opened with a single action, thus enabling fast switching among roles.

13.7   Practitioner's Summary

Multiple windows are an accepted feature of contemporary computers. Window-frame designs are distinctive, and graphic-design style is important, as are features such as title bars, action menus, and scroll bars. Users expect to perform rapidly, conveniently, and comprehensibly typical actions, such as opening, closing, moving, and sizing. The popular style of overlapping windows has an appealing three-dimensional (or two-and-one-half–dimensional) look and serves some purpose, but it can produce clutter and be inefficient. Approaches that limit the overlap can be helpful in many situations. Coordination strategies based on user tasks can bring further benefits by automating multiple-window actions, bringing windows close to where they are needed, reducing the housekeeping burden, and avoiding unwanted overlaps. Scrolls bars are applied widely and are usually successful, but there is room for improvement. Browsing large images with tightly-coupled windows simplifies and speeds the users' task completion.

13.8   Researcher's Agenda

Windows provide visually appealing possibilities and intriguing opportunities for designers, but advantages and disadvantages of design features are still poorly understood. Eye-motion studies might provide data for effective design, and basic task analysis can still be productive in classifying

the information needs and action sequences of users. A theory of window layouts, based on psychological principles and task analysis, would be a significant contribution. Even without a deeper theory, there seems to be much room for innovation in tight coupling of multiple-window layouts. As higher-resolution and larger displays appear, dense presentations and novel three-dimensional layouts may be possible, with dramatic animations and eye-catching designs. The addition of coordinated windows and window macros is possible, and may lead to a new generation of GUIs that support personal role management.

World Wide Web Resources   WWW

Window management is changing because of the World Wide Web, where new ideas such as frames can be tried, debated, and studied. Novel window management strategies are offered by some developers in demos and commercial products.

http://www.aw.com/DTUI

References

Apple Human Interface Guidelines: The Apple Desktop Interface, Addison-Wesley, Reading, MA (1987).

Bartram, Lyn, Ho, Albert, Dill, John, and Henigman, Frank, The continuous zoom: A constrained fisheye technique for viewing and navigating large information spaces, *Proc. User Interface Software and Technology '95,* ACM, New York (1995), 207–215.

Bederson, B., Hollan, J. D., Pad++: A zooming graphical interface for exploring alternate interface physics, *Proc. of the UIST '94, User Interface Software and Technology,* ACM, New York (1994), 17–26.

Billingsley, Patricia A., Taking panes: Issues in the design of windowing systems. In Helander, M. (Editor), *Handbook of Human–Computer Interaction,* Elsevier Science Publishers B.V., Amsterdam, The Netherlands (1988), 413–436.

Bly, Sara and Rosenberg, Jarrett, A comparison of tiled and overlapping windows, *Proc. CHI '86 Conference: Human Factors in Computing Systems,* ACM, New York (1986), 101–106.

Bury, K. F., Boyle, J. M., Evey, R. J., and Neal, A. S., Windowing versus scrolling on a visual display terminal, *Human Factors,* 24, 4 (1982), 385–394.

Bury, Kevin F., Davies, Susan E., and Darnell, Michael J., Window management: A review of issues and some results from user testing, IBM Human Factors Center Report HFC-53, San Jose, CA (June 1985).

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., Pavel, M., and Farrell, J. E., Window-based computer dialogues, *INTERACT '84, First IFIP Conference on Human–Computer Interaction*, London (1984), 239–243.

Card, Stuart K. and Henderson, Austin, A multiple virtual-workspace interface to support task switching, *Proc. CHI '87 Conference: Human Factors in Computing Systems*, ACM, New York (1987), 53–59.

Chimera, Richard, Value bars: An information visualization and navigation tool for multiattribute listings, *Proc. CHI '92 Conference: Human Factors in Computing Systems*, ACM, New York (1992), 293–294.

Engelbart, Douglas C. and English, William K., A research center for augmenting human intellect. In Greif, I. (Editor), *Computer-Supported Cooperative Work: A Book of Readings*, Morgan Kaufmann, Palo Alto, CA (1988), 81–105.

Gait, Jason, An aspect of aesthetics in human–computer communications: Pretty windows, *IEEE Transactions on Software Engineering*, SE-11, 8 (August 1985), 714–717.

Henderson, Austin and Card, and Stuart K., Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface, *ACM Transactions on Graphics*, 5, 3 (1986), 211–243.

Hopgood, F. R. A., Duce D. A., Fielding, E. V. C., Robinson, K., and Williams, A. S. (Editors), *Methodology of Window Management*, Springer-Verlag, Berlin (April 1985).

Johnson, Jeff, Roberts, Teresa L., Verplank, William, Smith, David C., Irby, Charles H., Beard, Marian, and Mackey, Kevin, The Xerox Star: A Retrospective, *IEEE Computer*, 22, 9 (September 1989), 11–29.

Kandogan, Eser and Shneiderman, Ben, Elastic windows: Evaluation of multi-window operations, *Proc. CHI '97 Conference: Human Factors in Computing Systems*, ACM, New York (1997), 250–257.

Kobara, Shiz, *Visual Design with OSF/Motif*, Addison-Wesley, Reading, MA (1991).

Marcus, Aaron, *Graphic Design for Electronic Documents and User Interfaces*, ACM Press, New York (1992).

Myers, Brad, Window interfaces: A taxonomy of window manager user interfaces, *IEEE Computer Graphics and Applications*, 8, 5 (September 1988), 65–84.

Myers, Brad, *All the Widgets, SIGGRAPH Video Review #57*, ACM, New York (1990).

Norman, Kent L., Weldon, Linda J., and Shneiderman, Ben, Cognitive layouts of windows and multiple screens for user interfaces, *International Journal of Man–Machine Studies*, 25, (1986), 229–248.

Plaisant, Catherine, Carr, David, and Shneiderman, Ben, Image browsers: Taxonomy and design guidelines, *IEEE Software*, 12, 2 (March 1995), 21–32.

Plaisant, Catherine and Shneiderman, Ben, Organization overviews and role management: Inspiration for future desktop environments, *Proc. IEEE Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Press, Los Alamitos, CA (April 1995), 14–22.

Sarkar, Manojit and Brown, Marc H., Graphical fisheye views, *Communications of the ACM*, 37, 12 (July 1994), 73–84.

Schmandt, Chris, Ackerman, Mark S., and Hindus, Debby, Augmenting a window system with speech input, *IEEE Computer*, 23, 8 (August 1990), 50–56.

Seabrook, Richard and Shneiderman, Ben, The user interface in a hypertext, multi-window browser, *Interacting with Computers*, 1, 3 (1989), 299–337.

Shneiderman, Ben and Plaisant, Catherine, The future of graphic user interfaces: Personal role managers, *People and Computers IX*, Cambridge University Press, Cambridge, U.K. (1994), 3–8.

Shneiderman, Ben, Shafer, Phil, Simon, Roland, and Weldon, Linda, Display strategies for program browsing: Concepts and experiment, *IEEE Software*, 3, 3 (May 1986), 7–15.

Smith, D. C., Irby, C., Kimball, R., and Verplank, W. L., Designing the Star user interface, *Byte*, 7, 4 (April 1982), 242–282.

Tesler, Larry, The Smalltalk Environment, *Byte*, 6, (August 1981), 90–147.

Tombaugh, J., Lickorish, A., and Wright P., Multi-window displays for readers of lengthy texts, *International Journal of Man–Machine Studies*, 26, (1987), 597–615.
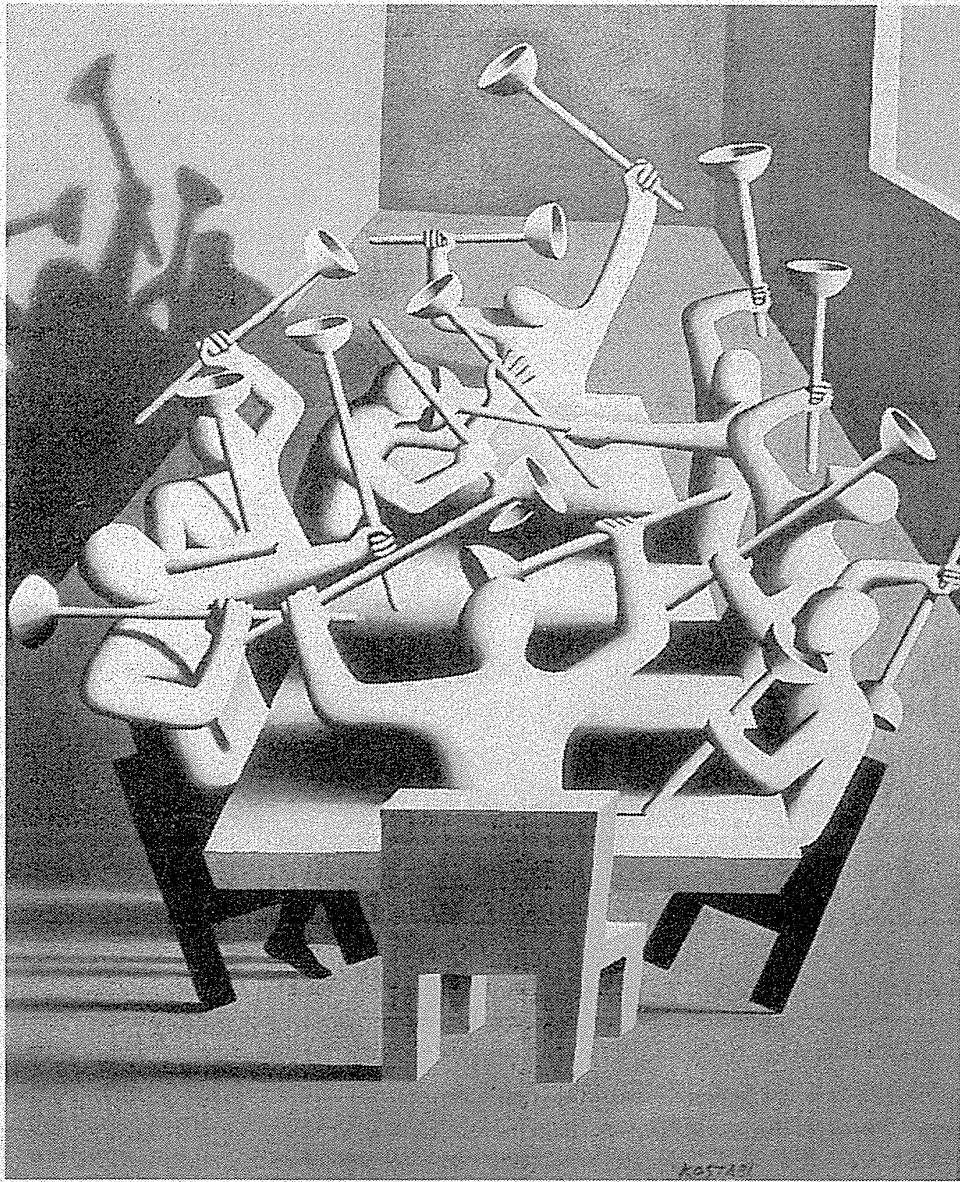
Mark Kostabi, *Upheaval*, 1982

14

Computer-Supported Cooperative Work

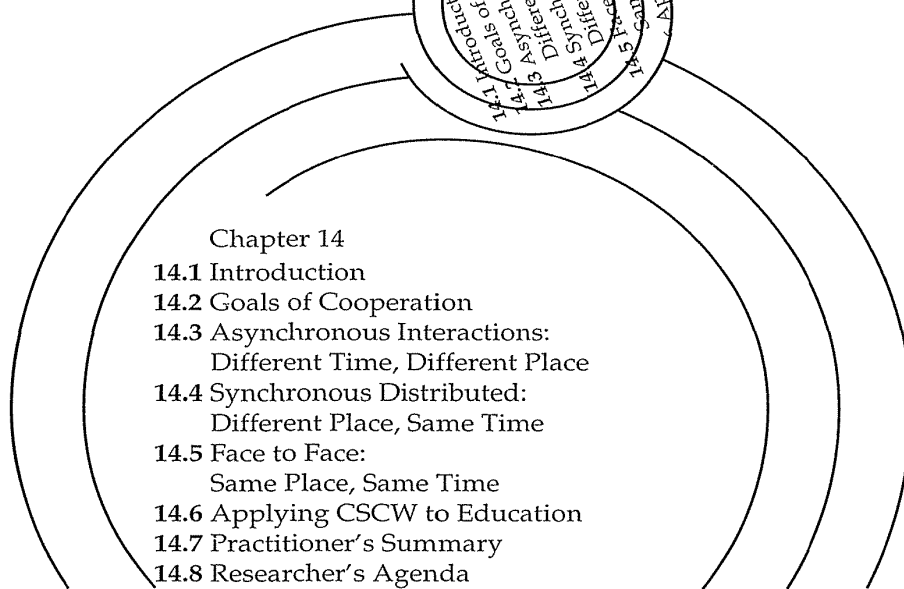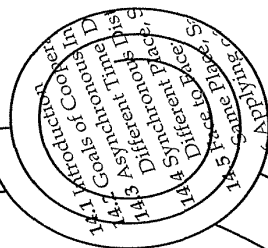Three helping one another will do as much as six working singly.

Spanish proverb

CH 14   CH 15   CH 16

Chapter 14
14.1 Introduction
14.2 Goals of Cooperation
14.3 Asynchronous Interactions:
     Different Time, Different Place
14.4 Synchronous Distributed:
     Different Place, Same Time
14.5 Face to Face:
     Same Place, Same Time
14.6 Applying CSCW to Education
14.7 Practitioner's Summary
14.8 Researcher's Agenda

14.1 Introduction

The introversion and isolation of early computer users has given way to
lively online communities of busily interacting dyads and bustling crowds of
chatty users. The pursuit of human connections has prompted millions of
users to join mailing lists, visit chat rooms, and fill newsgroups with useful
information and helpful responses, peppered with outrageous humor. But,
as in any human community, there is also controversy, slander, and pornog-
raphy. The World Wide Web (Chapter 16) has dramatically expanded the
communications richness, with colorful graphics and sometimes too-daz-
zling Java animations. The Web is sometimes derided as a playground, but
serious work and creative endeavors are enormously facilitated by the easy
flow of information.

Goal-directed people quickly recognized the benefits of electronic cooperation and the potential to live in the immediacy of the networked global village. The distance to colleagues is measured not in miles, but rather in intellectual compatibility and responsiveness; a close friend is someone who responds from 3000 miles away within three minutes at three A.M. with the reference that you need to finish a paper.

The good news is that computing, once seen as alienating and antihuman, is becoming a socially respectable and interpersonally positive force. Enthusiasts hail cooperative technologies, groupware, team processes, coordination science, and other communal utopias, but there may be a dark side to the force. Even optical fiber is not a channel through which you can send a handshake or a hug. How does intimacy survive when mediated by the remoteness in time and physical space? Can laughter and tears mean the same thing for electronic-dialog partners as for face-to-face partners? Will the speedup in work improve or reduce quality? Can cooperative systems be turned into oppressive tools or confrontive environments?

New terminology and metaphors are appearing daily. Although the conferences on *computer-supported cooperative work* have established *CSCW* as a new acronym, even the organizers debate whether that acronym covers *cooperative, collaborative,* and *competitive* work. The focus of CSCW researchers is on the design and evaluation of new technologies to support the social processes of work, often among distant partners. The implementers and marketeers quickly gravitated to *groupware* as a term to describe the team-oriented commercial products (Baecker, 1993). For researchers, new paradigms and fresh ideas are flowing from psychologists, sociologists, and anthropologists (Vaske and Grantham, 1990; Sproull and Kiesler, 1991). For educators, the movement toward social construction theories of learning is fostered by the World Wide Web and by classroom tools and techniques (Hiltz, 1994; Harasim et al., 1995; Shneiderman et al., 1995).

Networked communities have become talk-show topics, with social commentators celebrating or warning about the transformational power of CSCW. Howard Rheingold's (1993) popular book on *Virtual Communities* tells charming and touching stories of cooperation and support in the San Francisco–based WELL. At the same time, clinical psychologists analyze network addictions and deconstruct manufactured cyber-identities (Turkle, 1995).

14.2 Goals of Cooperation

People cooperate because doing so is satisfying or productive. Communication can have purely emotionally rewarding purposes or specific task-related goals. Communication can be sought individually or imposed managerially.

Relationships can be one-time encounters or enduring. The analysis of cooperative systems is governed by the goals and tasks of the participants:

- *Focused partnerships* are cooperations between two users who need each other to complete a task, such as joint authors of a technical report, two pathologists consulting about a cancer specimen, programmers debugging a program together, or astronaut and ground-controller repairing a faulty satellite. Often, there is an electronic document or image to "conference over." Partners can use electronic mail, voice mail, telephone, or video mail.

- *Lecture or demo* formats have one person sharing information with many users at remote sites. The start time and duration is the same for all; questions may be asked by the recipients. No history keeping is required, but a replay may be possible at a later time.

- *Conferences* allow groups to communicate at the same time or spread out over time, but with participants distributed in space. Many-to-many messaging may be used, with a record of previous conversations: a scientific-meeting program committee might discuss the plans for an upcoming event, or a group of students might discuss the most recent class examination. In more directed conferences, a leader or moderator supervises the online discussion to achieve goals within deadlines.

- *Structured work processes* let people with distinct roles cooperate on some task: a scientific-journal editor arranges online submission, reviewing, revisions, and publication; a health-insurance agency receives, reviews, and reimburses or rejects medical bills; or a university admissions committee registers, reviews, chooses, and informs high-school applicants.

- *Electronic commerce* includes short-term collaborations to inquire about and then order a standard product, and long-term negotiations to craft a major business deal or contract. Electronic negotiations can be distributed in time and space, while producing an accurate record and rapid dissemination of results.

- *Meeting and decision support* can be done in a face-to-face meeting, with each user working at a computer and making simultaneous contributions. Shared and private windows plus large-screen projectors, enable simultaneous shared comments that may be anonymous. Anonymity not only encourages shy participants to speak up, but also allows forceful leaders to accept novel suggestions without ego conflicts.

- *Teledemocracy* allows city, state, or national governments to conduct online town-hall meetings; to expose officials to comments from constituents; or to produce consensus through online conferences, debates, and votes.

There are undoubtedly other cooperative tasks; this list simply indicates the diversity. Within each task, there are numerous variations, and the poten-

tial market for innovative software products is large. However, designing for cooperation is a challenge because of the numerous and subtle questions of etiquette, dominance, ego, anxiety, and posturing. The tasks in our list are serious and professional, but there are also opportunities for entertaining multiperson games, challenging contests, or playful social encounters.

The traditional way (Ellis et al., 1991) to decompose cooperative systems is by a time–space matrix:

	Same Time	*Different Times*
Same place	face to face (classrooms, meeting rooms)	asynchronous interaction (project scheduling, coodination tools)
Different places	synchronous distributed (shared editors, video windows)	asynchronous distributed (email, listservs, conferences)

This decomposition focuses on two critical dimensions, and thus guides designers and evaluators.

Research in cooperative systems is more difficult than is that in single-user interfaces. The multiplicity of users makes it nearly impossible to conduct controlled experiments, and the flood of data from multiple users defies orderly analysis. Small-group psychology, industrial and organization behavior, sociology, and anthropology provide useful research paradigms, but many researchers must invent their own methodologies. Subjective reports, case studies, and users' eagerness to continue using the groupware tools are the strongest indicators of success (Kraut et al., 1994).

Cooperative systems are maturing, but the determinants of success are still not clear. Electronic mail is a widespread success story, and videoconferencing use grows slowly but steadily, while shared calendar programs are repeatedly spurned. Grudin (1994) outlines some of the causes of groupware failures: disparity between who does the work and who gets the benefit, threats to existing political power structures, insufficient critical mass of users who have convenient access, violation of social taboos, and rigidity that counters common practice or prevents exception handling.

Arguments over measures of success also complicate analysis. Whereas some people cite the high utilization of electronic mail, others question whether electronic mail aids or hinders job-related productivity. Videoconferencing may initially reduce travel expenses, but it can encourage cooperation with more distant partners, thus leading to increasing costs and possibly more travel.

In educational environments, outcomes can be measured by comparison of scores on final exams, but students are often learning new skills when they work collaboratively in networked environments. These skills are needed in the workplace, where teamwork and effective communication are essential.

Cooperation and discussion are a natural part of democratic processes, so online campaigning, organizing, and consensus building are likely to become required skills for politicians. Electronic mail to public officials and online town-hall meetings are already possible, but electronic parliaments with consensus building, committee caucusing, deal making, and voting are still to emerge. Utopian visionaries suggest increased and constructive participation in democratic processes, but other people warn about the dangers of uninformed citizens influencing legislation and of harmful speedup that reduces thoughtful deliberation.

Community networks are already successful and are spreading (Rheingold, 1993; Schuler, 1996). Some communities are geographically confined, such as the ones in Seattle, Washington, Taos, New Mexico, and Blacksburg, Virginia (Carroll and Rosson, 1996), whereas others have a global perspective but are topically focused, such as the ones for AIDS patients, archaeologists, and agronomists. The positive side is the facilitation of communication among like-minded people who have shared interests; the negative side is that electronic communities may have less commitment than do those that attend to face-to-face meetings of clubs, civic groups, and parent–teacher associations.

## 14.3   Asynchronous Interactions: Different Time, Different Place

Cooperation across time and space is one of the gifts of technology. Durable messages transmitted electronically enable cooperation and therefore, for many users, electronic mail is a popular starting point. Electronic mail is widely appreciated, but it can be too loosely structured (endless chatting with no process or leader to reach a goal or to make a decision), too overwhelming (hundreds of messages per day can be difficult to absorb effectively), and too transient (lack of storage organization may make it difficult to locate relevant messages, and late joiners in a discussion have no means to catch up on earlier comments). To remedy these problems, structured methods for electronic conferencing have been created (Hiltz and Turoff, 1978; Hiltz, 1984) and have been applied in widely used software such as COSY and FirstClass. Filtering and archiving tools in commercial electronic-mail packages—such as Eudora, Lotus cc:Mail, or Microsoft Mail—enable users to manage incoming and previously received electronic mail. Web browsers, such as Netscape Navigator and Microsoft Internet Explorer, and commercial services such as CompuServe and America Online, provide mail handling as well. Setting up private mailing lists, web sites, and discussion groups is also getting easier. Support for teams, whole organizations, or larger communities is beginning to emerge, along with more structured work processes that are supported by computing networks.

14.3.1 Electronic Mail

The atomic unit of cooperation is the electronic-mail message; the FROM party sends a message to the TO party. Electronic-mail systems (Fig. 14.1) share the notion that an individual can send a message to another individual or a list of individuals. Messages usually are delivered in seconds or minutes, and replying is easy and rapid.

Electronic-mail messages typically contain text only, but increasingly graphics, spreadsheets, sounds, animations, web pointers, or other structured objects can be included. The quests for more flexibility and for instantaneous distribution are persistent. Sending graphics or spreadsheets is becoming more common as standard formats and effective conversions emerge. Still rare is the availability of video electronic mail, but it should spread during the next decade. Those users who have been able to add video comments to their electronic mail say that they like this feature and use it regularly—it does seem to have the capacity to add a more personal touch. Problems of standardization
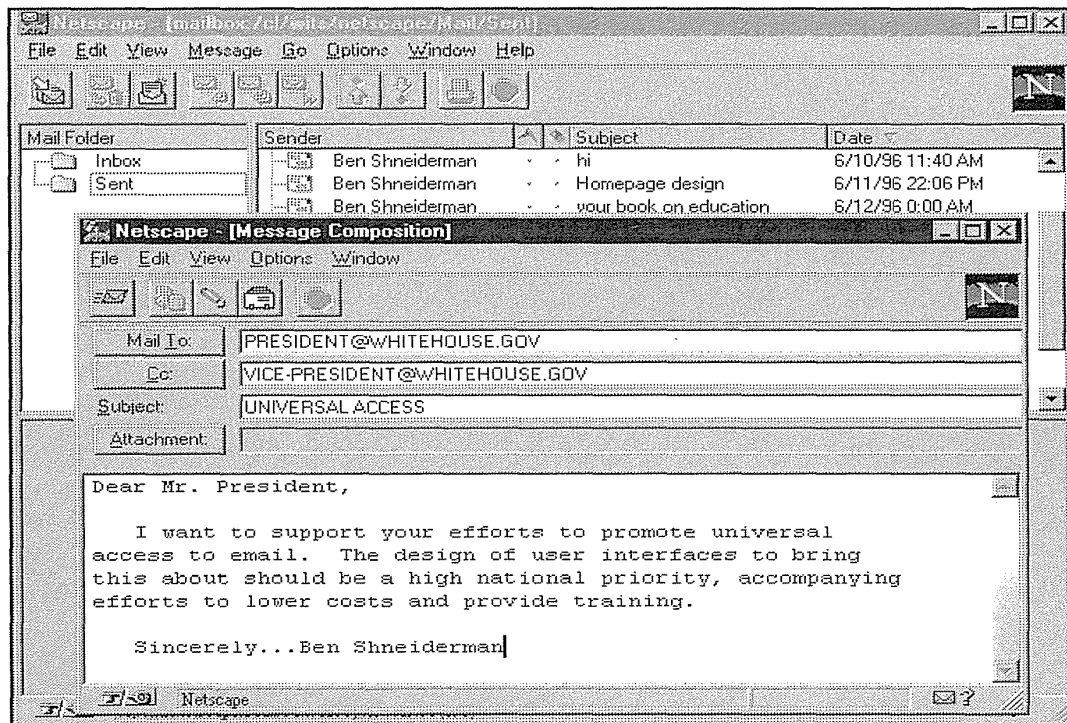
Figure 14.1

Electronic-mail system for Netscape Navigator, showing Inbox, Sent mail list, and a current note being composed. (© 1996 Netscape Communication Corporation. Used with permission.)

across video systems and the need for good user interfaces are apparent in the reports from early projects (Borenstein, 1991; Hoffert and Gretsch, 1991).

Voice mail is effectively sent by the normal telephone networks, but the inclusion of a voice annotation in an electronic-mail message is possible. Telephone service by way of the Internet is also possible, although speech quality is generally below the quality on the telephone network.

Most electronic-mail systems provide fields for TO (list of recipients), FROM (sender), CC (list of copy recipients), DATE, and SUBJECT. Malone and collegues (1987) showed the surprising benefit of semistructured messages in their pioneering system called the Information Lens. If messages were identified as being a lecture announcement, then they would have fields with the time, date, place, speaker, title, and host, in addition to the talk abstract. The semistructured portions of the message enable more automatic filtering of incoming messages or automatic routing and replies. Users can specify that they do not want to receive lecture announcements with times after 5 P.M. or on weekends. Alternatively, users can specify that copies of certain messages will be sent to colleagues, to the secretarial staff, or to their assistants. This feature provides a basis for dealing with the dangers of information overload as users begin to receive dozens or hundreds of messages per day.

A still more structured version of electronic mail was based on a *speech-acts theory* of requests and commitments that compelled users to be explicit about their expectations for responses when a message was sent. However, many users found this approach too rigid (Flores et al., 1988).

An interesting and successful product is Lotus Notes, which was sold only in 200-unit lots to large corporations, but was successful in reaching more than 75 sites with over 50,000 licenses after its first year on the market. Notes provides support to integrate electronic mail, newsgroups, telephone-call tracking, status reporting, text-database searching, document sharing, meeting scheduling, and other cooperation tools. Some large multinational corporations recognized Notes as providing a competitive advantage because widely distributed communities of employees could conveniently share information, make decisions, and carry out complex action plans. The emergence of the World Wide Web has challenged the success of Notes, but Notes still has advantages in better security control and a more structured environment.

Electronic mail has become widespread, but making it universal will require increased simplification, improved training, easier filtering, and lower-cost hardware plus network service (Anderson et al., 1995). The incompatibility of the dozens of systems is slowly being overcome, in part by the efforts of the Electronic Mail Association, by pressure from users, and by commercial realities that force cooperation. The possibility of including media richer than text is an attraction for some users who still prefer FAX machines to electronic mail, because even poor graphics are better than no graphics. Online directories, which are emerging on the web, might be facilitators, since it is still necessary to know a person's electronic mail address before sending

a message. Such online directories would also include group lists and the capacity to create new group lists conveniently, so that whole communities could be reached easily. Finally, improved archiving and retrieval systems would enable users to find old electronic-mail messages conveniently and rapidly. The dangers of junk electronic mail remain, and even noble ideas of cooperation can be undermined by users who fail to be polite, nuisances who persistently disrupt, electronic snoopers who do not respect privacy, or calculating opportunists who abuse their privileges.

14.3.2   Newsgroups and network communities

Electronic mail is a great way to get started in electronic communication, but its basic features need extension to serve the needs of communities. When a group of people use electronic mail for focused discussions, tools to organize the discussion and to provide an accessible historical record are needed. One popular strategy is the *USENET newsgroups,* in which thousands of topics are listed. Newsgroups users initiate action by selecting the newsgroup they want and reading as many previous notes and related comments as they wish. Typically, the past few weeks of notes are maintained on the user's machine. Global search of all newsgroups has only lately become an option by way of the web search engines. A more orderly community structure is the *listserv,* to which individuals must subscribe to receive electronic-mail notices. Still more structured is the *online conference,* in which additional tools are available for voting and for using online directories of users and documents.

Listservs can be moderated by a leader or can simply act as a mail reflector, sending out copies of received electronic-mail notes to all people who have subscribed. Users can get flooded with listserv electronic-mail notes, so the decision to subscribe can be a serious commitment. The listserv server machine keeps an archive of notes that is searchable, and a subscriber list. Users can obtain listserv commands by sending a one-word `help` command to the listserv host.

```
LISTSERV Version 1.8b: Most commonly used commands
Info          <topic|listname>        Order documentation
Lists         <Detail|Short|Global>   Get a description of all lists
SUBscribe     listname <full name>    Subscribe to a list
SIGNOFF       listname                Sign off from a list
SIGNOFF       * (NETWIDE              - from all lists on all servers
REView        listname <options>      Review a list
Query         listname                Query your subscription options
SET           listname options        Update your subscription options
INDex         <filelist_name>         Order a list of LISTSERV files
GET           filename filetype       Order a file from LISTSERV
REGister      full_name|OFF           Tell LISTSERV about your name
```

Thousands of newsgroups, listservs, and conferences have emerged around the world, administered by devoted individuals who keep the discussion moving, filter malicious or unsavory messages, and act as the
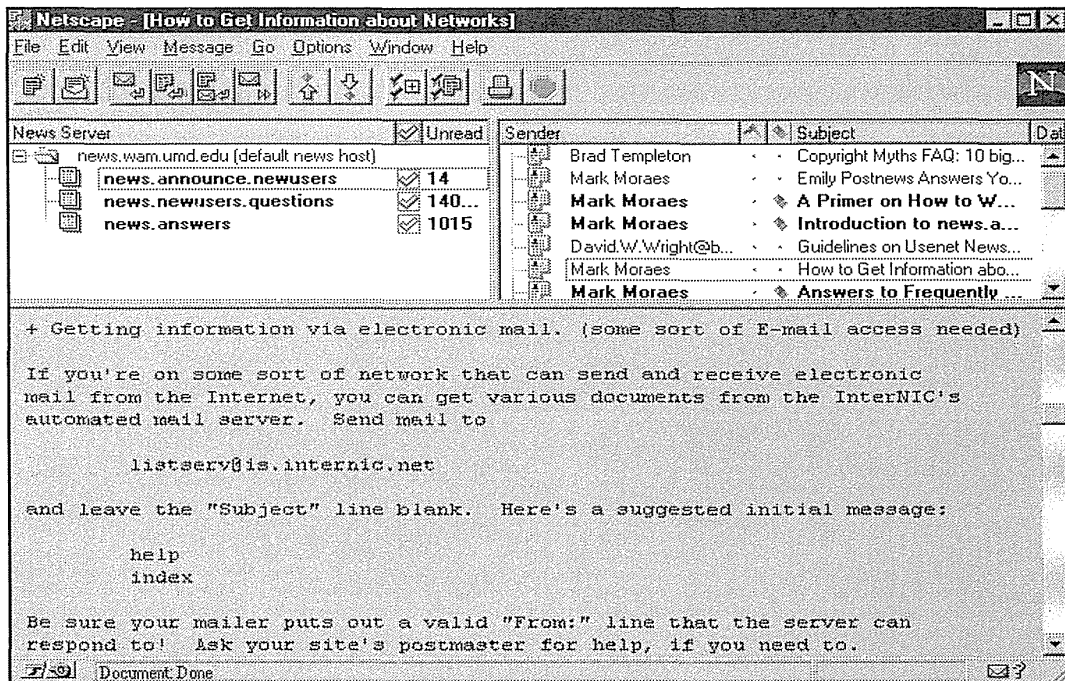
Figure 14.2

News reader system for Netscape Navigator, showing News Server list, a list of announcements for new users, and part of one announcement. (© 1996 Netscape Communication Corporation. Used with permission.)

Gertude Steins of the modern electronic salons. Notes may contain a question, an offer to buy or sell, interesting news, a joke, or a "flame" (abusive criticism) (Fischer and Stevens, 1991). Each item has a short one-line heading and an arbitrarily long body (Fig. 14.2).

The user interfaces tend to be simple to accommodate even low-speed dial-in access. Choices are few; the intrigue lies in the complexity of the conversations, especially in spirited replies and debates. As usage increases, the systems operator (often known as the SYSOP) must decide whether to split topics into more focused subtopics to avoid the overwhelming participants with thousands of new messages. Ensuring that the communities remain interesting is a challenge; if a group of advanced programmers discussing esoteric details is invaded by novices asking beginner-level questions, the experts will want to split off to re-form their own discussion group. Communities can be found for most computer-related issues, but other topics—such as movies, kayaking, rap music, folk dancing, and restaurants—are popular.

Practical information exchange is common for diverse groups such as cancer researchers, NASA scientists, handicapped users, and human-factors researchers. Within corporations, universities, or government agencies, specialized groups may be established for topics such as corporate policy, medical-insurance information, or product updates.

Newsgroups or listservs are usually open to all, whereas conferences are designed to provide access to a known community (Hiltz, 1984; Hiltz and Turoff, 1985). A conference is usually moderated, meaning that a conference leader invites participants, poses an issue or theme, and keeps the discussion going if a question is unanswered or if some participants fail to read new notes in a timely manner. Conferences are more likely to have voting features to allow consensus formation or decision making. Votes may be required within 48 hours, with results posted by the moderator. Thoughtful discussions within a conference are encouraged because participants can consider their position judiciously, consult other materials, and phrase their contributions carefully, without the pressure for an immediate comment that is inherent in a telephone call or face-to-face meeting.

Online magazines and newsletters are proliferating, with audiences growing rapidly. *HotWired, Electric Minds,* and *C | Net* focus on the web world; *Slate* is a general news and opinion magazine; and specialized newsletters are emerging in every discipline. Often, they are paid for by advertising, but some charge subscription fees. Hundreds of online newspapers complement the printed editions, and it seems likely that traditional newspaper features, such as financial tables and classified ads, will shift to online-only modes. Similarly, small scientific journals with fewer than 1000 subscribers seem likely to become available in electronic-only form.

Network communities have become controversial. In one case U.S. federal-government agents confiscated computer equipment alleging that illegal information was being posted, and the hacker community joined forces to protect the accused. The First Amendment principle of freedom of speech should be extended to electronic speech, but there are dangers of illegal activities. Some network communities have been criticized for spreading racist material, so the challenge is to preserve valued freedoms and rights without allowing harm. Congressional advocates of controlling online obscenity succeeded in quickly passing the Communications Decency Act of 1996, but it was just as quickly struck down as unconstitutional by federal judges.

Instead of seeking a conversation, many computer-network users are eager to scour remote databases for useful materials to download (load onto their personal computers). Creators of programs, images, databases, and so on often seek to publish their materials electronically, or to *upload* them, for people who can put these products to use. Some services provide

informative listings of available materials, but more could be done to provide effective library resources.

Enthusiasts delight in the free exchange of *shareware, freeware, and public-access software.* A slight variation is the honor system in which downloaded software includes a request for a modest fee ($10 to $100) to become a registered user; such users might receive notices of changes or a printed user manual.

14.4   Synchronous Distributed: Different Place, Same Time

The dream of being in two places at one time became realizable with modern technologies such as telephone and television; now, being in 10 places at once is possible through *synchronous distributed applications* such as *group editing.* For example, in the groundbreaking GROVE (for GRoup Outline Viewing Editor system), multiple users can edit the same document simultaneously (Ellis et al, 1991). Coordination is accomplished by voice communication. The default mode in GROVE is to allow every user to type simultaneously—there is no locking. Although the authors report that collisions are surprisingly infrequent, since users tend to work on different parts of a document, locking by sentence or paragraph would seem to be a necessary option for some situations. Both GROVE and a follow-on system, rIBIS (Rein et al., 1991) included small images of current users.

Important features in the development of ShrEdit (for shared editor) at the University of Michigan were the mixture of private and public workspaces, identity of participants, location of actions, and care with updating (Olson et al., 1990). These same issues of ownership and control were highly visible in groups of inner-city sixth graders who used a group editor to write a magazine on prejudice (Mitchell et al., 1995). Shared workspaces for drawing (Greenberg et al., 1995), creating hypermedia documents (Mark et al., 1996), performing flexible teamwork (Roseman & Greenberg, 1996), and doing collaborative design (Ishii et al., 1994) expand the possibilities (Figs. 14.3 and 14.4).

Shared-editor sessions or shared spreadsheets seem to be simple yet potentially popular applications. IBM's early CVIEW demonstrated the benefits of shared screens for customer assistance. Users with problems could call the customer engineer, and both could see the same screens as the engineer walked through the solution. People have given demonstrations of new software at multiple sites by showing screens to dozens of people while talking on a conference call. Another potential application is to allow sharing of
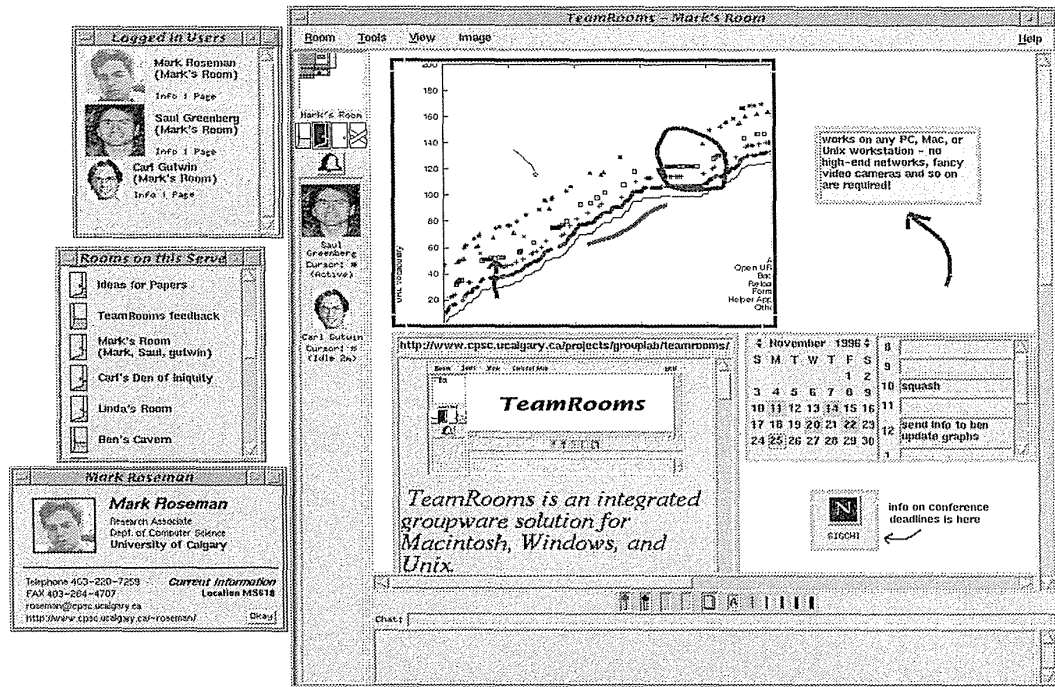
Figure 14.3

TeamRooms, a program that simulates real-life team meeting rooms that provide a shared space for a workgroup. When people enter, they are automatically connected with everyone in the room. In this example of Mark's room, Saul and Carl are commenting on and marking scientific data. Multiple graphical groupware applications can be active. (Used with permission of the University of Calgary, Alberta, Canada.)

information for applications such as airlines reservations. When the agent has located a selection of possible flights, it would be convenient to be able to *show* the customer rather than to read the list. The customer could then make the selection and would have an electronic copy to save, print, or include in other documents. An innovative commercial direction is the development of interactive games that permit two or more people to participate simultaneously in poker, chess, or complex fantasies.

Even simple exchanges of text messages in systems such as CHAT, Internet Relay Chat (IRC), or TALK produce lively social clubhouses on many distributed online information services. Participants may be genuinely caring and helpful, or maybe wisecracking *flamers* more intent on a putdown with a
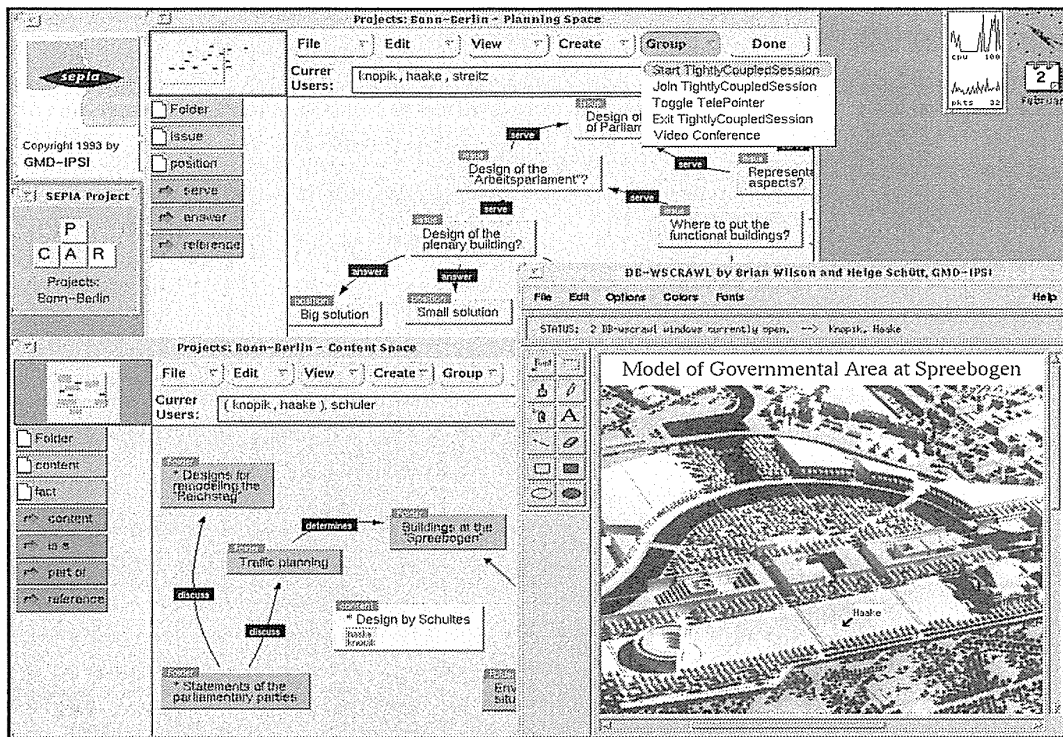
Figure 14.4

SEPIA (Structured Elicitation and Processing of Ideas for Authoring), a program that supports hypermedia-document production by group discussions with multiple remote users (Streitz et al., 1996). In this example, the users Knopik, Haake, Streitz, and Schuler are working in different Activity Spaces in loosely (top) and in tightly coupled (bottom) collaboration modes. They can share views at the network level of typed hypermedia nodes and links, as well as sharing the content of nodes (lower right) and using telepointers at all levels. SEPIA laid the basis for the DOLPHIN system (Mark et al., 1996), which supports electronic meeting collaboration. (Used with permission of Norbert Streitz, GMD-IPSI, Darmstadt, Germany.)

tendency to violent or obscene language. Sometimes, users take on new personalities with engaging names such as Gypsy, Larry Lightning, or Really Rosie. The social chatter can be light, provocative, or intimidating.

Elaborate MUD (for Multi-User Dungeons or Dimensions systems) offer fantasy environments (although most are still text-oriented), where users can take on novel identities, personas, or avatars while on heroic quests (Turkle, 1995). According to a web MUD page: "You can walk around, chat with other characters, explore dangerous monster-infested areas, solve puzzles, and even create your very own rooms, descriptions and items."

The hardware, network, and software architectures to support synchronous applications with multimedia capabilities are being developed at many

sites (Crowley et al., 1990; Patterson et al., 1990; Greenberg and Marwood, 1994); each project is dealing with the problems of delays, locking, sharing, and synchronization.

Innovative attempts to use video to bridge distance have been made. A simple approach is to have two video cameras and displays so that you can have an informal chat:

> Imagine sitting in your workplace lounge having coffee with colleagues. Now imagine that you and your colleagues are still in the same room, but are separated by a large sheet of glass that does not interfere with your ability to carry on a clear, two-way conversation. Finally, imagine that you have split the room into two parts and moved one part 50 miles down the road, without impairing the quality of your interaction with your friends.

That scenario illustrates the goal of the VideoWindow project (Fish et al., 1990): to extend a shared space over considerable distance without impairing the quality of the interactions among users or requiring any special actions to establish a conversation.

Approaches to videoconferencing range from special rooms to users working at their own desks using their normal computer systems while seeing one or more other participants in the video conference (Watabe, 1990; Mantei et al., 1991; Bly et al., 1993; Isaacs et al., 1995, 1996). The convenience of *desktop videoconferencing (DTVC)* is great, since this strategy permits users to have access to their papers and computer systems during the conference. Specialized videoconferencing rooms that are reserved by appointment give the event greater significance, and the equipment quality is usually higher because the equipment can be a shared resource.

In the University of Toronto's early DTVC system, called CAVECAT (for Computer Audio Video Enhanced Collaboration and Telepresence), up to four sites could be viewed on a single monitor. Problems included slow response time for entering or leaving a session, distracting background audio that exacerbated the difficulty of determining who was speaking, inappropriate lighting, difficulty in making eye contact (participants would look at their monitors rather than into the cameras), changed social status, small image size, and potential invasion of privacy.

The move from research prototypes to widely used DTVC systems is happening as costs drop, adequate bandwidth becomes available, and interfaces improve. Cornell University's CU-SeeMe system offers free software that runs on most personal computers with no special hardware and ordinary video cameras (Dorcey, 1995 and http://cu-seeme.cornell.edu). The low resolution (320 × 240) grayscale images are transmitted via the Internet at whatever frame rates are possible (Fig. 14.5). The images are often jerky, but CU-SeeMe is used for many personal conferences, professional work, and distance learning. More sophisticated commercial systems include Intel's
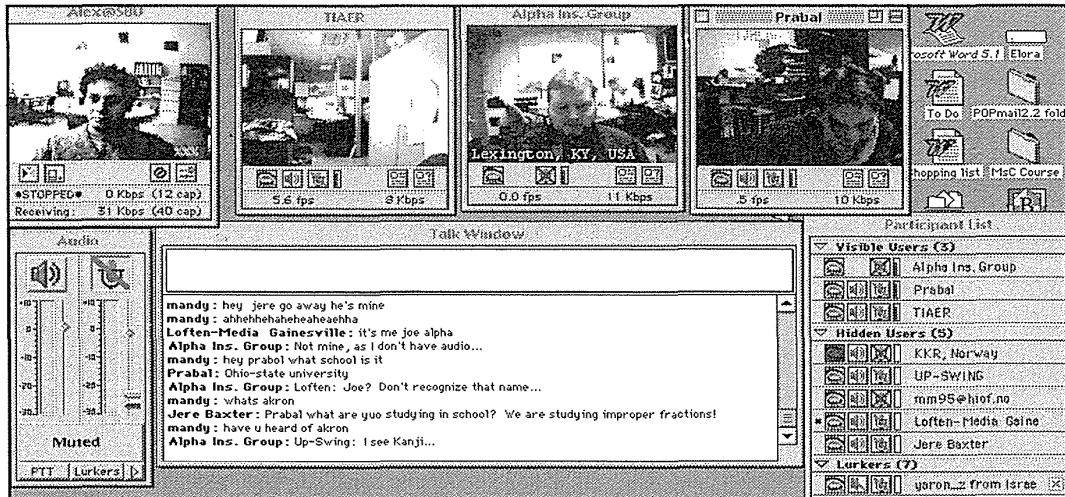
Figure 14.5

CUSee-Me used as a videoconferencing system. Four users are displayed in video windows as they share a common talk window. (Prepared by Alessandro Barabesi.)

ProShare II, which provides good images on Integrated Services Digital Network (ISDN) lines and supports shared workspaces for document processing, spreadsheets, and so on.

The PictureTel line of videoconferencing platforms provides increasingly higher-quality services on telephone lines, the Internet, local-area networks, and leased lines (Fig. 14.6) (http://www.picturetel.com). Once users have had the pleasure of seeing one another on video, done the required hand waving, and adjusted their lighting, cameras, hair, and clothes, it is time to get down to business. Some meetings are simple discussions that replace face-to-face visits, and the improvement over the telephone is the capacity to assess facial-expression and body-language cues for enthusiasm, disinterest, or anger. Many meetings include conferencing over some object of interest, such as a document, map, or photo. Developers emphasize the need for convenient turn taking and document sharing by using terms such as *smooth, lightweight,* or *seamless integration.*

Controlled experimentation on performance with different media is guiding designers in shaping effective systems. Chapanis's classic studies (1975) and recent work confirm that a voice channel is an important component for discussion of what participants see on a shared display. In one comparison, a shared workspace on a computer display was used without audio or video, with audio alone, and with audio and video (Gale, 1990). One group of four performed three tasks five times in each media format. Significant differences were found for the meeting-scheduling task, which took almost twice as long with the