Select the Proper Kinds of Windows

A window is an area of the screen, usually rectangular in shape, defined by a border that contains a particular view of some area of the computer or some portion of a person's dialog with the computer. It can be moved and rendered independently on the screen. A window may be small, containing a short message or a single field, or it may be large, consuming most or all of the available display space. A display may contain one, two, or more windows within its boundaries. In this step the following is addressed:

- A window's characteristics.
- A window's components.
- A window's presentation styles.
- The types of windows available.
- Organizing window system functions.
- A window's operations.
- Web system frames and pop-up windows.

Window Characteristics

A window is seen to possess the following characteristics:

- A name or title, allowing it to be identified.
- A size in height and width (which can vary).

- A state, accessible or active, or not accessible. (Only active windows can have their contents altered.)
- Visibility—the portion that can be seen. (A window may be partially or fully hidden behind another window, or the information within a window may extend beyond the window's display area.)
- A location, relative to the display boundary.
- Presentation, that is, its arrangement in relation to other windows. It may be tiled, overlapping, or cascading.
- Management capabilities, methods for manipulation of the window on the screen.
- Its highlight, that is, the part that is selected.
- The function, task, or application to which it is dedicated.

The Attraction of Windows

The value of windowing is best seen in the context of a task or job. A person performs a variety of tasks, often in a fairly unstructured manner. A person is asked to monitor and manipulate data from a variety of sources, synthesize information, summarize information, and reorganize information. Things are seldom completed in a continuous time frame. Outside events such as telephone calls, supervisor or customer requests, and deadlines force shifts in emphasis and focus. Tasks start, stop, and start again. Materials used in dealing with the tasks are usually scattered about one's desk, being strategically positioned in the workspace to make handling the task as efficient as possible. This spatial mapping of tools helps people organize their work and provides reminders of uncompleted tasks. As work progresses and priorities change, materials are reorganized to reflect the changes.

Single-screen technology supports this work structure very poorly. Since only one screen of information can be viewed at one time, comparing or integrating information from different sources and on different screens often requires extensive use of one's memory. To support memory, a person is often forced to write notes or obtain printed copies of screens. Switching between tasks is difficult and disruptive, and later returning to a task requires an extensive and costly restructuring of the work environment.

The appeal of windowing is that it allows the display workspace to mirror the desk workspace much more closely. This dramatically reduces one's short-term memory load. One's ability to do mental calculations is limited by how well one keeps track of one's place, one's interim conclusions and products, and, finally, the results. Windows act as external memories that are an extension of one's internal memory. Windows also make it much easier to switch between tasks and to maintain one's context, since one does not have to reestablish one's place continually. In addition, Windows provide access to more information than would normally be available on a single display of the same size. Overwriting, or placing more important information on top of that of less importance at that moment, does this.

While all the advantages and disadvantages of windows are still not completely understood, windows do seem to be useful in the following ways.

Presentation of Different Levels of Information

Information can be examined in increasing levels of detail. A document table of contents can be presented in a window. A chapter or topic selected from this window can be simultaneously displayed in more detail in an adjoining window. Deeper levels are also possible in additional windows.

Presentation of Multiple Kinds of Information

Variable information needed to complete a task can be displayed simultaneously in adjacent windows. An order-processing system window could collect a customer account number in one window and retrieve the customer's name and shipping address in another window. A third window could collect details of the order, after which another window could present factory availability of and shipping dates for the desired items. Significant windows could remain displayed so that details may be modified as needed prior to order completion. Low inventory levels or delayed shipping dates might require changing the order.

Sequential Presentation of Levels or Kinds of Information

Steps to accomplish a task can be sequentially presented through windows. Successive windows are presented until all the required details are collected. Key windows may remain displayed, but others appear and disappear as necessary. This sequential preparation is especially useful if the information-collection process leads down various paths. An insurance application, for example, will include different types of coverage. A requested type of coverage might necessitate the collection of specific details about that type of coverage. This information can be entered into a window presented to collect the unique data. The windows disappear after data entry, and additional windows appear when needed.

Access to Different Sources of Information

Independent sources of information may have to be accessed at the same time. This information may reside in different host computers, operating systems, applications, files, or areas of the same file. It may be presented on the screen alongside the problem, greatly facilitating its solution. For instance, a writer may have to refer to several parts of a text being written at the same time. Or, a travel agent may have to compare several travel destinations for a particularly demanding client.

Combining Multiple Sources of Information

Text from several documents may have to be reviewed and combined into one. Pertinent information is selected from one window and copied into another.

Performing More Than One Task

More than one task can be performed at one time. While waiting for a long, complex procedure to finish, another can be performed. Tasks of higher priority can interrupt less important ones. The interrupted task can then be resumed without the necessity to "close down" and "restart."

Reminding

Windows can be used to remind the viewer of things likely to be of use in the near future. Examples might be menus of choices available, a history of the path followed or the command choices to that point, or the time of an important meeting.

Monitoring

Changes, both internal and external, can be monitored. Data in one window can be modified and its effect on data in another window can be studied. External events, such as changes in stock prices, out of normal range conditions, or system messages can be watched while another major activity is carried out.

Multiple Representations of the Same Task

The same thing can be looked at in several ways—for example, alternate drafts of a speech, different versions of a screen, or different graphical representations of the same data. A maintenance procedure may be presented in the form of textual steps and illustrated graphically at the same time.

Constraints in Window System Design

Windowing systems, in spite of their appeal and obvious benefits, have failed to completely live up to their expectations. In the past, a windows user interface has been described as "chaotic" because of the great amount of time users must spend doing such things as pointing at tiny boxes in window borders, resizing windows, moving windows, closing windows, and so forth. The problems with windowing systems can be attributed to three factors: historical considerations, hardware limitations, and human limitations.

Historical Considerations

Historically, system developers have been much more interested in solving hardware problems than in user considerations. Since technical issues abound, they have received the most attention. There has been very little research addressing design issues and their impact on the usability of window systems. Therefore, there are few concrete window design guidelines to aid designers.

This lack of guidelines makes it difficult to develop acceptable and agreeable window standards. While many companies have developed style guides, they are very general and limited in scope to their products. Standardization is also made more difficult by the complexity and range of alternatives available to the designer. Without user performance data, it is difficult to compare realistically the different alternatives, and design choices become a matter of preference.

Standardization of the interface is also inhibited by other factors. Some software developers, who are proud of their originality, see standards as a threat to creativity and its perceived monetary rewards. Some companies are wary of standards because they fear that other companies are promoting standards that reflect their own approach. Finally, some companies have threatened, or brought, legal action against anyone who adopts an approach similar to their own.

The result is that developers of new systems create another new variation each time they design a product, and users must cope with a new interface each time they encounter a new windowing system.

Hardware Limitations

Many of today's screens are not large enough to take full advantage of windowing capabilities. As a result, many windows are still of "Post-It" dimensions. As already mentioned, there is some evidence that many users of personal computers expand their windows to cover a full screen. Either seeing all the contents of one window is preferable to seeing small parts of many windows or the operational and visual complexity of multiple windows is not wanted.

Also, the slower processing speeds and smaller memory sizes of some computers may inhibit the use of windows. A drain on the computer's resources may limit feedback and animation capabilities, thereby reducing the system's usability. Poor screen resolution and graphics capability may also deter effective use of windows by not permitting sharp and realistic drawings and shapes.

Human Limitations

A windowing system, because it is more complex, requires the learning and using of more operations. Much practice is needed to master them. These window management operations are placed on top of other system operations, and window management can become an end in itself. This can severely detract from the task at hand. In one study comparing full screens with screens containing overlapping windows, task completion times were longer with the window screens, but the non-window screens generated more user errors. After eliminating screen arrangement time, however, task solution times were shorter with windows. The results suggest that advantages for windows do exist, but they can be negated by excessive window manipulation requirements.

It is also suggested that to be truly effective, window manipulation must occur implicitly as a result of user task actions, not as a result of explicit window management actions by the user.

Other Limitations

Other possible window problems include the necessity for window borders to consume valuable screen space, and that small windows providing access to large amounts of information can lead to excessive, bothersome scrolling.

Where To?

In spite of their problems, windows do have enormous benefits and are here to stay. So, we must cope with their constraints for now and, in the meantime, enjoy the benefits they possess.

Components of a Window

A typical window may be composed of up to a dozen or so elements. Some appear on all windows; others only on certain kinds of windows, or under certain conditions. For consistency purposes, these elements should always be located in the same position within a window. Most windowing systems provide consistent locations for elements in their own windows. Some inconsistencies do exist in element locations between different systems, however, as do some differences in what the elements are named, or what graphic images or icons are chosen to identify them. What follows is a description of typical window components and their purposes, with emphasis on the most popular windowing system, *Microsoft Windows*. Specifically reviewed will be primary windows, secondary windows, and a form of secondary window called the dialog box. An illustration of a primary window is found in Figure 5.1. Illustrations of secondary

Figure 5.1 Microsoft Windows primary window.

windows and dialog boxes are illustrated in Figures 5.8 and 5.13. How these different types of windows are used will be described in a later section in this chapter. A summary of window components for these types of windows is also found in Table 5.1.

Frame

A window will have a frame or border, usually rectangular in shape, to define its boundaries and distinguish it from other windows. While a border need not be rectangular, this shape is a preferred shape for most people. Also, textual materials, which are usually read from left to right, fit most efficiently within this structure. The border comprises a line of variable thickness and color. This variation can be used as an aid in identifying the type of window being displayed. Windows filling an entire screen may use the screen edge as the border. If a window is resizable, it may contain control points for sizing it. If the window cannot be resized, the border coincides with the edge of the window.

Title Bar

The title bar is the top edge of the window, inside its border and extending its entire width. This title bar is also referred to by some platforms as the *caption, caption bar,* or *title area*. The title bar contains a descriptive title identifying the purpose or content of the window. In Microsoft Windows, the title bar may also possess, at the extreme left and right ends, control buttons (described below) for retrieving the system menu and performing window resizing. The title bar also serves as a control point for moving the window and as an access point for commands that apply to a window. For example, as an access point, when a user clicks on the title bar using the secondary mouse button, the pop-up or shortcut menu for the window appears. Pressing the Alt-Spacebar key combination also displays the shortcut menu for the window. Title bars are included on all primary and secondary windows. Title bar text writing guidelines are described in Step 8 "Write Clear Text and Messages."

Microsoft recommends that one never place application commands or other controls in the title bar. Doing so may conflict with the special user controls Windows adds for configurations that support multiple languages.

Title Bar Icon

Located at the left corner of the title bar in a primary window, this button is used in Windows to retrieve a pull-down menu of commands that apply to the object in the window. It is 16 × 16 version of the icon of the object being viewed. When clicked with the secondary mouse button, the commands applying to the object are presented. Microsoft suggests that:

- If the window contains a tool or utility (that is, an application that does not create, load, and save its own data files), a small version of the application's icon should be placed there instead.

Table 5.1 Microsoft Windows Components

COMPONENT	WINDOWS CONTAINING COMPONENT		
	PRIMARY	SECONDARY	DIALOG BOX
Frame or Border • Boundary to define shape. • If sizable, contains control points for resizing.	X	X	X
Title Bar Text • Name of object being viewed in window. • Control point for moving window.	X	X	X
Title Bar Icon • Small version of icon for object being viewed. • Access point for commands that apply to the object.	X		
Title Bar Buttons • Shortcuts to specific commands.	X	X	X
Close	X	X	X
Minimize/Maximize/Restore	X		
What's This? — Displays context-sensitive Help about any object displayed on window.		X	X
Menu Bar • Provides basic and common application commands.	X		
Status Bar • An area used to display status information about what is displayed in window.	X		
Scroll Bar • Standard control to support scrolling.	X		
Size Grip • Control to resize window, located at right side of status bar.	X		

- If the application creates, loads, and saves documents or data files and the window represents the view of one of its files, a small version of the icon that represents its document or data file type should be placed there.
- Even if the user has not yet saved the file, display the data file icon rather than the application icon, and again display the data file icon after the user saves the file.

Window Sizing Buttons

Located at the right corner of the title bar, these buttons are used to manipulate the size of a window. The leftmost button, the *minimize* button — inscribed with a short horizontal line toward the bottom of the button — is used to reduce a window to its minimum size, usually an icon. It also hides all associated windows. The *maximize* button — typically inscribed with a large box — enlarges a window to its maximum size, usually the entire screen. When a screen is maximized, the *restore* button replaces the maximize button, since the window can no longer be increased in size. The restore button — typically inscribed with a pair overlapping boxes — returns a window to the size it had before a minimize or maximize action was performed. A *close* button — typically inscribed with an X — closes the window. Minimize, maximize, and close buttons are shown in Figure 5.1. These command buttons are graphical equivalents to the actions available through the Title Bar icon.

Sizing buttons are included on primary windows only. All buttons on a primary window's title bar must have equivalent commands on the pop-up or shortcut menu for that window.

When these buttons are displayed, use the following guidelines:

- When a window does not support a command, do not display its command button.
- The *Close* button always appears as the rightmost button. Leave a gap between it and any other buttons.
- The *Minimize* button always precedes the *Maximize* button.
- The *Restore* button always replaces the *Maximize* button or the *Minimize* button when that command is carried out.

What's This? Button

The *What's This?* Button, which appears on secondary windows and dialog boxes, is used to invoke the What's This? Windows command to provide contextual Help about objects displayed within a secondary window. When provided, it is located in the upper-right corner of the title bar, just to the left of the close button. It is inscribed with a question mark, as illustrated in Figure 5.2.

On a primary window this command is accessed from the Help drop-down menu. This command may also be included as a button on a toolbar or as a command on a pop-up menu for a specific object. This command is described more fully in Step 9 "Provide Effective Feedback and Guidance and Assistance."

Figure 5.2 What's This? button.

Menu Bar

A menu bar is used to organize and provide access to actions. It is located horizontally at the top of the window, just below the title bar. A menu bar contains a list of topics or items that, when selected, are displayed on a pull-down menu beneath the choice. A system will typically provide a default set of menu actions that can be augmented by an application. In the past, some platforms have called the menu bar an *action bar*. Menu bar design guidelines were presented in Step 4 "Develop System Menus and Navigation Schemes." The contents of the menu bar and its pull-downs are determined by the application's functionality and the context in which the user is interacting with it.

Status Bar

Information of use to the user can be displayed in a designated screen area or areas. They may be located at the top of the screen in some platforms and called a *status area*, or at the screen's bottom. Microsoft recommends the bottom location and refers to this area as the *status bar*. It is also referred to by other platforms as a *message area* or *message bar*.

Microsoft Windows suggests using the status bar to display information about the current state of what is being viewed in the window, descriptive messages about a selected menu or toolbar button, or other noninteractive information. It may also be used to explain menu and control bar items as the items are highlighted by the user.

Scroll Bars

When all display information cannot be presented in a window, the additional information must be found and made visible. This is accomplished by scrolling the display's contents through use of a scroll bar. A scroll bar is an elongated rectangular container consisting of a scroll area or shaft, a slider box or elevator, and arrows or anchors at each end. For vertical scrolling, the scroll bar is positioned at the far right side of the work area, extending its entire length. Horizontal scrolling is accomplished through a scroll bar located at the bottom of the work area. Scroll bars are more fully described in Step 7 "Choose the Proper Screen-Based Controls."

Split Box

A window can be split into two or more pieces or panes by manipulating a *split box* located above a vertical scroll bar or to the left of a horizontal scroll bar. A split box is sometimes referred to as a *split bar*. A window can be split into two or more separate viewing areas that are called *panes*. Splitting a window permits multiple views of an object. A split window allows the user to:

- Examine two parts of a document at the same time.
- Display different, yet simultaneous, views of the same information.

To support the splitting of a window that is not presplit by design, include a split box. The split box should be just large enough for the user to successfully target it with the pointer; the default size of a size handle, such as the window's sizing border, is a good guideline.

Toolbar

Toolbars, illustrated in Figure 5.3, are permanently displayed panels or arrays of choices or commands that must be accessed quickly. They are sometimes called *command bars*. Toolbars are designed to provide quick access to specific commands or options. Specialized toolbars are sometimes referred to as *ribbons, toolboxes, rulers,* or *palettes*. Each toolbar band includes a single-grip handle to enable the user to resize or rearrange the toolbars. When the user moves the pointer over the grip, it changes to a two-headed arrow. When the user drags the grip, the pointer changes to a split move pointer. To resize the toolbar to its maximum or minimum size, the user clicks the grip.

Toolbars may occupy a fixed position or be movable. The design of toolbars is discussed in Step 7.

Command Area

In situations where it is useful for a command to be typed into a screen, a command area can be provided. The desired location of the command area is at the bottom of the window. If a horizontal scroll bar is included in the window, position the command area just below it. If a message area is included on the screen, locate the command area just above it.

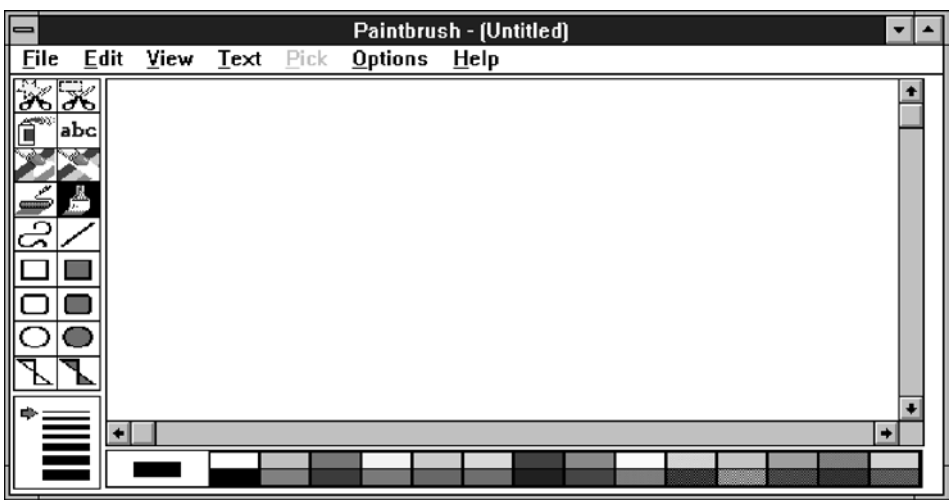

Figure 5.3 Toolbars.

Size Grip

A size grip is a Microsoft Windows special handle included in a window to permit it to be resized. When the grip is dragged the window resizes, following the same conventions as the sizing border. Three angled parallel lines in the lower-right corner of a window designate the size grip. If the window possesses a status bar, the grip is positioned at the bar's right end. Otherwise, it is located at the bottom of a vertical scroll bar, the right side of a horizontal scroll bar, or the junction point of the two bars. A size grip is shown in the lower-right corner of Figure 5.2.

Work Area

The work area is the portion of the screen where the user performs tasks. It is the open area inside the window's border and contains relevant peripheral screen components such as the menu bar, scroll bars, or message bars. The work area may consist of an open area for typing, or it may contain controls (such as text boxes and list boxes) or customized forms (such as spreadsheets). The work area may also be referred to as the *client area*.

Window Presentation Styles

The presentation style of a window refers to its spatial relationship to other windows. There are two basic styles, commonly called tiled or overlapping. In early windowing days, most systems commonly used one or the other style exclusively, seldom using both at the same time. Now, the user is usually permitted to select the style to be presented on the display.

Tiled Windows

Tiled windows, illustrated in Figure 5.4, derive their name from common floor or wall tile. Tiled windows appear in one plane on the screen and expand or contract to fill up the display surface, as needed. Most systems provide two-dimensional tiled windows, adjustable in both height and width. Some less-powerful systems, however, are only one-dimensional, the windows being adjustable in only one manner (typically the height). Tiled windows, the first and oldest kind of window, are felt to have these advantages:

- The system usually allocates and positions windows for the user, eliminating the necessity to make positioning decisions.
- Open windows are always visible, eliminating the possibility of them being lost and forgotten.
- Every window is always completely visible, eliminating the possibility of information being hidden.
- They are perceived as less complex than overlapping windows, possibly because there are fewer management operations or they seem less "magical."

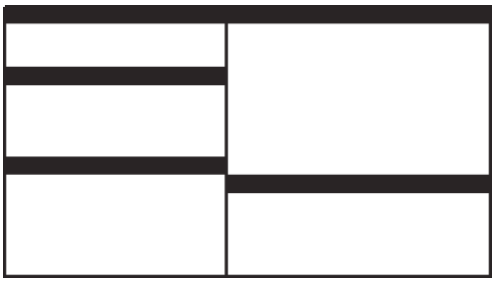

Figure 5.4 Tiled windows.

- They are easier, according to studies, for novice or inexperienced people to learn and use.
- They yield better user performance for tasks where the data requires little window manipulation to complete the task.

Perceived disadvantages include the following:

- Only a limited number can be displayed in the screen area available.
- As windows are opened or closed, existing windows change in size. This can be annoying.
- As windows change in size or position, the movement can be disconcerting.
- As the number of displayed windows increases, each window can get very tiny.
- The changes in sizes and locations made by the system are difficult to predict.
- The configuration of windows provided by the system may not meet the user's needs.
- They are perceived as crowded and more visually complex because window borders are flush against one another, and they fill up the whole screen. Crowding is accentuated if borders contain scroll bars or control icons. Viewer attention may be drawn to the border, not the data.
- They permit less user control because the system actively manages the windows.

Overlapping Windows

Overlapping windows, illustrated in Figure 5.5, may be placed on top of one another like papers on a desk. They possess a three-dimensional quality, appearing to lie on different planes. Users can control the location of these windows, as well as the plane in which they appear. The sizes of some types of windows may also be changed. Most systems today normally use this style of window. They have the following advantages:

- Visually, their look is three-dimensional, resembling the desktop that is familiar to the user.
- Greater control allows the user to organize the windows to meet his or her needs.
- Windows can maintain larger sizes.

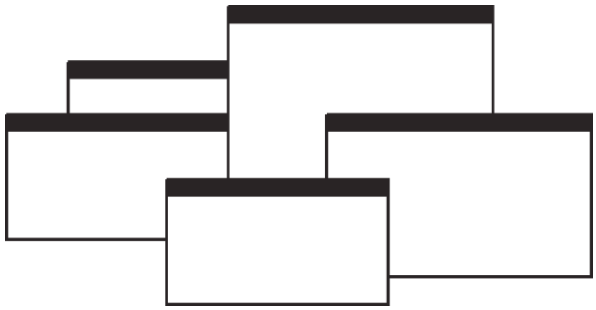

Figure 5.5 Overlapping windows.

- Windows can maintain consistent sizes.
- Windows can maintain consistent positions.
- Screen space conservation is not a problem, because windows can be placed on top of one another.
- There is less pressure to close or delete windows no longer needed.
- The possibility exists for less visual crowding and complexity. Larger borders can be maintained around window information, and the window is more clearly set off against its background. Windows can also be expanded to fill the entire display.
- They yield better user performance for tasks where the data requires much window manipulation to complete the task.

Disadvantages include the following:

- They are operationally much more complex than tiled windows. More control functions require greater user attention and manipulation.
- Information in windows can be obscured behind other windows.
- Windows themselves can be lost behind other windows and be presumed not to exist.
- That overlapping windows represent a three-dimensional space is not always realized by the user.
- Control freedom increases the possibility for greater visual complexity and crowding. Too many windows, or improper offsetting, can be visually overwhelming.

Cascading Windows

A special type of overlapping window has the windows automatically arranged in a regular progression. Each window is slightly offset from others, as illustrated in Figure

Advantages of this approach include the following:

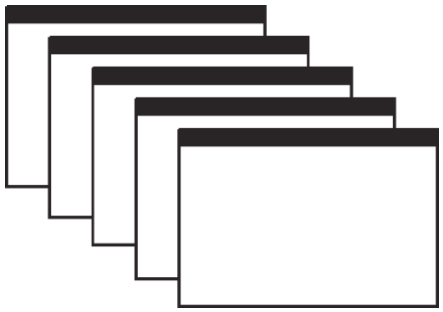

Figure 5.6 Cascading windows.

- No window is ever completely hidden.
- Bringing any window to the front is easier.
- It provides simplicity in visual presentation and cleanness.

Picking a Presentation Style

- Use tiled windows for:
  - Single-task activities.
  - Data that needs to be seen simultaneously.
  - Tasks requiring little window manipulation.
  - Novice or inexperienced users.
- Use overlapping windows for:
  - Switching between tasks.
  - Tasks necessitating a greater amount of window manipulation.
  - Expert or experienced users.
  - Unpredictable display contents.

Tiled windows. Tiled windows seem to be better for single-task activities and data that must be seen simultaneously. A study found that tasks requiring little window manipulation were carried out faster using tiled windows. They also found that novice users performed better with tiled windows, regardless of the task.

Overlapping windows. Overlapping windows seem to be better for situations that necessitate switching between tasks. A research study concluded that tasks requiring much window manipulation could be performed faster with overlapping windows but only if user window expertise existed. For novice users, tasks requiring much window manipulation were carried out faster with tiled windows. Therefore, the advantage to overlapping windows comes only after a certain level of expertise is achieved. Overlapping windows are the preferred presentation scheme.

Types of Windows

People's tasks must be structured into a series of windows. The type of window used will depend on the nature and flow of the task. Defining standard window types is again difficult across platforms because of the varying terminology and definitions used by different windowing systems, and changes in terminology for new versions of systems. For simplicity, the Microsoft Windows windowing scheme will be described. Summarized are a description of the window, its purpose, and its proper usage. Any other platform's windows may not behave exactly as presented, and some platform windows may exhibit characteristics common to more than one of the described window types.

Primary Window

- Proper usage:
  - Should represent an independent function or application.
  - Use to present constantly used window components and controls.
    - Menu bar items that are:
      - Used frequently.
      - Used by most, or all, primary or secondary windows.
    - Controls used by dependent windows.
  - Use for presenting information that is continually updated.
    - For example, date and time.
  - Use for providing context for dependent windows to be created.
  - Do not:
    - Divide an independent function into two or more primary windows.
    - Present unrelated functions in one primary window.

The *primary* window is the first one that appears on a screen when an activity or action is started. It is required for every function or application, possessing a menu bar and some basic action controls, as previously described. It should present the framework for the function's commands and data, and provide top-level context for dependent windows. It has also been variously referred to as the *application* window or the *main* window. In addition, it may be referred to as the *parent* window if one or more *child* windows exist. A Microsoft Windows primary window is shown in Figure 5.7.

The primary window is the main focal point of the user's activities and should represent an independent function. Avoid dividing an independent function into two or more primary windows, and avoid presenting unrelated functions in a single primary window. This tends to confuse people.

Independent functions should begin in a primary window. A primary window should contain constantly used window components such as frequently used menu bar items and controls (for example, control bars) used by dependent windows. Also include in a primary window continually updated information such as the date and time. The components of a primary window are summarized in Table 5.2.

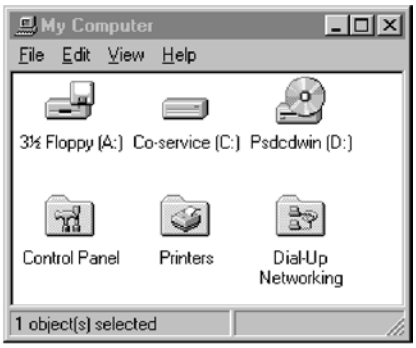

Figure 5.7 Microsoft Windows primary window.

Secondary Windows

- Proper usage:
  - For performing subordinate, supplemental, or ancillary actions that are:
    - Extended or more complex in nature.
    - Related to objects in the primary window.
  - For presenting frequently or occasionally used window components.
- Important guidelines:
  - Should typically not appear as an entry on the taskbar.
  - A secondary window should not be larger than 263 dialog units x 263 dialog units.

Secondary windows are supplemental windows. Secondary windows may be dependent upon a primary window or displayed independently of the primary window. They structurally resemble a primary window, possessing some of the same action controls (Close button) and possibly a What's This? button.

A *dependent* secondary window is one common type. It can only be displayed from a command on the interface of its primary window. It is typically associated with a single data object, and appears on top of the active window when requested. It is movable, and scrollable. If necessary, it uses the primary window's menu bar. Most systems permit the use of multiple secondary windows to complete a task. In general, dependent secondary windows are closed when the primary window closes, and hidden when their primary window is hidden or minimized.

An *independent* secondary window can be opened independently of a primary window—for example, a property sheet displayed when the user clicks the Properties command on the menu of a desktop icon. An independent secondary window can typically be closed without regard to the state of any primary window unless there is an obvious relationship to the primary window.

A Microsoft Windows secondary Window is illustrated in Figure 5.8.

Proper usage. Although secondary windows share many characteristics with primary windows, they also differ from primary windows in behavior and use. Secondary

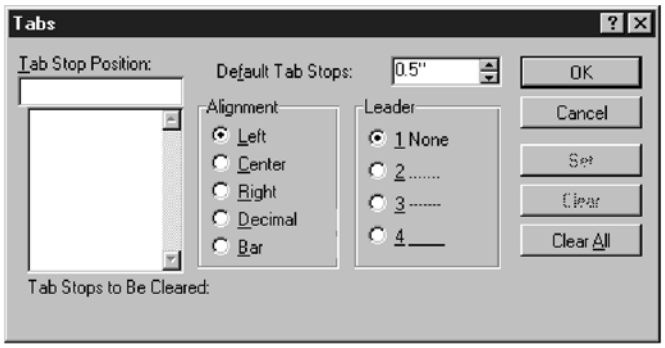

Figure 5.8 Microsoft Windows secondary window.

windows are used to perform supplemental or subordinate tasks, or tasks that are extended in nature. Frequently and occasionally used window components should also be presented in secondary windows. Microsoft Windows possesses several types of secondary windows called *dialog boxes, property sheets, property inspectors, message boxes, palette windows,* and *pop-up* windows.

Guidelines. A secondary window should typically not appear as an entry on the taskbar. Secondary windows obtain or display supplemental information that is usually related to the objects that appear in a primary window.

A secondary window is typically smaller than its associated primary window and smaller than the minimum display resolution. Microsoft recommends not displaying any secondary window larger than 263 dialog units × 263 dialog units. Microsoft defines size and location of user-interface elements not in pixels but in *dialog units* (DLUs), a device-independent unit of measure.

- One horizontal DLU is equal to one-fourth of the average character width for the current system font.
- One vertical DLU is equal to one-eighth of the average character height for the current system font.

These sizes keep the window from becoming too large to display at most resolutions. However, they still provide reasonable space to display supportive information, such as Help information, that applies.

The components of a secondary window are summarized in Table 5.2.

Modal and Modeless

- Modal:
  - Use when interaction with any other window must not be permitted.
  - Use for:
    - Presenting information.
      - For example, messages (sometimes called a message box).

- Receiving user input.
  - For example, data or information (sometimes called a prompt box).
- Asking questions.
  - For example, data, information, or directions (sometimes called a question box).

– Use carefully because it constrains what the user can do.

■ Modeless:
– Use when interaction with other windows must be permitted.
– Use when interaction with other windows must be repeated.

A secondary window can be modal or modeless.

Modal. Most secondary windows will be *modal*. Modal windows will not permit interaction with another window until the current dialog is completed. It remains displayed until the appropriate action is taken, after which it is removed from the screen. Modal dialog boxes typically request critical information or actions that must be reacted to before the dialog can continue. Since modal dialog boxes constrain what the user can do, limit their use to situations in which additional information is required to complete a command or when it is important to prevent any further interaction until satisfying a condition.

Modeless. A *modeless* dialog box permits the user to engage in parallel dialogs. Switching between the box and its associated window is permitted. Other tasks may be performed while a modeless dialog box is displayed, and it may be left on the screen after a response has been made to it. Actions leading to a modeless dialog box can be canceled, causing the box to be removed from the screen.

Use a modeless dialog box when interaction with a primary window or another secondary window must be permitted, for example, during the accessing of the Help function. Also, use a modeless dialog box when interaction with other windows must be repeated; for example, in a word search operation.

Cascading and Unfolding

■ Cascading:
– Purpose:
  - To provide advanced options at a lower level in a complex dialog.

– Guidelines:
  - Provide a command button leading to the next dialog box with a "To a Window" indicator, an ellipsis (. . .).
  - Present the additional dialog box in cascaded form.
  - Provide no more than two cascades in a given path.
  - Do not cover previous critical information.
    – Title Bar.
    – Relevant displayed information.
  - If independent, close the secondary window from which it was opened.

- Unfolding:
  - Purpose:
    - To provide advanced options at the same level in a complex dialog.
  - Guidelines:
    - Provide a command button with an expanding dialog symbol (>>).
    - Expand to right or downward.

Access to additional options can be accomplished by inclusion of a command button that opens another secondary window. These multiple secondary windows needed to complete a task may be presented in two forms, cascading or expanding.

Cascading. A *cascading* window keeps the original window displayed, with the dependent window displayed on top, offset slightly to the right and below the original secondary window. A cascade, illustrated in Figures 5.9 and 5.10, is generally used when advanced options at a lower level in a complex dialog must be presented. An indication that the dialog will be cascading is signaled by an ellipsis placed in the command button used to display the additional dialog box. Because of the confusion that can develop with too many cascades, restrict the number of

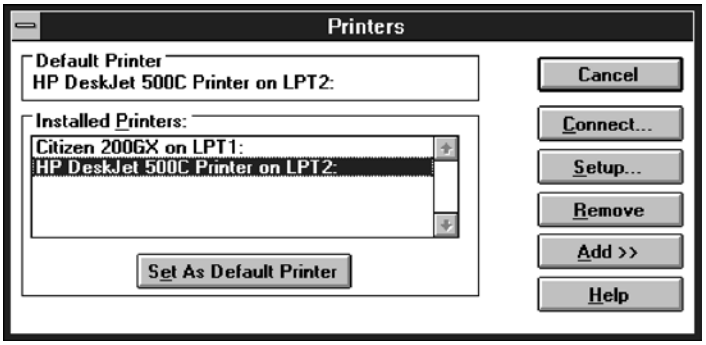

Figure 5.9 Printers secondary window with Connect... cascade button.

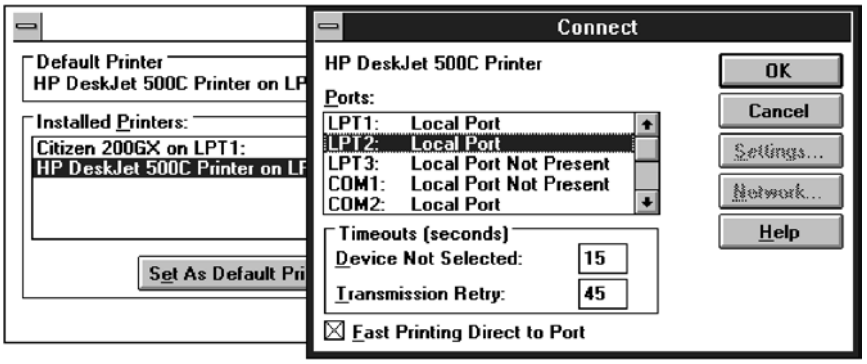

Figure 5.10 Cascading Connect secondary window.

cascades to no more than two in a given path. Do not cover information on the upper-level dialog boxes that may have to be referred to, such as box title bars and other critical or relevant information. If the cascaded window is independent in its operation, close the secondary window from which it was opened and display only the new window.

Unfolding. An *unfolding* secondary window expands to reveal additional options, a form of progressive disclosure. Unfolding windows, sometimes called *expanding* windows, are generally used to provide advanced options at the same level in a complex dialog. They are good alternatives when the interface contains a fixed set of options or controls that seldom need to be accessed. An unfolding window is illustrated in Figures 5.11 and 5.12. An indication that the dialog will be expanding is signaled by a double arrow (>>) placed in the command button used to display

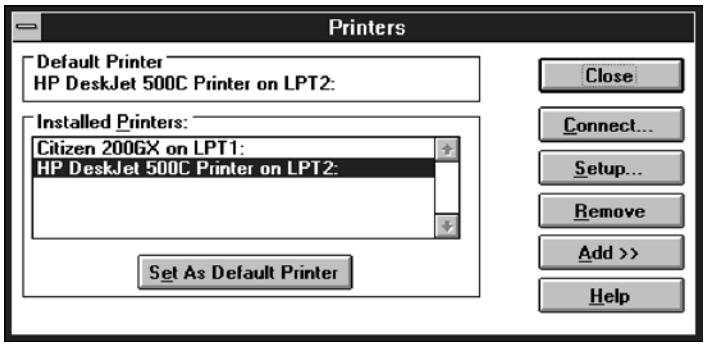

Figure 5.11 Printers secondary window with *Add >>* unfolding button.

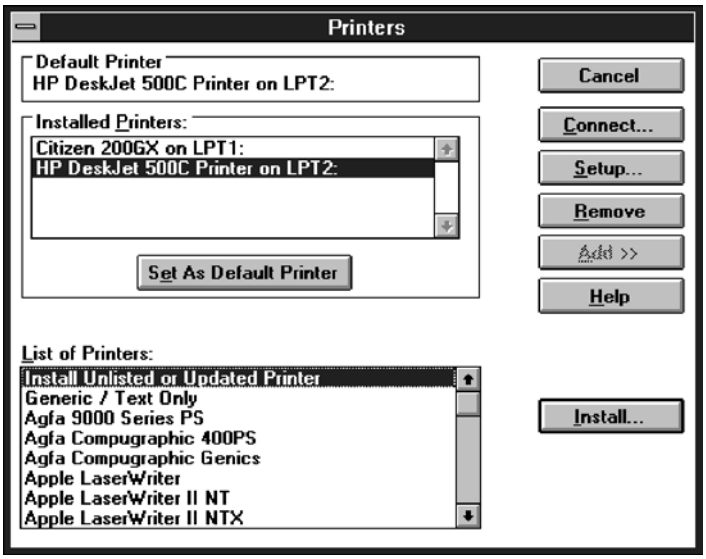

Figure 5.12 Unfolded Printers secondary window.

the additional dialog box. Expand the box to right, preferably, or downward if screen space constraints exist. As an option, the same button can be used to "refold" the additional part of the window.

Dialog Boxes

- Use for presenting brief messages.
- Use for requesting specific, transient actions.
- Use for performing actions that:
  - Take a short time to complete.
  - Are not frequently changed.
- Command buttons to include:
  - OK.
  - Cancel.
  - Others as necessary.

Dialog boxes are used to extend and complete an interaction within a limited context. Dialog boxes are always displayed from another window, either primary or secondary, or another dialog box. They may appear as a result of a command button being activated or a menu choice being selected, or they may be presented automatically by the system when a condition exists that requires the user's attention or additional input. They may possess some basic action controls (Close button and possibly a What's This? button), but do not have a menu bar. A Microsoft Windows dialog box is illustrated in Figure 5.13.

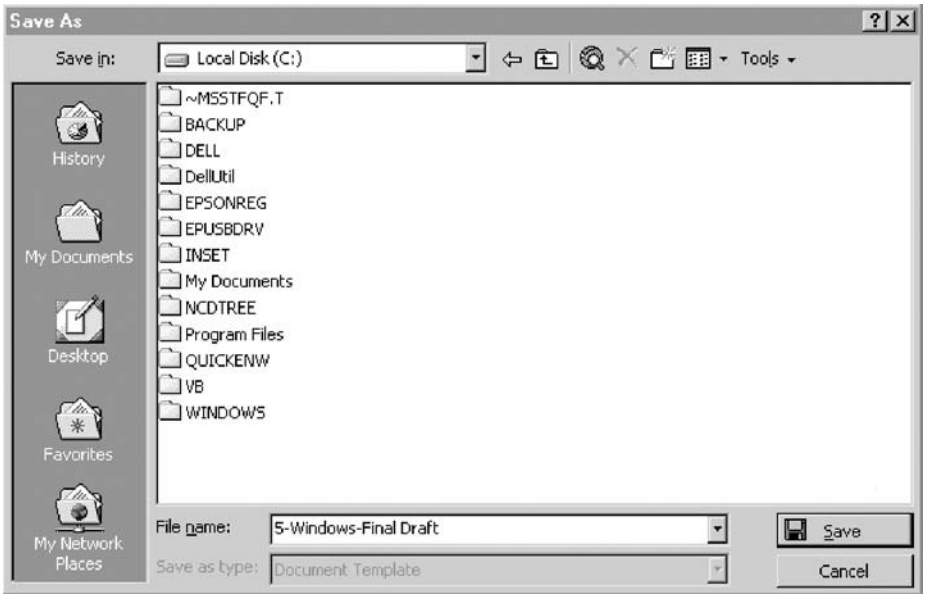

Figure 5.13 Microsoft Windows dialog box.

Most windowing systems provide standard dialog boxes for common functions, some examples being *Open, Save As*, and *Print*. Many platforms also recommend a set of standard command buttons for use in the various kinds of dialog boxes, such as OK, Cancel, and so on. Dialog boxes are of two types, modal and modeless, as recently described. They may also cascade or unfold.

Uses. Dialog boxes are used for presenting brief amounts of information or to request specific transient actions. Dialog box actions will usually be those that do not occur frequently.

Command buttons. Dialog boxes commonly include OK and Cancel command buttons. OK and Cancel buttons work best in dialog boxes that allow the user to set the parameters for a particular command. OK is typically defined as the default command button when the dialog box window opens. Other command buttons may be included in a dialog box in addition to or instead of the OK and Cancel buttons. Follow the design conventions for command buttons found in Step 7.

Property Sheets and Property Inspectors

The properties of an object in an interface can be displayed in a variety of ways. For example, the image and name of an icon on the desktop reflect specific properties of that object, as do other interface components such as toolbars, status bars, and even scroll bars. Secondary windows provide two other techniques for displaying properties, *property sheets* and *property inspectors*.

Property Sheets

- Use for presenting the complete set of properties for an object.
- Categorize and group within property pages, as necessary.
  - Use tabbed property pages for grouping peer-related property sets.
  - The recommended sizes for property sheets are:
    - 252 DLUs wide x 218 DLUs high
    - 227 DLUs wide x 215 DLUs high
    - 212 DLUs wide x 188 DLUs high
  - Command buttons to include:
    - OK.
    - Cancel.
    - Apply.
    - Reset.
    - Others as necessary.
  - For single property sheets, place the commands on the sheet.
  - For tabbed property pages, place the commands outside the tabbed pages.

Use. A property sheet is the most common way to present an object's complete set of properties in a secondary window. A property sheet is a modeless secondary window that displays the user-accessible properties of an object, properties that

may be viewed but not necessarily edited. A single page property sheet is illustrated in Figure 5.14.

Property pages. Because there can be many properties for an object and the object's context, the categorization and grouping of properties within sets may be necessary. A technique for supporting navigation to groups of properties in a property sheet is a tabbed *property page,* where each set of properties is presented within the window as a separate page. Each page tab is labeled with the name of the set, as shown in Figure 5.15. Use tabbed property pages for grouping peer-related property sets, tabs are described in Step 7.

Size. The sizes recommended for property sheets by Microsoft are shown above. These will create a window smaller than its associated window and smaller than the minimum display resolution.

Command buttons. Property sheets typically allow the values for a property to be changed, and then applied. Include the following common command buttons for handling the application of property changes. For common property sheet transaction buttons, use OK, Cancel, and Apply. A Reset command button to cancel pending changes without closing the window can also be included. Other command buttons can be included in property sheets. Avoid including a Help command button. If a Help button seems necessary, the best solution is to simplify the window.

Command buttons on *tabbed property pages* should be located outside of the tabbed page but still within the window. Buttons placed on a page imply that the action being performed applies *only* to that page. Buttons outside the pages imply

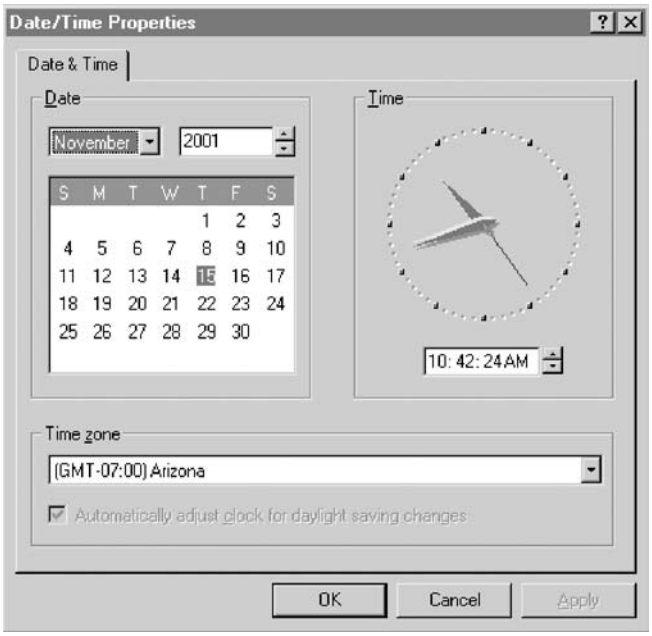

Figure 5.14 Microsoft Windows property sheet.

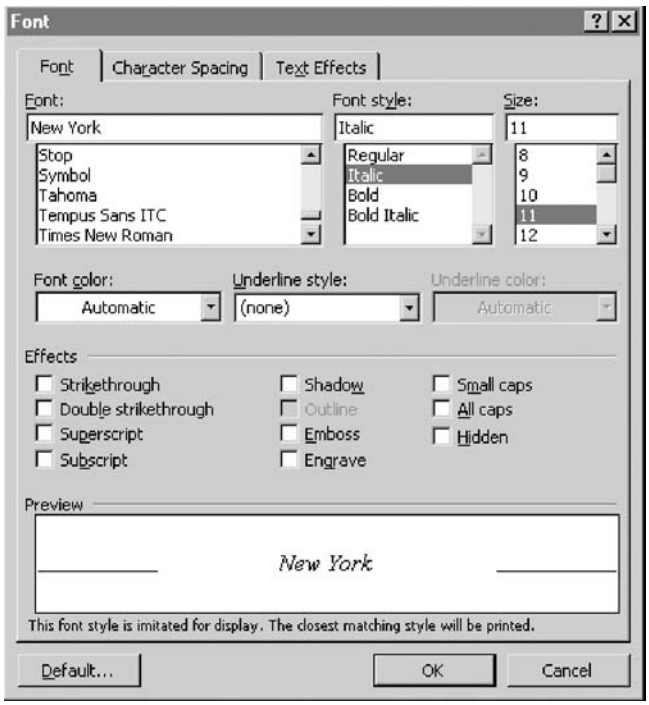

Figure 5.15 Microsoft Windows property sheet tabbed pages.

the action performed applies to *all* pages. This is the desired positioning because, most often, the tabs are considered by the user as simple grouping or navigation techniques. If the properties are to be applied on a page-by-page basis, however, then place the command and buttons on the property pages, and always in the same location on each page. When the user switches pages without selecting a command button, any property value changes for that page are applied. In these situations, it is useful to prompt the user by displaying a message box that asks whether to apply or discard any changes made.

Property Inspectors

- Use for displaying only the most common or frequently accessed object properties.
- Make changes dynamically.

Use. Display only the most common or frequently accessed properties in a property inspector. Properties of an object are displayed by using a dynamic viewer or browser that reflects the properties of the current selection. A property inspector differs from a property sheet. Even when a property sheet window is modeless, the window is typically modal with respect to the object for the properties being displayed. If the user selects another object, the property sheet continues to display

the properties of the original object. A property inspector, on the other hand, always reflects the current selection.

A palette window (described shortly) or a toolbar is used to create a property inspector, as shown in Figure 5.16. An even better alternative is to use a palette window that the user can also configure. Another control in a property inspector can be used to enable the user to display the properties of various objects in the primary window. For example, as the first control in the property inspector, include a drop-down list box that displays the name of the object being viewed. To view another object's properties within the inspector, the object is selected in the drop-down list box.

Dynamic changes. Changes a user makes in a property inspector should be made dynamically. That is, the property value in the selected object should be changed as soon as the user makes the change in the related property control.

Property inspectors and property sheets are not exclusive interfaces. Both can be included in an interface. The most common or frequently accessed properties can be displayed in a property inspector and the complete set in the property sheet. Multiple property inspectors can also be included, each optimized for managing certain types of objects. An interface's behavior can also be changed between that of a property sheet and that of a property inspector. A control can be provided that "locks" its view, making it modal to the current object, rather than tracking the entire selection.

Message Boxes

- Use for displaying a message about a particular situation or condition.
- Command buttons to include:
  - OK.
  - Cancel.
  - Help.
  - Yes and No.
  - Stop.
  - Buttons to correct the action that caused the message box to be displayed.
- Enable the title bar close box only if the message includes a cancel button.
- Designate the most frequent or least destructive option as the default command button.

Use. A message box, as illustrated in Figure 5.17, is a secondary window that displays a message about a particular situation or condition.

Figure 5.16 Microsoft Windows property inspector.

Figure 5.17 Microsoft Windows message box.

Command buttons. Typically, message boxes contain only command buttons with the appropriate responses or choices offered to the user. The command buttons used should allow the message box interaction to be simple and efficient. Microsoft suggests providing the following:

- If a message requires no choices to be made but only acknowledgment, include an OK button and, optionally, a Help button.
- If the message requires the user to make a choice, include a command button for each option.
- Include OK and Cancel buttons only when the user has the option of continuing or stopping the action.
- Use Yes and No buttons when the user must decide how to continue.
- If these choices are too ambiguous, label the command buttons with the names of specific actions, for example, *Save* and *Delete*.

Command buttons to correct the action that caused the message box to be displayed can also be included in a message box. For example, if the message box indicates that the user must switch to another application window to take corrective action, a button that opens that application window can also be included.

Stop. If Cancel is used as a command button in a message box, remember that to users, cancel implies that the state of the process or task that started the message is being restored. If you use Cancel to interrupt a process and the state cannot be restored, use *Stop* instead.

Help. A Help button can be included in a message box for messages needing more detail. This allows the message text to be more succinct.

If other command buttons are needed, consider the potential increase in complexity that their inclusion will cause.

Close box. Enable the title bar Close box only if the message includes a Cancel button. Otherwise, the meaning of the Close operation may be ambiguous.

Default. Designate the most frequent or least destructive option as the default command button.

A definition of message types, and guidelines for writing message, are found in Step 8.

Palette Windows

- Use to present a set of controls.
- Design as resizable.
  - Alternately, design them as fixed in size.

Use. Palette windows are modeless secondary windows that present a set of controls, as shown in Figure 5.18. Palette windows are distinguished by their visual appearance, a collection of images, colors or patterns. The title bar for a palette window is shorter and includes only a *close* button.

Sizing. A palette window can be defined as fixed in size, or, more typically, resizable by the user. Two techniques should indicate when the window is resizable, changing the pointer image to the size pointer, and placing a *size* command in the window's shortcut menu. Preserve the window's size and position so the window can be restored if it, or its associated primary window, is closed and then reopened.

Pop-up Windows

- Use pop-up windows to display:
  - Additional information when an abbreviated form of the information is the main presentation.
  - Textual labels for graphical controls.
  - Context-sensitive Help information.

Pop-up windows can be used to display additional information when an abbreviated form of the information is the main presentation technique, as illustrated in Figure

Other examples of pop-up windows used to display contextual information are *ToolTips* and *balloon tips* that provide the names for controls in graphical toolbars. Pop- up windows are also used to provide context-sensitive Help information. Pop-up win- dows do not contain standard secondary window components such as a title bar and close button.

Figure 5.18 Microsoft Windows palette window.

Figure 5.19 Microsoft Windows pop-up window.

Table 5.2 Microsoft Windows Window Types and Components

PRIMARY WINDOW	
Purpose:	To perform a major interaction.
Components:	Frame or border. Title bar. —Access point for commands that apply to the window, with commands displayed in a pop-up menu. Title Bar icon. —Small version of the icon of the object being viewed. —Access point for commands that apply to the object being displayed in the window, with commands displayed in a pop-up window. Title bar text. Title bar buttons to: close/minimize/maximize /restore a window. Menu bar. Status bar. Scroll bar. Size grip.
SECONDARY WINDOWS	
Purpose:	To obtain or display supplemental information related to the objects in the primary window.
Components:	Frame or border. Title bar. Title bar text. Close button. What's This? button. —Context-sensitive Help about components displayed in the window; this is optional.
Kinds:	Modal and modeless.

(*continues*)

Table 5.2 Continued

SECONDARY WINDOWS	
Dialog Boxes	
Purpose:	To obtain additional information needed to carry out a particular command or task.
Description:	Secondary window. Contains the following common dialog box interfaces: – Open/Replace/Find. – Save As /Print/Print Setup. – Page Setup/Font/Color.
Property Inspectors	
Purpose:	To display the most common or frequently accessed properties of a current selection, usually of a particular type of object.
Description:	A modeless secondary window. Typically modal with respect to the object for which it displays properties.
Usage:	Displayed when requested from selected object.
Property Sheets	
Purpose:	For presenting the complete set of properties for an object.
Description:	A modeless secondary window. Typically modal with respect to the object for which it displays properties.
Usage:	Displayed when requested from selected object.
Message Boxes	
Purpose:	To provide information about a particular situation or condition.
Description:	Secondary window. Types of message boxes: – Information/Warning/Critical.
Palette Windows	
Purpose:	To present a set of controls such as palettes or toolbars.
Description:	Modeless secondary window.

Table 5.2 Continued

SECONDARY WINDOWS	
Pop-Up Windows	
Purpose:	To display additional information when an abbreviated form of the information is the main presentation.
Description:	Secondary window. Does not contain standard secondary window components such as title bar and close button. Example: ToolTip.

Window Management

Microsoft Windows also provides several window management schemes, a *single-document interface, a multiple-document interface, workbooks*, and *projects*. To choose the right scheme to present an application's collection of related tasks or processes, consider a number of design factors, including: the intended users and their skill level, the application and its objects or tasks, and the most effective use of display space. These window management schemes are not exclusive design techniques. They may be combined, or others developed. These techniques, however, are the most frequently used schemes. Each is briefly described below. For greater detail, see Microsoft (2001).

Single-Document Interface

- Description:
  - A single primary window with a set of secondary windows.
- Proper usage:
  - Where object and window have a simple, one-to-one relationship.
  - Where the object's primary presentation or use is as a single unit.
  - To support alternate views with a control that allows the view to be changed.
  - To support simultaneous views by splitting the window into panes.
- Advantages:
  - Most common usage.
  - Window manipulation is easier and less confusing.
  - Data-centered approach.
- Disadvantage:
  - Information is displayed or edited in separate windows.

Often, the window interface can be established using a single primary window. A single-document window design is sufficient when the object's primary presentation or use is as a single unit, such as a folder or document, even when the object contains different types. In a single-document window design, the primary window provides the primary view or work area. Secondary windows can be used for supplemental forms of input, and to view information about objects presented in the primary window.

Multiple-Document Interface

- Description:
  - A technique for managing a set of windows where documents are opened into windows.
  - Contains:
    - A single primary window, called the parent.
    - A set of related document or child windows, each also essentially a primary window.
  - Each child window is constrained to appear only within the parent window.
  - The child windows share the parent window's operational elements.
  - The parent window's elements can be dynamically changed to reflect the requirements of the active child window.
- Proper usage:
  - To present multiple occurrences of an object.
  - To compare data within two or more windows.
  - To present multiple parts of an application.
  - Best suited for viewing homogeneous object types.
  - To clearly segregate the objects and their windows used in a task.
- Advantages:
  - The child windows share the parent window's interface components (menus, toolbars, and status bars), making it a very space-efficient interface.
  - Useful for managing a set of objects.
  - Provides a grouping and focus for a set of activities within the larger environment of the desktop.
- Disadvantages:
  - Reinforces an application as the primary focus.
  - Containment for secondary windows within child windows does not exist, obscuring window relationships and possibly creating confusion.
  - Because the parent window does not actually contain objects, context cannot always be maintained on closing and opening.
  - The relationship between files and their windows is abstract, making an MDI application more challenging for beginning users to learn.
  - Confining child windows to the parent window can be inconvenient or inappropriate for some tasks.
  - The nested nature of child windows may make it difficult for the user to distinguish a child window in a parent window from a primary window that is a peer with the parent window but is positioned on top.

A multiple-document interface (MDI) may be used when multiple views of an object, or multiple documents, must be looked at simultaneously. The purpose of this scheme of windows is to provide multiple views of the same object, to permit comparisons among related objects, and to present multiple parts of an application. An MDI interface consists of multiple document windows that are easy to move between, essentially primary windows constrained to appear only within the parent windows boundary (instead of on the desktop). These windows may be referred to by a name that describes their contents, such as "Main" in Windows Program Manager. When minimized, they are displayed at the bottom of their parent window in iconic form. They are also resizable, movable, and scrollable. The primary window menu bar content may change dynamically, depending on the MDI window with the focus.

MAXIM The information to make a decision *must be there* when the decision is needed.

With MDI, the parent window provides a visual and operational framework for its child windows. Child windows typically share the menu bar of the parent window and can also share other parts of the parent's interface, such as a toolbar or status bar. These components can be changed to reflect the commands and attributes of the child window active at that moment. Secondary windows displayed as a result of interaction within the MDI parent or child window (dialog boxes, message boxes, or property sheets, for example), typically are not contained or clipped by the parent window. These windows should activate and display content according to the conventions for secondary windows associated with a primary window.

If an MDI document is opened, the MDI parent window opens first, and then the child window for the document opens within it. When the user closes the parent window, all of its child windows are closed. Where possible, the state of a child window, such as its size and position within the parent window, should be preserved and restored when the user reopens the file.

Workbooks

- Description:
  - A window or task management technique that consists of a set of views organized like a tabbed notebook.
  - It is based upon the metaphor of a book or notebook.
  - Views of objects are presented as sections within the workbook's primary windows; child windows do not exist.
  - Each section represents a view of data.
  - Tabs can be included and used to navigate between sections.
  - Otherwise, its characteristics and behavior are similar to those of the multiple-document interface with all child windows maximized.
- Proper usage:
  - To manage a set of views of an object.
  - To optimize quick navigation of multiple views.
  - For content where the order of the sections is significant.

- Advantages:
  - Provides a grouping and focus for a set of activities within the larger environment of the desktop.
  - Conserves screen real estate.
  - Provides the greater simplicity of the single-document window interface.
  - Provides greater simplicity by eliminating child window management.
  - Preserves some management capabilities of the multiple-document interface.
- Disadvantage:
  - Cannot present simultaneous views.

A workbook is a scheme for managing a set of views that uses the metaphor of a book or notebook. Within the workbook, views of objects, in the form of sections, are presented within the workbook's primary window, rather than in individual child windows. Tabs are used as a navigational interface to move between different sections. (Tabs are described in Step 7.) Order the tabs to fit the content and organization of the presented information. Each tabbed section represents a view of data. One section can be used to list the workbook's table of contents.

Projects

- Description:
  - A technique that consists of a container: a project window holding a set of objects.
  - The objects being held within the project window can be opened in primary windows that are peers with the project window.
  - Visual containment of the peer windows within the project window is not necessary.
  - Each opened peer window must possess its own menu bar and other interface elements.
  - Each opened peer window can have its own entry on the task bar.
  - When a project window is closed, all the peer windows of objects also close.
  - When the project window is opened, the peer windows of the contained objects are restored to their former positions.
  - Peer windows of a project may be restored without the project window itself being restored.
- Proper usage:
  - To manage a set of objects that do not necessarily need to be contained.
  - When child windows are not to be constrained.
- Advantages:
  - Provides a grouping and focus for a set of activities within the larger environment of the desktop.
  - Preserves some management capabilities of the multiple document interface.
  - Provides the greatest flexibility in the placement and arrangement of windows.

- Disadvantage:
  - Increased complexity due to difficulty in differentiating peer primary windows of the project from windows of other applications.

A project is similar to a multiple-document interface (MDI), but does not visually contain the child windows. Objects represented by icons contained within it can be opened into primary windows that are peers with the parent window. Opened peer windows in the project do not share the menu bar or other areas contained with the parent window. Each opened peer window within the project must possess its own menu bar and other interface elements. (A palette window can be developed for sharing by all project windows, however.) Each peer child window can have its own entry on the taskbar.

Organizing Window Functions

Information and functions must be presented to people when and where they need them. Proper organization and support of tasks by windows will only be derived through a thorough and clear analysis of user tasks.

Window Organization

- Organize windows to support user tasks.
- Support the most common tasks in the most efficient sequence of steps.
- Use primary windows to:
  - Begin an interaction and provide a top-level context for dependent windows.
  - Perform a major interaction.
- Use secondary windows to:
  - Extend the interaction.
  - Obtain or display supplemental information related to the primary window.
- Use dialog boxes for:
  - Infrequently used or needed information.
  - "Nice-to-know" information.

People most often think in terms of tasks, not functions or applications. Windows must be organized to support this thinking. The design goal is to support the most common user tasks in the most efficient manner or fewest steps. Less frequently performed tasks are candidates for less efficiency or more steps.

Mayhew (1992) suggests that poor functional organization usually occurs because of one of, or a combination of, these factors:

- Emphasis on technical ease of implementation rather than proper analysis of user tasks.

- Focus on applications, features, functions, or data types instead of tasks.
- Organization of the design team into applications, with little cross-team communication.
- Blindly mimicking the manual world and carrying over manual inefficiencies to the computer system.

Emphasis on implementation ease puts the needs of the designer before the needs of the customer. Focusing on tasks conforms to the model of how people think. Application orientation imposes an unnatural boundary between functions, and lack of cross-team communication seldom yields consistent task procedures. Mimicking "what is" will never permit the capabilities of the computer system to be properly exploited.

Recommended usages for the various window types are summarized in the above guidelines. These recommendations were discussed more fully earlier in this chapter.

Number of Windows

- Minimize the number of windows needed to accomplish an objective.

A person does not work with windows simply for the joy of working with windows. Windows are a means to an end, a method of accomplishing something. Multiple windows on a display, as discussed elsewhere in this text, can be confusing, can increase the load on the human visual system, or may be too small to effectively present what needs to be contained within them.

Guidelines that appeared in early stages of window evolution concerning the maximum number of windows that a person could deal with were quite generous, a limit of seven or eight being suggested. As experience with windows has increased, these numbers have gradually fallen. One study found the mean number of windows maintained for experienced users was 3.7. Today, based on expressions of window users, a recommendation of displaying no more than two or three at one time seems most realistic. The guidelines on limitations for items like cascades (1–2) reflect today's feelings. The exact number of windows a person can effectively deal with at one time will ultimately depend on both the capabilities of the user and the characteristics of the task. Some users and situations may permit handling of more than three windows; for other users and situations, three windows may be two too many.

The general rule: Minimize the number of windows used to accomplish an objective. Use a single window whenever possible. Consider, however, the user's task. Don't clutter up a single window with rarely used information when it can be placed on a second, infrequently used, window.

Window Operations

Guidelines for windows operations are still evolving. Because of the paucity of published research data, many of the guidelines are still more anecdotal and intuitive than scientific. Guidelines will continue to develop and change as our understanding of, and

experiences with, the windows interface continue to increase. Today, the following operational guidelines seem appropriate.

Active Window

- A window should be made active with as few steps as possible.
- Visually differentiate the active window from other windows.

While a system supports the display of multiple windows, the user generally works within a single window at one. This window, called the active window, may be designated either by the system or the user. Many systems make a window active when it is the object of another windowing operation. It is assumed that if users wish to change one aspect of a window's structure, they also wish to change its contents. They should also be permitted to move to and make any window active with as few steps as possible. This can be accomplished by simply allowing the user to move the selection cursor to the window's interior and then signal by pressing a key or button. For hidden windows, a menu of open windows might be presented from which the user selects a new open window.

In some situations, it may be desirable to allow multiple open windows. One research study compared a single open window with multiple open windows in performing queries and found that multiple open windows were described by people as more "natural." Performance was slower with multiple open windows, however. The study concluded that if user acceptance is important, multiple open windows might be the better alternative. If speed of task handling is critical, a single active window is more desirable.

Visually differentiate the active window from other windows. It is important that the user be able to quickly identify the active window. Methods to do this include a contrasting window title bar, border, or background color. An "active" indicator in the window border, which is turned on or off, may also be used. A combination of two or more of these visual cues may be used as well. The visual cue selected should be of moderate intensity, neither too powerful nor too subtle. Powerful cues will be distracting; subtle cues will be easily overlooked.

General Guidelines

- Design easy to use and learn windowing operations.
  - Direct manipulation seems to be a faster and more intuitive interaction style than indirect manipulation for many windowing operations.
- Minimize the number of window operations necessary to achieve a desired effect.
- Make navigating between windows particularly easy and efficient to do.
- Make the setting up of windows particularly easy to remember.
- In overlapping systems, provide powerful commands for arranging windows on the screen in user-tailorable configurations.

Easy to use. Design easy to use and learn window operations. The complexity of a windowing system should not cancel out its potential advantages. Operations must be carefully designed to achieve simplicity. As has been suggested, the ideal is that window manipulations should occur *implicitly* as a result of the user's task actions, not as a result of explicit, conscious, window management actions.

Minimize number. Minimize the number of window operations needed to achieve a desired effect. Establish the kinds of window operations that people are likely to want, and minimize the number of operations that must be performed to attain these configurations.

Easy navigation. Make navigating between windows easy and efficient. A study found that navigation between windows was the most frequent manipulation activity performed. High-frequency operations should always be easy to do.

Setting up. Make the process of setting up windows easy to remember. A study found that window arrangement (opening, resizing, moving, and so on) was a less frequent activity. Low-frequency operations should always be easy to learn.

User-tailorable configurations. In overlapping systems, provide powerful commands for arranging windows in user-tailorable configurations. When an overlapping window system is used, provide easy operations to achieve desired windowing configurations. Specific configurations should be capable of being created, named, and recalled.

Opening a Window

- Provide an iconic representation or textual list of available windows.
  - If opening with an expansion of an icon, animate the icon expansion.
- When opening a window:
  - Position the opening window in the most forward plane of the screen.
  - Adapt the window to the size and shape of the monitor on which it will be presented.
  - Designate it as the active window.
  - Set it off against a neutral background.
  - Ensure that its title bar is visible.
- When a primary window is opened or restored, position it on top.
  - Restore all secondary windows to the states that existed when the primary window was closed.
- When a dependent secondary window is opened, position it on top of its associated primary window.
  - Position a secondary window with peer windows on top of its peers.
  - Present layered or cascaded windows with any related peer secondary windows.
- When a dependent secondary window is activated, its primary window and related peer windows should also be positioned at the top.
- If more than one object is selected and opened, display each object in a separate window. Designate the last window selected as the active window.

- Display a window in the same state as when it was last accessed.
  - If the task, however, requires a particular sequence of windows, use a fixed or consistent presentation sequence.
- With tiled windows, provide an easy way to resize and move newly opened windows.

Typically, when windows are opened, they are designated as active and positioned in the most forward plane of the screen so that they can be used immediately. When opening a window one should not assume a fixed monitor size, but reflect the size of the monitor on which the window will actually be displayed. The window's title bar must be visible. To focus attention on the newly opened window, display the screen background behind the window in a neutral or subdued manner. When opening windows from an iconic representation, gradually expand the window so that the movement is visible. This will aid association of the icon with the window in the mind of the viewer.

When a primary window is opened or restored, position it at the top. Restore all secondary windows to the states that existed when the primary window was closed. When a dependent secondary window is opened, position it on top of its associated primary window. If a secondary windows has peers, position it on top of its peers. Present layered or cascaded windows with any related peer secondary windows.

When a dependent secondary window is activated, its primary window and related peer windows should also be positioned at the top. If more than one object is selected and opened, display each object in a separate window. Designate the last window selected as the active window. Always display a window in the same state as when it was last accessed. If the task, however, requires a particular sequence of windows, use a fixed or consistent presentation sequence.

The first opened tiled window will consume the entire screen. Subsequent windows are usually positioned by defaults in the system. The system positioning of these subsequent windows may not always be consistent with the user's needs. The system should allow the user to change the default positions, or provide a way for the user to move and resize the system-provided windows easily.

Sizing Windows

- Provide large-enough windows to:
  - Present all relevant and expected information for the task.
  - Avoid hiding important information.
  - Avoid crowding or visual confusion.
  - Minimize the need for scrolling.
    - But use less than the full size of the entire screen.
- If a window is too large, determine:
  - Is all the information needed?
  - Is all the information related?
- Otherwise, make the window as small as possible.
  - Optimum window sizes:

- For text, about 12 lines.
- For alphanumeric information, about seven lines.

Larger windows seem to have these advantages:

- They permit displaying of more information.
- They facilitate learning: Data relationships and groupings are more obvious.
- Less window manipulation requirements exist.
- Breadth is preferred to depth (based on menu research).
- More efficient data validation and data correction can be performed.

Disadvantages include:

- Longer pointer movements are required.
- Windows are more crowded.
- More visual scanning is required.
- Other windows more easily obscure parts of the window.
- It is not as easy to hide inappropriate data.

Always provide large enough windows to present all the relevant and expected information for the task. Never hide important or critical information, and minimize the need for scrolling. A study has found that very small windows requiring a significant amount of scrolling appear to increase decision-making time. Scrolling is also a cumbersome operation. To avoid scrolling, consider using unfolding dialog boxes, cascading windows, or a tab control. Avoid, however, making a window's default size the full size of the display. Doing so leads to any underlying windows being completely hidden from the user's view. The option to maximize primary and secondary windows always exists.

If, through analysis and design, a window appears to be too large, determine:

- Is all the information needed?
- Is all the information related?

Important, critical, or frequently used information must be maintained on a screen, but perhaps information exists that is needed or used infrequently, for example. only 10 to 20 percent of the time. This kind of information is a good candidate for placement on another window or dialog box. Perhaps information is included on a screen that is not related to the task being performed. This kind of information should be located with the information to which it is related. As a last resort, consider shortening some window control captions or other included window text to achieve a proper fit.

At least two studies have looked at optimum window sizes. Procedural text in window sizes of 6, 12, and 24 lines were evaluated by one study. Fastest and most accurate completion occurred with the 12-line window. The retrieval of alphanumeric information was compared in 7-, 13-, and 19-line windows in another study. A seven-line window was found to be more than adequate.

Window Placement

- Considerations:
  - In placing a window on the display, consider:
    - The use of the window.
    - The overall display dimensions.
    - The reason for the window's appearance.
- General:
  - Position the window so it is entirely visible.
  - If the window is being restored, place the window where it last appeared.
  - If the window is new, and a location has not yet been established, place it:
    - At the point of the viewer's attention, usually the location of the pointer or cursor.
    - In a position convenient to navigate to.
    - So that it is not obscuring important or related underlying window information.
  - For multiple windows, give each additional window its own unique and discernible location.
    - A cascading presentation is recommended.
  - In a multiple-monitor configuration, display the secondary window on the same monitor as its primary window.
  - If none of the above location considerations apply, then:
    - Horizontally center a secondary window within its primary window just below the title bar, menu bar, and any docked toolbars.
  - If the user then moves the window, display it at this new location the next time the user opens the window.
    - Adjust it as necessary to the current display configuration.
  - Do not let the user move a window to a position where it cannot be easily repositioned.
- Dialog boxes:
  - If the dialog box relates to the entire system, center it on screen.
  - Keep key information on the underlying screen visible.
  - If one dialog box calls another, make the new one movable whenever possible.

Considerations. In placing a window on the display, consider how the window is used in relation to other windows, the overall dimensions of the display, and the reason that the window is being presented.

General. First, locate the window so it is fully visible. If the window is being restored, locate it where it last appeared. If the window is new and the location has not yet been established, place it at the point of the viewer's attention. This will usually be the location of the pointer or cursor. Also, place the window in a position where it will be convenient to navigate to, and where it will not obscure important underlying screen information. Preferred positions are essentially below and right. The suggested order of placement is below right, below, right, top right, below left, top, left, top left.

In a multiple-monitor configuration, display the secondary window on the same monitor as its primary window. If none of these situations applies, horizontally center a secondary window within the primary window, just below the title bar, menu bar, and any docked toolbars. Give each additional window its own unique location. A cascading presentation, from upper left to lower right is recommended. If the user then moves the window, display it at this new location the next time the user opens the window, adjusted as necessary for the current display configuration. Do not let the user move a window to a position where it cannot be easily repositioned.

Dialog boxes. If a dialog box relates to the entire system, center it on display, keeping key information on an underlying window visible. If one dialog box calls another, make the new one movable whenever possible.

Window Separation

- Crisply, clearly, and pleasingly demarcate a window from the background of the screen on which it appears.
  - Provide a surrounding solid line border for the window.
  - Provide a window background that sets the window off well against the overall screen background.
  - Consider incorporating a drop shadow beneath the window.

Component separation is especially critical in a graphics environment because of the spatial layering that can occur. All windows must be clearly set off from the underlying screen or windows. The demarcation must be crisp and visually pleasing. A solid single-line border is recommended for this purpose. Also provide a window background that sets the window off well against the overall screen background. If color is used, exercise caution and choose compatible colors. (See Step 12 "Choose the Proper Colors.") Another alternative is to use for the window a lighter shade of the color used for the screen background. Changes in the density of shades are often more visually pleasing than different colors. To emphasize the three-dimensional aspects of graphic windows, incorporate a drop shadow beneath each window.

Moving a Window

- Permit the user to change the position of all windows.
- Change the pointer shape to indicate that the move selection is successful.
- Move the entire window as the pointer moves.
  - If it is impossible to move the entire window, move the window outline while leaving the window displayed in its original position.
- Permit the moving of a window without its being active.

An indication that the move operation has been successfully selected and that the move may begin should be indicated to the user by changing the pointer's shape. This

will provide feedback indicating that it is safe to begin the move operation and will avoid false starts. Ideally, the entire window should move along with the pointer. If the entire window cannot be moved, move the window outline while leaving the full window displayed on the screen. Displaying only the window's outline during the move operation, and not the window itself, may make it harder for the user to decide when the window has been repositioned correctly. It may be necessary for a window to be moved without being active. This should be possible. The action of moving the window may implicitly activate it, however.

Resizing a Window

- Permit the user to change the size of primary windows.
  - Unless the information displayed in the window is fixed or cannot be scaled to provide more information.
- Change the pointer shape to indicate that the resizing selection is successful.
- The simplest operation is to anchor the upper-left corner and resize from the lower-right corner.
  - Also permit resizing from any point on the window.
- Show the changing window as the pointer moves.
  - If it is impossible to show the entire window being resized, show the window's outline while leaving the window displayed in its original position.
- When window size changes and content remains the same:
  - Change image size proportionally as window size changes.
- If resizing creates a window or image too small for easy use, do one of the following:
  - Clip (truncate) information arranged in some logical structure or layout when minimum size is attained, or
  - When no layout considerations exist, format (restructure) information as size is reduced, or
  - Remove less useful information (if it can be determined), or
  - When minimum size is attained, replace information with a message that indicates that the minimum size has been reached and that the window must be enlarged to continue working.
- Permit resizing a window without its being active.

Make your primary windows resizable unless the information displayed in the window is fixed or cannot be scaled to provide the user with more information. (For example, a calculator window.)

An indication that the resize operation has been successfully selected, and that the move may begin, should be indicated to the user by changing the pointer's shape. This will provide the necessary feedback that it is safe to begin the resizing and will avoid false starts. The simplest operation for the user, conceptually, is always to resize from the lower-right corner and anchor the window in the upper-left corner. Flexibility can be provided by permitting resizing to occur from any point on the border (the anchor is always opposite the pulling point), but conceptually this is more complex. Some people

may have difficulty predicting which window sides or corners will be resized from specific pulling points.

Ideally, the entire window should move along with the pointer. If the entire window cannot be moved, move the window outline while leaving the full window displayed on the screen. Displaying only the window's outline during the move operation, and not the window itself, may make it harder for the user to decide when the window has been repositioned correctly.

The effect of a resizing operation on the window's contents usually depends on the application. In enlarging, more data may be displayed, a larger image may be created, or blank space may be added around the image. In reducing, less data may be displayed, the image may be made smaller, blank space may be eliminated, or the data may be reformatted. If resizing creates a window or image too small for easy use, clip or truncate information arranged in some logical structure, format, or layout. When no layout considerations exist, as is the case for text, format or restructure the displayed information. Also consider removing less useful information, if it can be determined. When the minimum size has been attained and any additional attempts to reduce window size occur, replace the information with a message that indicates that the minimum size has been reached and that the window must be enlarged to continue working.

It may be necessary for a window to be resized without being active. This should be possible. The action of moving the window may implicitly activate it, however.

Other Operations

- Permit primary windows to be maximized, minimized, and restored.

Maximizing. Maximizing a window increases the size of the window to its largest optimum size. The system default setting for the maximum size is as large as the display. This should be adjustable, as necessary.

Minimizing. Minimizing a window reduces it to its smallest size.

Restoring. Restoring returns a window to its previous size and position after the user has maximized or minimized it.

Window Shuffling

- Window shuffling must be easy to accomplish.

Window shuffling should be easy to perform in as few steps as possible. One method is to permit the toggling of the two most recently displayed windows. Another is to permit rapid window shuffling and the swapping of the front window and the second or back window.

Keyboard Control/Mouseless Operation

- Window actions should be capable of being performed through the keyboard as well as with a mouse.

- Keyboard alternatives should be designated through use of mnemonic codes as much as possible.
- Keyboard designations should be capable of being modified by the user.

All window actions should be capable of being performed using the keyboard as well as the mouse. This will provide a more efficient alternative for applications that contain tasks that are primarily keyboard-oriented, for users skilled in touch typing, and for any other situations in which frequent movement between keyboard and mouse may be required. The use of mnemonic codes to reflect window mouse actions will greatly aid user learning of the keyboard alternatives. To provide the user flexibility, all keyboard designations should be capable of being user modified.

Closing a Window

- Close a window when:
  - The user requests that it be closed.
  - The user performs the action required in the window.
  - The window has no further relevance.
- If a primary window is closed, also close all of its secondary windows.
- When a window is closed, save its current state, including size and position, for use when the window is opened again.

Close a window when the user requests it to be closed, when the action required in the window is performed, or when the window has no further relevance to the task being performed. If the closed window is a primary window, also close its associated secondary windows. When a window is closed, it is important that its current state, including size and position, be saved for use when the window is again opened.

Web Systems

Web systems have limited windowing capabilities. The *frame* concept does provide window-like ability, and JavaScript does provide *pop-up* windows.

Frames

- Description:
  - Multiple Web screen panes that permit the displaying of multiple documents on a page.
  - These documents can be independently viewed, scrolled, and updated.
  - The documents are presented in a tiled format.
- Proper usage:
  - For content expected to change frequently.
  - To allow users to change partial screen content.
  - To permit users to compare multiple pieces of information.

- Guidelines:
  - Use only a few frames (three or less) at a given time.
  - Choose sizes based upon the type of information to be presented.
  - Never force viewers to resize frames to see information.
  - Never use more than one scrolling region on a page.

Description. The Web is, historically, essentially a single page (or, by analogy, a single window) entity. While providing significant interface benefits, it is also a reversal of the interface evolution process that led from single-screen technology to windowing. To counteract this shortcoming, *frames* were created. A frame is an independent pane of information presented in a Web page, or, again by analogy, as multiple windows. Frames, however, are presented as tiled, with no overlapping capability. The interaction richness, support, and contextual cues provided by overlapping windows are lacking. Frames, then, allow the displaying of multiple documents on a single Web page. These multiple documents can be independently viewed, scrolled, and updated.

Proper usage. Frames are useful in situations where portions of the page content are expected to change frequently. The volatile information can be separated from other page content and placed within a frame, thereby requiring only a portion of the page's content to be modified. Frames are also useful for allowing the user to change page content; navigation links can be placed in one frame and the resulting content displayed within another frame. As different links are selected, the content in the related frame changes. Finally, frames more effectively allow users to compare multiple pieces of related information displayed within the different frames.

Advantages and disadvantages. Frames, like most interface entities, have advantages and disadvantages. At this moment in their existence, the disadvantages seem to outweigh the advantages. These disadvantages, however, are being whittled away as Web technology advances.

Frames *advantages* mostly cluster around their ability to reduce the user's content comprehension mental load. These include the following:

- They decrease the user's need to jump back and forth between screens, thereby reducing navigation-related cognitive overhead.
- They increase the user's opportunity to request, view, and compare multiple sources of information.
- They allow content pages to be developed independently of navigation pages.

The *disadvantages* mostly cluster around navigational shortcomings, including:

- The difference between a single Web page and a page with frames is not always obvious to the user.
- They suffer some of the shortcomings of tiled screens:
  - Only a limited number can be displayed in the available screen area.
  - They are perceived as crowded and more visually complex because frame borders are flush against one another and they fill up the whole screen.

Crowding is accentuated if the borders contain scroll bars and/or control icons. Viewer attention may be drawn to the border, not the data.

- Frames-based pages behave differently from regular Web pages.
  - Page-printing difficulties and problems can exist.
  - Page interaction can be clumsy.
  - URLs cannot be e-mailed to other users.
- Frames will not work on older browsers.

Past problems, now being addressed and mostly solved, have included difficulties in bookmarking pages, difficulties in creating browser history lists, and inconsistencies in behavior of the browser's Back button. So, while no longer an interface disaster, frames do make a page clumsy to work with. One study has found that well-designed frames didn't hinder users, but they didn't help, either. A review of the top 100 Web sites (in 1999) found that one common characteristic they all shared was no use of frames.

Guidelines. Guidelines to consider in using frames are the following: Use no more than three frames at a time. Using more will shrink each frame's usable area to the point where little space will be available for presenting content. Then, users will not be able to see much and be forced to scroll. Choose frame sizes based upon the type of information you want to present. Navigational links, for example, should be presented in a small frame and the corresponding information in a larger adjacent frame. Never force people to resize frames to see information. If people feel they must resize frames, the page design is poor. Finally, do not use more than one scrolling region in frames contained on a page. This may be confusing to many users.

Technological advances in frames will continue to occur. Knowledge related to frame usability will also advance. Always be aware of the latest developments.

Pop-Up Windows

- Be extremely cautious in the use of pop-up windows.

JavaScript pop-up windows began appearing on the Web in 1996. Their use is multiplying and, in the view of almost all Web users, polluting screens. Because they are most frequently used in advertising, they have become a source of great aggravation to almost every user. Anecdotal evidence suggests that when a pop-up window begins to appear, most people close them before they are rendered. So, if a pop-up window is used, it may never be completely seen or read by the user. Use them with extreme caution