Motif and CDE Style Guide

Preface

- Audience
- Applicability
- Organization
- Related Documents
- Conventions
- Problem Reporting

The User Interface: Elements of Style

- The Motif User Interface
- Elements of the User Interface
- Element Behavior

Controls

- Displaying or Entering Text
- Displaying Lists
- Using Interactive Controls
- Setting Values
- Containing or Organizing Elements
- Using Window Manager Controls

Input Devices

- Keyboards
- Pointing Devices

Audible and Visual Interface Cues

- Audible Cues
- Visual Cues

Specifying Attributes and Navigating

- Introducing the Window Manager
- Supporting and Designing Windows
- Interacting with the Interface
- Navigating Within the Interface
- Activating Controls
- Changing Values in Controls
- Using Shortcuts

Selection

- Selection Models
- Mouse–Based Selection Techniques
- Keyboard–Based Selection Techniques
- Mouse–Based Selection Modes
- Keyboard–Based Selection Modes
- Complex Scopes

Data Transfer

- Transferring Data
- Using Transfer Techniques

Fundamental Design Principles

- Placing the User in Control of the Interface
- Reducing the User's Memory Load
- Promoting Consistency

Visual Presentation Principles

- Design Methodology
- Visual Variables
- Visual Priority

Visual Design Tasks

Application Development Principles

Developing a Menu Structure

Creating Windows

Designing Controls Within Windows

Designing for User Interaction

Developing an Object–Oriented Interface

International Design Guidelines

General Guidelines for International Design

Bidirectional Language Support

Vertical Language Support

Designing for Accessibility

Standard Accessibility Guidelines

Types of Disability

Existing Keyboard Access Features

Resources for More Information on Accessibility

Common Desktop Environment Guidelines

Advantages of a Common User Interface

Relationship of CDE to Motif

CDE Visual Design

CDE Application Design Guidelines

CDE Window and Session Control

CDE Application Messages

Drag and Drop

Widgets and Components Cross Reference

Keyboard Model and Key Bindings

Model Keyboard

Keyboard Function

Mouse Techniques

Mouse Model and Buttons

Glossary

Preface

The Style Guide provides developers who design and implement new products with a framework of behavior specifications that is consistent with the Motif and Common Desktop Environment (CDE) user interface. This behavior is established by drawing out the common elements from a variety of current behavioral models.

For specific details on coding an application program, widget, or window manager, refer to Related Documents .

Audience

This document is written for five audiences:

Application designers

Any designer who uses Motif as a basis for building an application

Widget designers

Any designer who adds functionality to the existing Motif software

User interface system designers

Any designer who designs an interface for users with specific cultural and accessibility needs

Window manager designers

Any designer who uses Motif to govern the layout and controls that affect application windows

Common Desktop Environment application designers

Any designer who builds an application for the Common Desktop Environment

Read through the entire Style Guide once to familiarize yourself with all user interface design concepts. If you are already familiar with graphical user interfaces and their design concepts, but need specific information, skip to the topic you are interested in by using the following table:

If you want to...	**Read...**
Incorporate Motif interface design guidelines into new or existing applications	The User Interface: Elements of Style ; Controls ; Audible and Visual Interface Cues ; Specifying Attributes and Navigating ; Selection ; and Data Transfer .
Learn more about the parts of the user interface	The User Interface: Elements of Style and Controls .
Learn how to interact with the user interface	Input Devices ; Audible and Visual Interface Cues ; Specifying Attributes and Navigating ; Selection ; and Data Transfer .
Locate definitions of terms	The Glossary .
Understand the guidelines for designing a user interface	Style Guide Reference.
Review the keyboard and functions of keys	Keyboard Model and Key Bindings .
Compare mouse mappings for mouse devices with varying number of buttons	Mouse Techniques .
Compare Motif widgets to controls	Widgets and Components Cross Reference .
Know the guidelines required for style certification	Style Guide Certification Checklist .

Applicability

This is revision 2.1 of this document. It applies to Version 2.1 of the Motif software system and Version 2.1 of the Common Desktop Environment system.

Organization

This document is organized into thirteen chapters and three appendixes.

- The User Interface: Elements of Style provides an introduction to the Motif and CDE interfaces, a history of Motif, and an introduction to elements that Motif and this style guide use.
- Controls describes standard window and window manager controls.
- Input Devices describes keyboard and mouse input models.
- Audible and Visual Interface Cues describes audible and visual cues.
- Specifying Attributes and Navigating describes navigation and activation within an application or interface.
- Selection describes keyboard–based and mouse–based selection techniques and selection modes.
- Data Transfer describes user interactions and techniques for transferring data.
- Fundamental Design Principles discusses fundamental design and style issues for a Motif–based application or interface.
- Visual Presentation Principles discusses visual guidelines for presenting a Motif–based application or interface.
- Application Development Principles discusses guidelines for menus, windows, controls, and interface development.
- International Design Guidelines discusses internationalization and localization concepts and issues as they relate to user interface design.
- Designing for Accessibility discusses guidelines for making software applications accessible to users with disabilities.
- Common Desktop Environment Guidelines discusses guidelines for developing an application in the Common Desktop Environment.
- Widgets and Components Cross Reference shows the relationship between the Motif widgets and components that this guide describes.
- Keyboard Model and Key Bindings shows the model keyboard functions and key bindings.
- Mouse Techniques shows mouse functions and button–binding variations.

Related Documents

For more information on Motif and CDE style, refer to the following documents:

- Motif and Common Desktop Environment: Style Guide Reference
- Motif and Common Desktop Environment: Style Guide Certification Checklist

For additional information about Motif, refer to the following documents:

- Motif User's Guide
- Motif Widget Writer's Guide
- Motif Programmer's Guide
- Motif Programmer's Reference
- Motif Release Notes

For additional information about the Common Desktop Environment, refer to the following documents:

- Common Desktop Environment: User's Guide
- Common Desktop Environment: Advanced User's Guide
- Common Desktop Environment: Information Manager Users Guide
- Common Desktop Environment: Application Builder User's Guide
- Common Desktop Environment: ToolTalk Messaging Overview
- Common Desktop Environment: Programmer's Guide
- Common Desktop Environment: Help System Author's and Programmer's Guide
- Common Desktop Environment: Internationalization Programmer's Guide
- Common Desktop Environment: Information System Author's and Programmer's Guide

Conventions

This document uses the following conventions:

Typeface or Symbol	Meaning	Example
`Enter`	A key on the keyboard	When the user presses `Delete`, delete the text.
SELECT, ADJUST, TRANSFER, MENU	A virtual mouse button	The SELECT button selects text.
MB1, MB2, MB3	A physical mouse button (where MB1 is the leftmost button)	Make MB1 the SELECT button.
Push button	A new term	A *push button* is a control that displays a label or graphic that represents an action.
Control	A reference page	For more information, see the Control reference page.
ls	A command	Use **ls –a** to list all files.

Problem Reporting

If you have any problems with the software or documentation, contact your software vendor's customer service department.

The User Interface: Elements of Style

The user interface is the area where the user and an application interact. Any tool or machine has a user interface that is designed according to the user's needs.

A user interface can be as simple as a set of buttons, like those on a telephone or video recorder. In the computer world, a user interface can include a keyboard, a pointing device, and the items that appear on a display screen. The user interface, then, is the ensemble of hardware and software that lets a user and a computer communicate.

The Motif User Interface

The Motif user interface is a graphical user interface for the X Window system. By providing a consistent interface for the UNIX environment, Motif enables users to work with multiple applications that have consistent and similar characteristics.

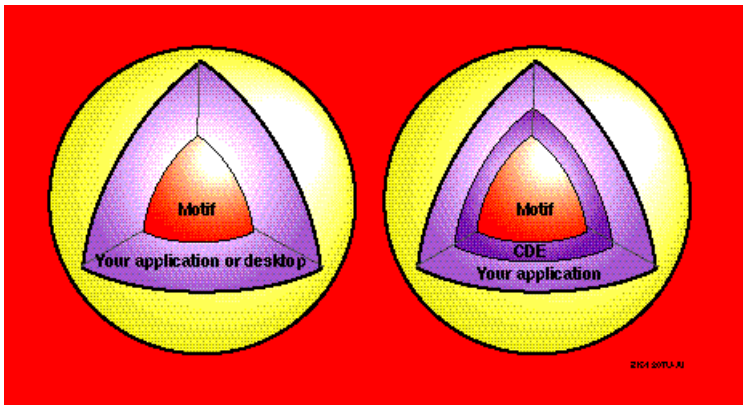
Motif and the Common Desktop Environment

The Common Desktop Environment (CDE) is a specification and reference implementation of a common UNIX desktop user environment and has many cross−process relationships. By comparison, the process relationship between an application, a window manager, and the X Window System server is much simpler than in CDE.

The area covered by CDE is broad, and although the layering in the system is not as rigorous as that of Motif, the relationship between high −level system components is more diverse. This is primarily due to the ToolTalk extensions that have been built into CDE. (For more information on ToolTalk, see the ToolTalk User's Guide and CDE ToolTalk Messaging Overview .)

CDE, however, uses Motif as its underlying toolkit. Applications developed for CDE must adhere to the same style guidelines as Motif applications running under another window manager as well as to CDE−specific style guidelines (see Common Desktop Environment Guidelines for more information). Relationship of Applications With and Without CDE shows the relationship of applications with and without CDE.

Figure 1 Relationship of Applications With and Without CDE



History of Motif

Motif was developed to fill a growing need for a consistent graphical user interface for the X Window system. To develop such an interface, the Open Software Foundation (OSF) solicited technical input from the computer industry in 1989. History of Motif and the Motif Style Guide outlines the history of Motif and the Motif Style Guide.

Table 1 History of Motif and the Motif Style Guide

Version	Description
Motif 1.0 and 1.1	Early releases of Motif. OSF published a style guide.
Motif 1.2 − 1.2.3	Many technical and behavioral upgrades. Technical upgrades included toolkit enhancement, support for X11R5, additional internationalization functions, and performance improvements. Behavioral upgrades included drag−and−drop capability, tear−off menus, and window manager implementation. OSF also improved the style guide.
Motif 1.2.5	Based on Motif 1.2.3. This version was developed exclusively for the Common Desktop Environment (CDE) Version 1.0. A style guide was released with CDE.
Motif 2.0	Based on Motif 1.2.3. This version included additional controls, C++

	support, international language improvements, and text input modifications. A preliminary style guide, based on IBM's Common User Access, was developed but never released.
Motif 2.1	The current version of Motif. Now administered by the X Consortium, this release of Motif combines all the features of Motif 1.2.5 and Motif 2.0. Developers can build their own desktop environment and applications or build CDE 2.1 applications with the same set of libraries. This style guide includes both Motif 2.1 and CDE 2.1 style guidelines. Since CDE 2.1 was built with Motif 2.1, the Motif 2.1 guidelines also apply to CDE 2.1 unless stated otherwise.

Elements of the User Interface

An *element* is a distinguishable part of the user interface that may or may not contain subelements. The elements in a user interface are:

- Data
- Graphic
- Object
- Icon
- Control
- Choice
- Composite
- Workspace
- Window

The following sections describe each of these elements.

Data

Data is an element whose visual representation denotes its value. For example, text is data that is denoted by readable numbers and letters; a binary file is denoted by 0s and 1s.

Graphic

A *graphic* is an element that appears as an image. Graphic elements use a unique format that an application recognizes. XBM, GIF, and JPEG are common graphic formats.

Object

An *object* is an element that visually represents items that have behavior and contents not revealed solely by their visual representation. Objects are often represented as icons, but could be list elements as well. An object generally has a pop−up menu associated with it through which its contents or behavior are accessed.

Icon

An *icon* is an object represented as a graphic, often with an associated label.

Control

A *control* is an element whose behavior is predefined and that is generally supported directly by the Motif toolkit implementation. Types of controls include spring−loaded and tear−off controls.

Choice

A *choice* is a control that is selected by using the SELECT button (or keyboard) to initiate a user action or to toggle a value. There are three kinds of choices: action choices, modal choices, and toggle choices.

Composite Element

A *composite element* includes one or more of the following:

- Elements for packaging and tying together behavior
- Elements for controlling a view
- Elements for implementing selection scopes

Workspace

A *workspace* is a container where all elements reside. Workspaces generally contain the following elements:

- Windows (both primary and secondary)
- Window icons represented by graphic elements with associated labels
- Optional window icon boxes
- Other controls and objects defined by the workspace model, such as objects represented by icons

Window

A *window* is an element on the workspace that presents a view of an object or conducts a dialog with the user. Windows are used to present objects, messages, or menus, or to prompt for information. The two types of windows are primary windows and secondary windows.

Element Behavior

Users can perform unique actions with elements. That is, elements have behaviors associated with them. Elements and Their Behaviors lists some common element behaviors.

Table 1 Elements and Their Behaviors

Element Behavior	Description
Selected	An element that can be added or removed from the selection scope that encloses it
Edited	An element whose displayed value or contents can be changed through user interaction
Static	A data element whose value cannot be changed by the user
Visual	A static element that cannot be selected or activated

Controls

This chapter discusses standard window controls. A *control* is a visually recognizable element or group of elements available for end−user interaction. Controls allow the user to manipulate data in a well−defined manner. The following list categorizes the standard controls by their function:

- Displaying and entering text
- Displaying lists
- Using interactive controls
- Setting values
- Containing or organizing elements
- Using window manager controls

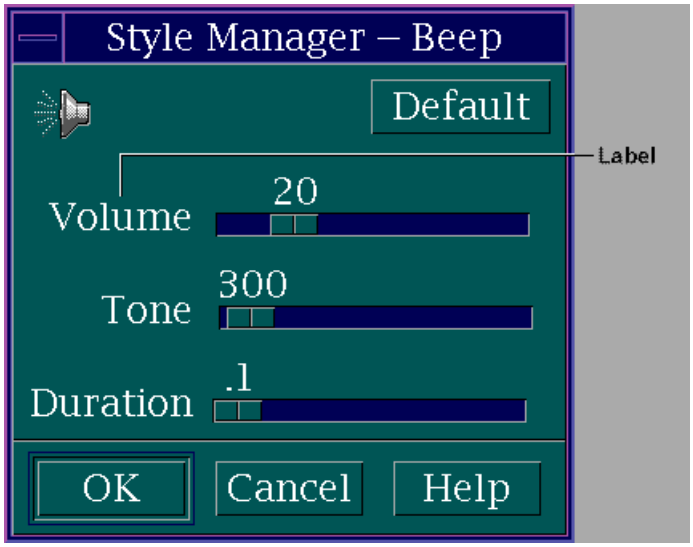
Displaying or Entering Text

Users interact with the following text controls:

- Label
- Text–entry field
- Text–display field

Label

A *label* displays data, but does not allow the user to change it. In some environments, a label might be called a static text field, read–only message, or a read–only text field. An example of a label would be an age that is calculated from a birth date.



For more information, see the Label reference page.

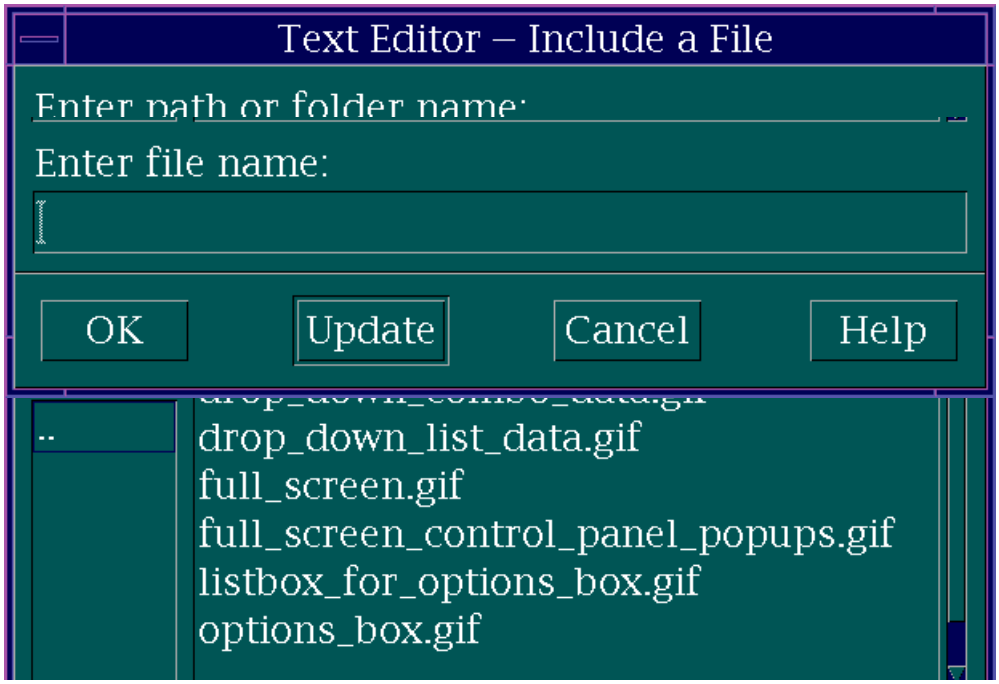
Text–Entry Field

A *text–entry field,* sometimes known as a *''text field''* or *''entry field,''* is an area where the user enters alphanumeric text within a specific boundary. The user can type one or more lines in a text–entry field, depending upon the information that the object or application needs. Multiline text can have both horizontal and vertical scroll bars for moving the text up, down, and across the field.

For more information, see the Text−Entry Field (Control) reference page.

Text−Display Field

The user uses the *text−display field* to display or select only text; the user cannot edit the text. There are single−line and multiple−line variants. Multiline text can have both horizontal and vertical scroll bars for scrolling the text up, down, and across the field.



For more information, see the Text–Display Field (Control) reference page.

Displaying Lists

The following controls display elements of a list:

- List box
- Drop–down list box
- Combination box
- Drop–down combination box
- Selection box
- Spin box
- Option menu

List Box

A *list box* displays a fixed or variable list of objects or value choices. The user can scroll a list box that contains more items than can be displayed at one time within the border of the list box. Your application may allow the user to select only one or more than one item from the list.

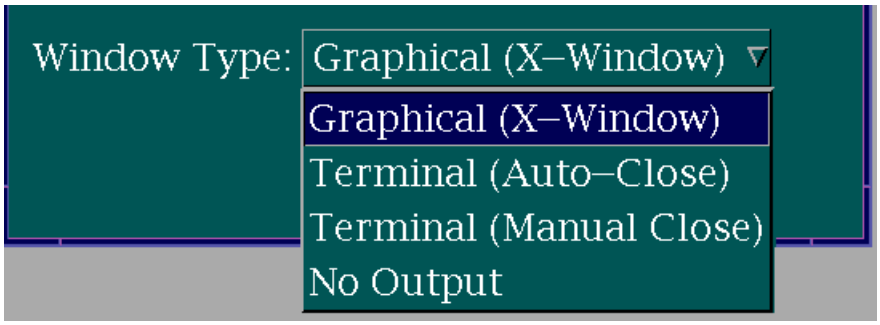
For more information, see the List Box (Control) reference page.

Drop–Down List Box

A *drop–down list box* is a variation of a list box. Only one item in the list is displayed until the user takes an action to display the rest of the list.

The drop–down list box contains a *list cascade button*, a button that contains a down–arrow graphic that presents the cascaded list of items.

You should use a drop–down list box when a list box would take up too much space within the window in which it is displayed. The drop–down list box appears only when the user chooses the list cascade button.

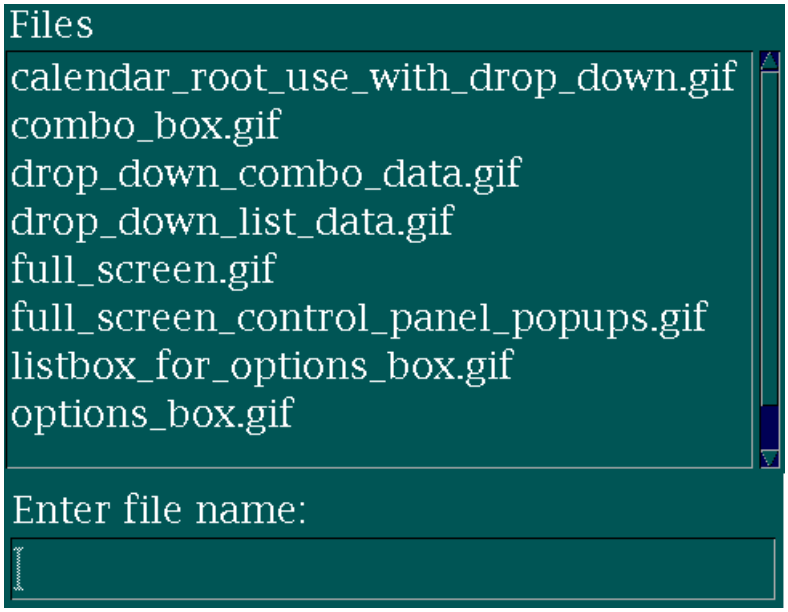


For more information, see the Drop–Down List (Control) reference page.

Combination Box

A *combination box* combines a text–entry field with a list box. The user can select an item from the list or type data directly into the text–entry field. When the user selects an item from the list, that element appears in the text–entry field. The user can select only one item from the combination box.

You should use a combination box when there is an indeterminate number of items to be presented in a list box. Your application can fill the list box portion of the combination box with a set of standard choices for the user to select, while still allowing the user to enter a choice if it does not appear in the list.

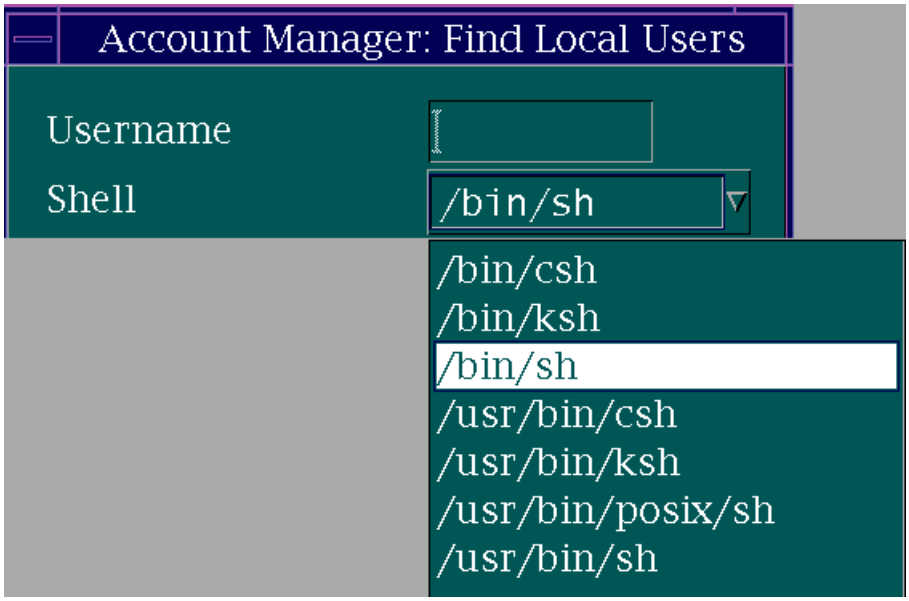


For more information, see the Combination Box (Control) reference page.

Drop–Down Combination Box

A *drop–down combination box* is a variation of a combination box. It displays only the text–entry field portion until the user takes an action to display the list box portion of the control. The user can select only one item from the drop–down combination box.

You should use a drop–down combination box when a combination box would take up too much space within the window in which it is displayed. The drop–down list box appears only when the user chooses the list cascade button.

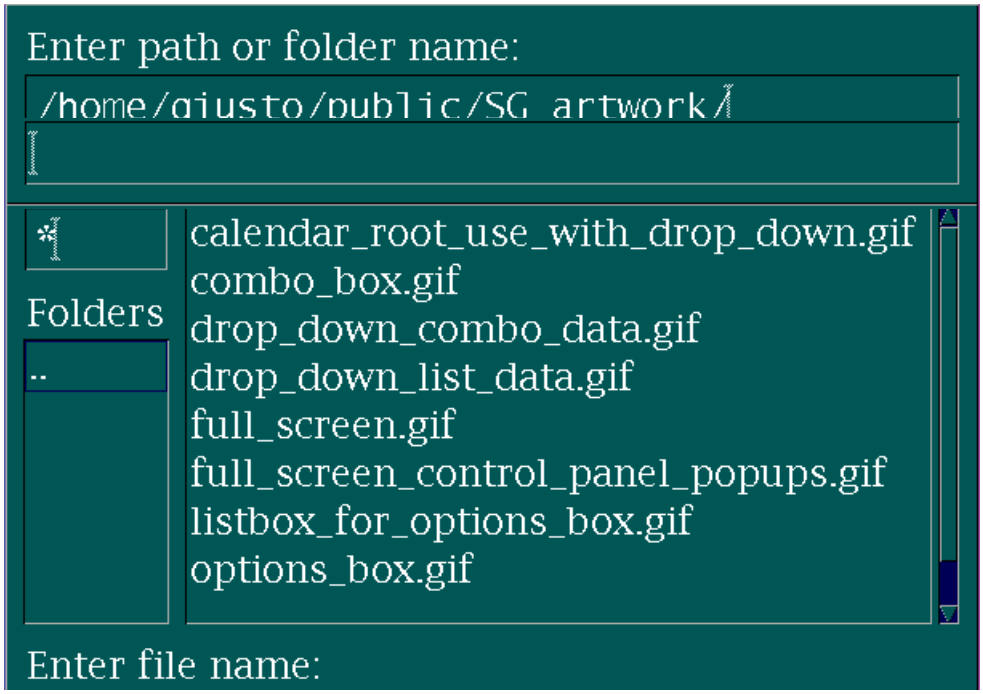


For more information, see the Drop–Down Combination Box (Control) reference page.

Selection Box

A *selection box* allows the user to make a single selection from a list of elements.

The selection box, like a combination box, combines a text–entry field with a list box. In addition, it contains push buttons, for example, to allow the user to confirm the value entered.



For more information, see the Selection Box (Control) reference page.

Spin Box

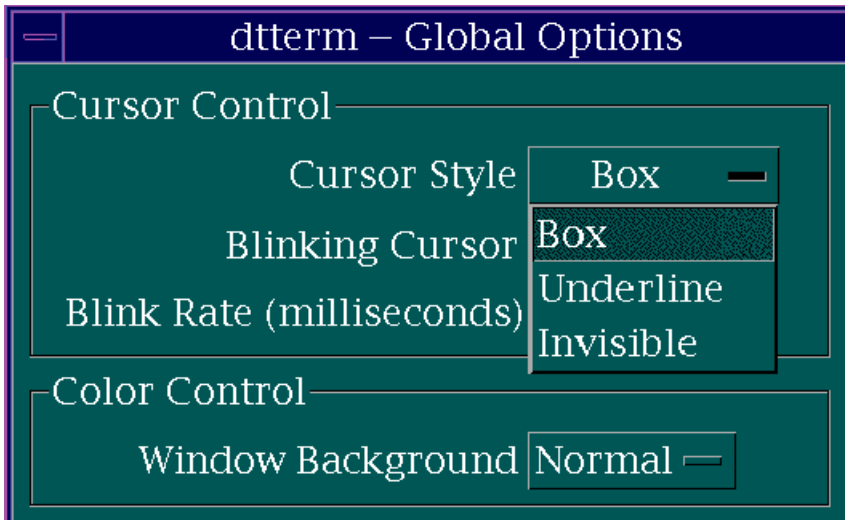
A *spin box* displays related but mutually exclusive choices that have a natural sequence. Unlike other list controls, the spin box operates as though the top and bottom of the list are joined. The user cycles through the choices as if the fields were on a spinning cylinder. A spin box can control multiple fields by using a single set of spin arrows to change choices in each field. A spin box that contains a single field is sometimes called a *spin button*.

The user can also type a choice directly in the spin box field instead of using the spin arrow to change a value.

For more information, see the Spin Box (Control) and Text–Entry Field (Control) reference pages.

Option Menu

An *option menu* is a menu that contains only value and cascading choices. To display an option menu, the user can use an option menu cascade button or choose an item from a cascading list within another option menu.



For more information, see the Option Menu (Menu Type) reference page.

Using Interactive Controls

The following controls present and organize actions:

- Push button
- Menu cascade button
- Menu bar
- Pull–down menu
- Cascaded menu
- Pop–up menu
- Tear–off menu
- Command box

Push Button

A *push button* is a control that displays a label or graphic that represents an action. When the user activates a push button, that action executes immediately.

You should use a push button when you want an action to occur quickly and easily.

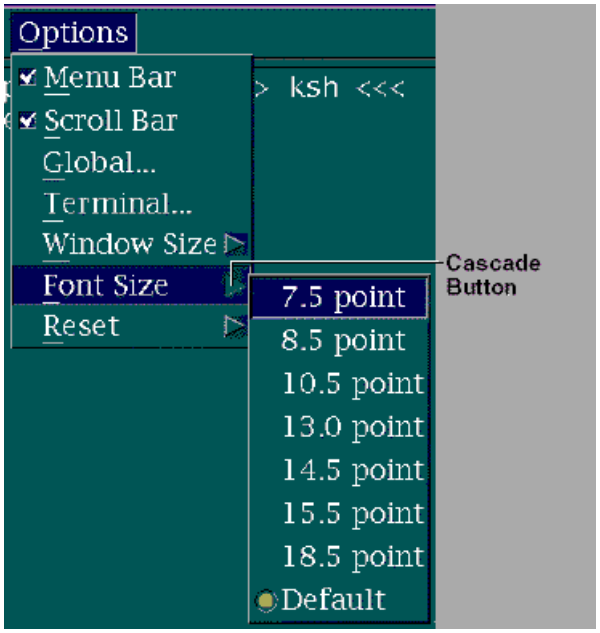
Examples of push buttons include:

- An action choice such as Cancel
- A dialog choice such as Find

For more information, see the Push Button (Control) and Push Button (Predefined) reference pages.

Menu Cascade Button

A *menu cascade button* is a control that represents cascading choices from which the user can display a pull–down menu.



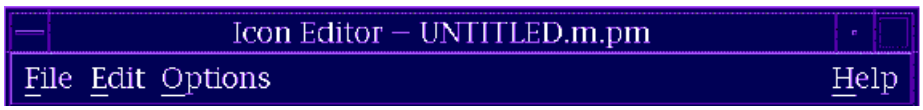
For more information, see the Menu Cascade Button (Control) reference page.

Menu Bar

A *menu bar* appears across the top of a window, just below the title bar. It contains a horizontal list of cascading choices called *menu−bar items*. When the user chooses a menu−bar item, an associated pull−down menu appears.

The name of each menu−bar item indicates choices on the associated pull−down menu. For example, the associated pull−down menu for the Help menu−bar item could have the following choices:

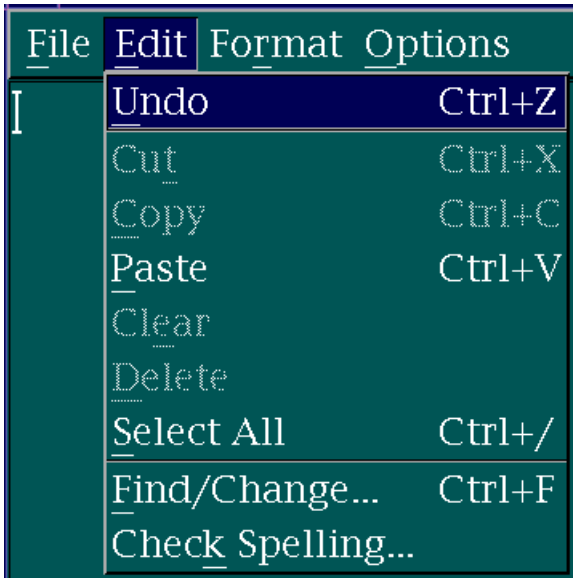
- Contents
- Search for...
- On Help
- About...



For more information, see the Menu Bar (Menu Type) reference page.

Pull−Down Menu

A *pull−down menu* appears when a user chooses a menu−bar item, window menu button, or a menu cascade button.

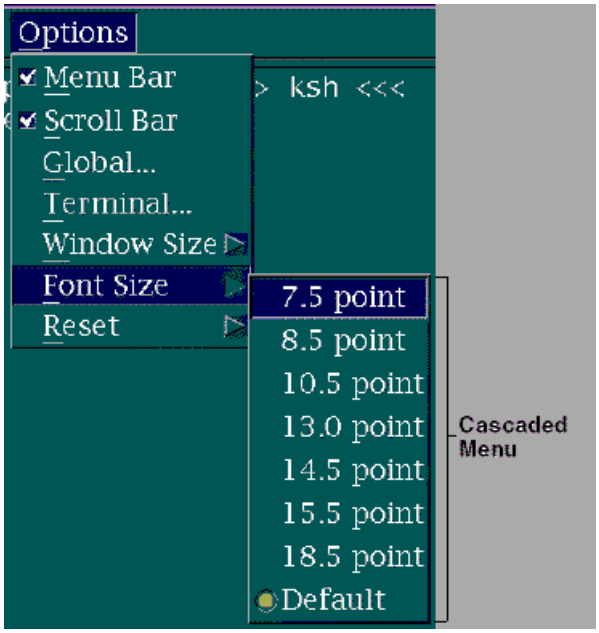


For more information, see the Cascading (Choice Type) and Pull–Down Menu (Menu Type) reference pages.

Cascaded Menu

A *cascaded menu* appears next to, and contains choices related to, a cascading choice in another menu. It is not visible until the user makes a cascading choice. A cascaded menu contains choices that modify or are related to the cascading choice.

Cascaded menus allow you to layer choices so that the user can have access to a wide range of functions without using long pull–down or pop –up menus.



For more information, see the Cascading (Choice Type) reference page.

Pop–Up Menu

A *pop–up menu* provides choices specific to an element. A pop–up menu is not visible until the user invokes it. Only choices that are currently valid appear in a pop–up menu.

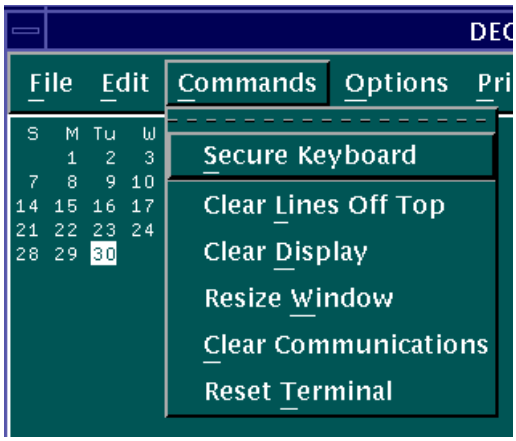
Pop–up menus provide the user with convenient access to menus. The user does not need to move the mouse pointer to the menu bar nor does the user see unavailable choices.

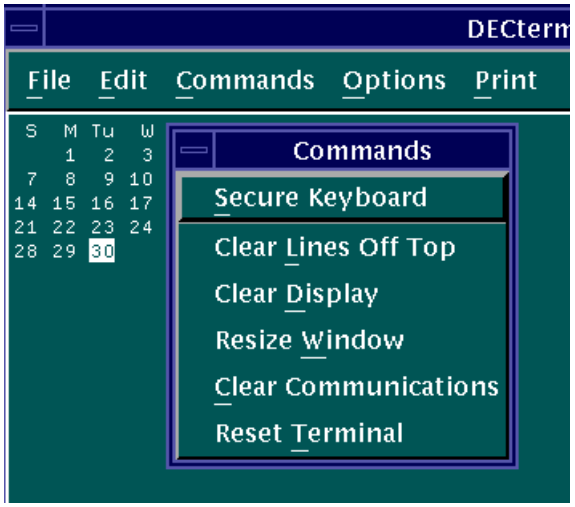
For more information, see the Pop–Up Menu (Menu Type) reference page.

Tear–Off Menu

A *tear–off menu* is a copy of a pull–down, pop–up, or cascaded menu that the user has "torn off" and placed in a separate window. The tear–off menu always contains the same choices as the original menu, with the exception of the tear–off choice.

For example, the user can *''tear off''* a frequently used menu and position it in a convenient place on the screen.

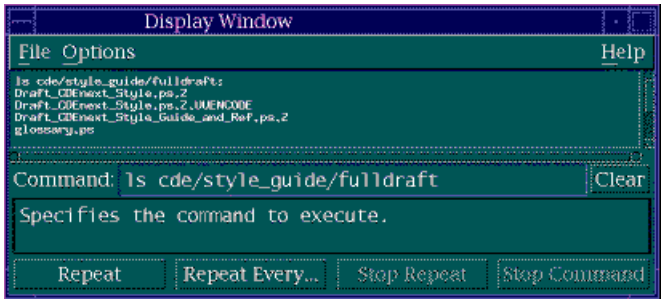




For more information, see the Tear–Off Menu (Menu Type) reference page.

Command Box

A *command box* allows the user to review previous commands and to either choose to reissue a previous command or to issue a new command.



For more information, see the Command Box (Control) reference page.

Setting Values

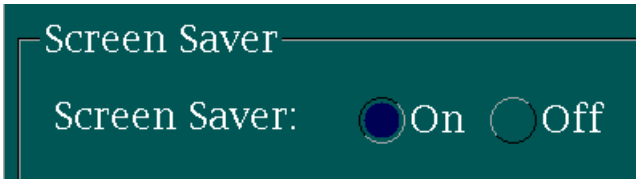
The following controls allow the user to set values other than entering text:

- Radio button
- Check box
- Slider
- Value set

Radio Button

The user presses a *radio button* to display mutually exclusive value choices that have an on or off state. Within a group, the user can turn on at most one radio button at a time. When the user chooses a radio button, any previously chosen radio button within the group turns off.

You should use a radio button when you want the user to make a mutually exclusive choice, for example, to choose among available pen widths in a drawing program.



For more information, see the Properties (Choice) and Radio Button (Control) reference pages.

Check Box

A *check box* has two states: on and off. Occasionally there is an indeterminate third state. A check box is sometimes called a check button.

You should use a check box when you want to display choices that have two clearly discernible states and, possibly, an indeterminate third state.

The check box appears in the indeterminate state when its value reflects a property of selected data that is not completely in either state. For example, if the user selects a section of text where some characters are bold and some are not, a bold check box will show an intermediate state. The user can reset the check box to remove the bold, check the box to make all characters bold, or leave the check box in the indeterminate state to leave the text as is.

For more information, see the Check Box (Control) reference page.

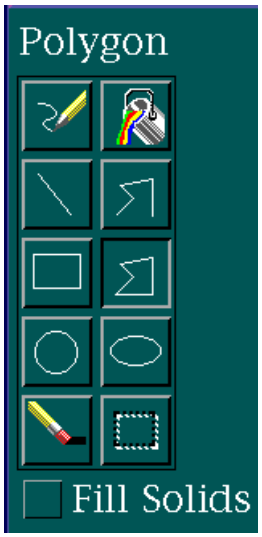
Slider

A *slider* is a control that represents a bounded value. A slider typically shows a scale marked in equal units from which the user can select a particular value from within a slider's boundary.

For more information, see the Slider (Control) reference page.

Value Set

A *value set* is a matrix of mutually exclusive choices that are represented graphically. The value set is similar in concept to a group of radio buttons and primarily presents graphical representations of choices, for example, tools in a palette.



For more information, see the Value Set (Control) reference page.

Containing or Organizing Elements

The following controls contain or organize elements:

- Container
- Group box
- Notebook
- Paned box
- Split bar
- Scroll bar

Container

A *container* displays objects. For example, when the user opens a calendar container, it might contain icons for representing months.

For more information, see the Container (Control) reference page.

Group Box

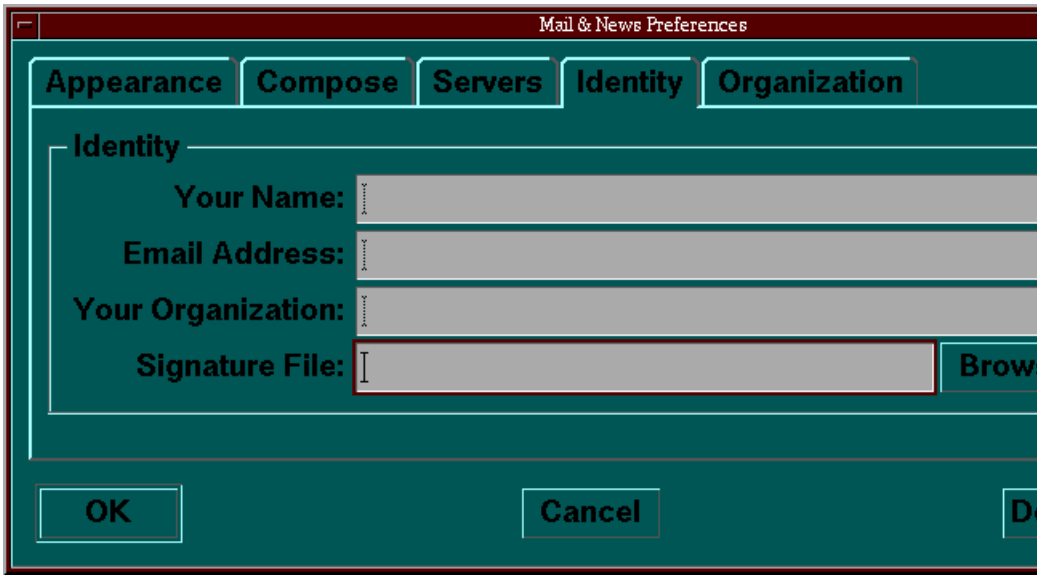
A *group box* groups controls together visually. Group boxes help the user to understand when a group of controls belong together, for example, the radio buttons used to set a text format.

For more information, see the Group Box reference page.

Notebook

A *notebook* groups data. It visually resembles a bound notebook that contains pages separated into sections by tabbed divider pages. A user can *''turn''* individual pages of the notebook or choose tabs to move from one entire section to another.

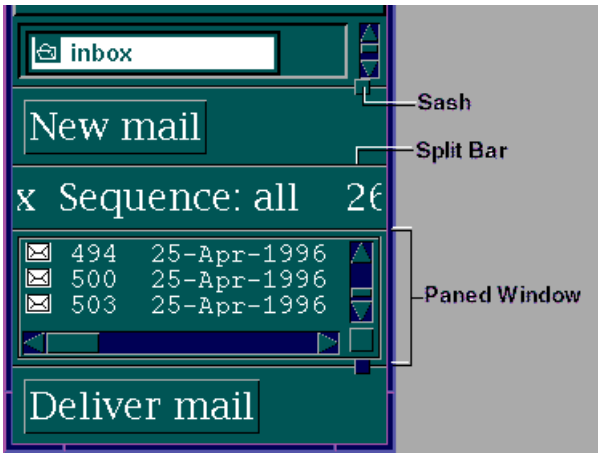
Users can use notebooks to organize information that they would expect to find in a real (hardcopy) notebook. Examples include an address book, an alphabetized index, or a clip art library.



For more information, see the Notebook (Control) reference page.

Paned Box

A *paned box* is a control that the user can divide into panes by using split bars. Each pane has one or more controls.

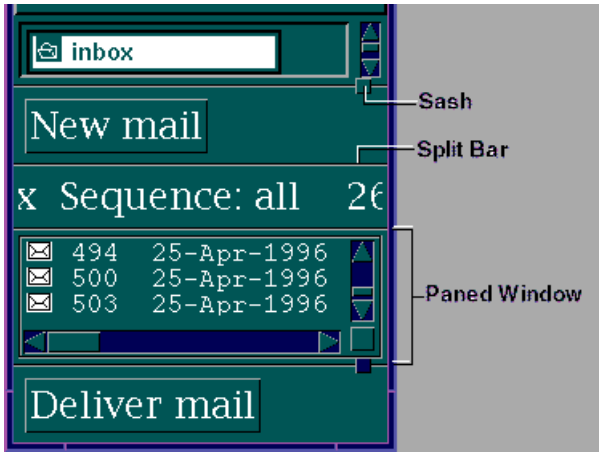


For more information, see the Paned Box (Control) reference page.

Split Bar

A *split bar* is a control that separates panes in a window and allows the user to change the size of the panes.

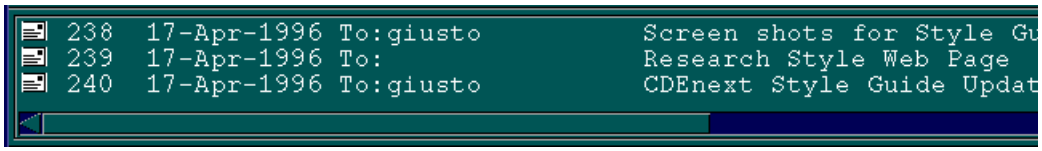
If the window is too small to hold all of the information provided, the user can split the window and scroll each pane separately.



For more information, see the Sash (Control) reference page.

Scroll Bar

A *scroll bar* is a control associated with an area that is too large to be completely displayed. It indicates to the user that more information is available and can be scrolled into view.



For more information, see the Scroll Bar (Control) reference page.

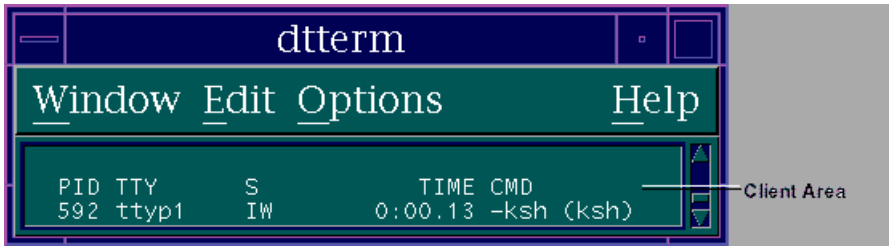
Using Window Manager Controls

The *window manager* is a specific application that manages the windows in an interface. Each window contains defined controls for user interaction provided by the window manager. The following window controls assist the user in interacting with and navigating in the interface:

- Client area
- Window title
- Window menu
- Maximize button
- Minimize button
- Window icon
- Size borders
- Additional buttons

Client Area

The *client area* is the application's display area where the user performs most application–level tasks. For example, if a user is working with a graphics editor or text editor, the client area contains the figure or document being edited. The client area is inside the window frame and can contain multiple work areas.

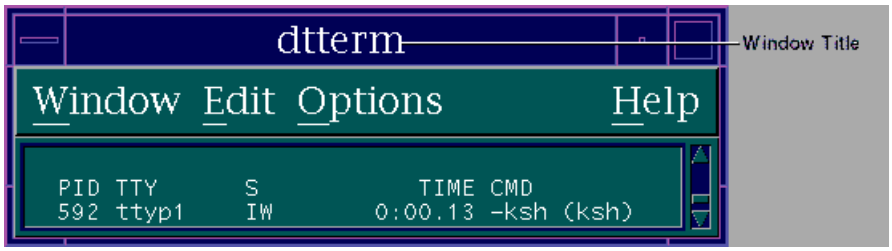


For more information, see the Client Area reference page.

Window Title

The *window title* contains a short string of text that identifies the window. Depending on the type of window and its function, the title is the title of the application or it indicates the purpose of the window.

If a window includes a title, it is in a horizontal bar at the top of the window, just above the client area and between the window menu button and the window control buttons (maximize and minimize).

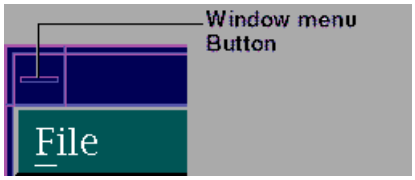


For more information, see the Window Title reference page.

Window Menu

The *window menu* displays a list of window actions. You should list all possible actions because keyboard users must interact with the window manager through the window menu. Almost all windows have window menus.

Locate a window menu button just above the client area. Align the left edge of the button with the left edge of the client area and the button just to the left of the window title, unless you include other buttons between the window menu button and the window title.

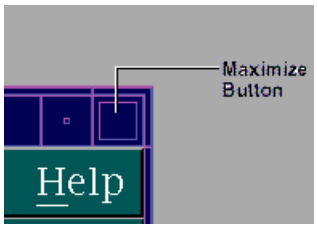


For more information, see the Window Menu reference page.

Maximize Button

The *maximize button* contains a graphic that indicates whether or not the window is maximized. If it is not, the user can press the maximize button to enlarge the window. If the window is already maximized, the user can press the maximize button to change the window to its original size and location. Using the maximize button is faster than choosing the Maximize or Restore choice from the window menu.

If the window includes a maximize button, it appears just above the client area and its right border is aligned with the right border of the client area. Primary windows contain maximize buttons; secondary windows usually do not.

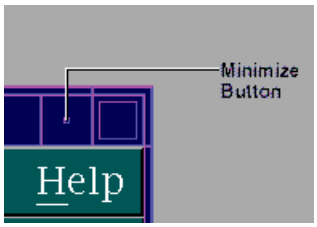


For more information, see the Maximize (Choice) reference page.

Minimize Button

The *minimize button* is a small square or down arrow graphic. The user can press the minimize button to minimize or iconify the window to an icon. Using the minimize button is faster than choosing the Minimize choice from the window menu.

If the window includes a minimize button, it appears just above the client area and directly to the left of the maximize button. Primary windows usually have minimize buttons; secondary windows do not have them.



For more information, see the Minimize (Choice) reference page.

Window Icon

The *window icon* is a minimized (or iconified) representation of a window or window family. The window manager places window icons in the window icon viewer.

The window manager minimizes all windows of a window family together. When a window is minimized, the application running inside the window continues running. To interact with window icons, the user opens the window icon viewer.

For more information, see the Window Icon and Window Icon Box reference pages.

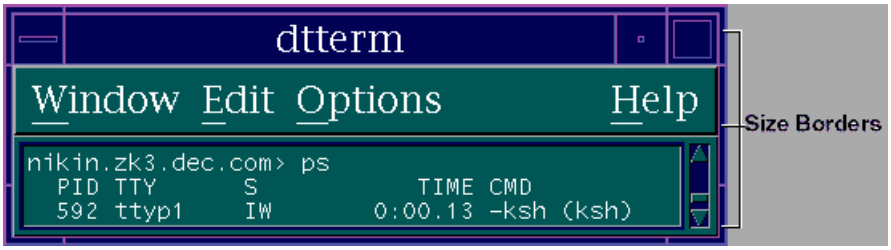
Size Borders

Although your application can supply an initial size for windows to the window manager, the user can vary window size for easier interaction. Using *size borders* is faster for mouse users than the Size choice in the window menu. Secondary windows do not include size borders.

The size borders are the outermost controls of the window frame and are made up of two parts: the corner handles and the edge handles. If you want to include size borders, put one corner handle in each corner of the window and one edge handle between each pair of corner handles.

Handles let the user change the height and width of windows without affecting the relative position of the window, as follows.

- Corner handles let users change the height and width of the window.
- Top and bottom handles let users change the height of the window without affecting the width.
- Side handles let users change the width of the window without affecting the height.



For more information, see the Window Frame reference page.

Additional Buttons

You can present additional window manager functions as buttons on the window frame. Place any additional buttons directly to the left of the minimize button or directly to the right of the window menu button and above the client area. Any additional buttons should correspond to an entry in the window menu.

Input Devices

This chapter describes input devices. The user interacts with the operating system through input devices such as a keyboard, a mouse, a track ball, a pen, a graphics tablet, and a joystick. This chapter focuses on standard keyboard and pointing devices, with special emphasis on the mouse.

Users employ input devices to:

- Manipulate elements (moving a slider, or moving or copying elements)
- Select elements (identifying or grouping elements for subsequent interaction)
- Activate elements (choosing elements such as an action choice that initiates a subsequent action)

Keyboards

Most users are familiar with using keyboards to interact with computers. The keyboard should allow the user to perform all of the functions available on the interface. Users also employ keys to augment pointing–device functions.

Keyboards differ in the number and type of keys available. This guide uses a model to describe keyboards and keys. The model does not correspond directly to any existing keyboard, but it assumes a keyboard with a standard set of keys. Keyboard Model and Key Bindings lists the keys that comprise the model, along with their corresponding functions.

In addition to the standard letter, number, and character keys, the keyboard model has the following special keys:

- Printing character keys: `/`, `\` , and `!`
- Modifier keys: `Ctrl`, `Alt`, and `Shift`
- Function keys: `F1` through `F10`
- Arrow keys, also called directional keys: `[larr ]`, `[rarr ]`, `[uarr ]`, and `[darr ]`
- The following additional keys:
  - `Backspace`
  - `Backtab`
  - `Begin`
  - `Cancel`
  - `Delete`
  - `End`
  - `Esc`
  - `Help`
  - `Insert`
  - `Menu`
  - `PageDown`
  - `PageUp`
  - `Enter`
  - `Return`
  - `Spacebar`
  - `Tab`
- The following optional keys (which, although useful, are either unnecessary or can be created by combinations of other keys):
  - `CapsLock`
  - `Copy`
  - `Cut`
  - `KeypadEnter`
  - `Home`
  - `ModeSwitch`
  - `NumLock`
  - `PageLeft`
  - `PageRight`
  - `Paste`
  - `ScrollLock`
  - `Select`
  - `Undo`

Throughout this guide, descriptions use the keyboard model keys. When the user can use an optional key from the keyboard model, the optional key is also described. Each of the keys described in this model should be available either as specified or by using other keys or key combinations if the specified key is unavailable.

Alternate Keys for Unavailable Specified Keys lists some of the most–often used alternative keys.

Table 1 Alternate Keys for Unavailable Specified Keys

Unavailable Specified Key	Key or Key Combination to Use
Cancel	Esc
Help	F1
Menu	Shift F10
F10	Shift Menu
Begin (and Home, if Home is available)	Alt [larr]
End	Alt [rarr]

Pointing Devices

A pointing device enables the user to move a pointer on the screen and to manipulate interface elements directly. Most keyboard actions can also be accomplished using a pointing device.

The most common pointing device is the mouse; however, a graphics tablet and stylus, pen, track ball, joystick, and other tools are also pointing devices. The model in this book uses the mouse as the pointing device.

Mouse

The mouse is the most common pointing device and contains one or more buttons with which a user can interact with an application or the operating environment. The actions users perform with the buttons vary depending on the functions assigned to each mouse button. You can map functions to buttons in different ways; therefore, the descriptions in this book actually describe virtual mouse buttons. Button names given when describing a user's action represent either a particular mouse button or a combination of buttons and actions that the user performs to complete a task.

This book uses the following virtual mouse button names:

SELECT

> Selects elements, directly manipulates elements to change values, activates elements, and moves the cursor. This is always the leftmost button for right−handed users.

ADJUST

> Adjusts selections. This is always `Shift` SELECT. It may also correspond to the middle button on a 3−button mouse.

TRANSFER

> Transfers and manipulates elements. This is usually the second mouse button; or it can be the same button as the SELECT button, in which case a single button is used for selection and transfer.

MENU

> Displays a pop−up, cascaded, or pull−down menu. The MENU button can also be used to activate elements within a menu system. This is always the rightmost button on a 3−button mouse for right−handed users. On a 2−button mouse `Alt` SELECT can be used instead.

Left−handed users can switch the mouse buttons to be the reverse of the previous descriptions.

The user can combine the use of the mouse buttons to perform different tasks; this is called a *chord.* A chord is performed by clicking or pressing two or more mouse buttons (not necessarily simultaneously) so that the buttons have the effect of being pressed at the same time. That is, the user presses one button, holds it down, and then presses another button.

For information about binding application functions to pointing device buttons and about the virtual mouse buttons, see the Mouse (Device) reference page and Mouse Techniques .

Mouse Techniques

The following list describes mouse techniques:

Move

> Moving the mouse, which moves the pointer in the interface.

Press

> Pressing and holding down a mouse button. A press can initiate an action, such as dragging an object to a printer icon.

Press and move

Pressing a mouse button without releasing it and then moving the position of the pointer, such as moving an element (like an icon) across the screen.

Release

Releasing a mouse button after pressing it. Releasing the mouse button performs an action initiated by pressing it, such as activating a push button.

Click

Pressing and releasing a mouse button without moving the pointer. Clicking a mouse button can perform an action such as activating a menu item.

Double−click

Performing two clicks in quick succession. A double−click performs the default action of a control. The time interval between clicks can be user defined.

Multi−click

Performing a number of clicks in quick succession.

Double−press

Performing one click followed by a press in quick succession without moving the pointer.

Multi−press

Performing a number of clicks followed by a press in quick succession without moving the pointer.

Mouse Motion

Two concepts define how the pointer tracks the motion of the mouse: gain and acceleration.

Gain refers to the ratio of distance the pointer moves to the distance the mouse moves. If the gain is increased, the pointer moves farther for a given mouse movement. The gain remains constant across the environment.

Acceleration refers to a change of gain based on the speed of mouse movements. The user can set the acceleration so that the gain is increased if the mouse is moved quickly, allowing faster pointer movement. If the mouse is moved slowly, the gain is reduced to allow finer pointer position adjustments.

Gain and acceleration are handled on a global scale by the operating system. Unless your application involves user configuration of mouse motion, your application should not change the gain and acceleration characteristics of mouse movement.

Audible and Visual Interface Cues

This chapter discusses audible and visual cues that enable the user to navigate in your interface. The user requires information from your interface to interpret interface interactions and responses. You can provide cues to assist the user.

The user also requires information when an event or situation occurs that prevents, or has the potential to prevent, the user's action from being completed as expected. Such situations are called *exceptions*. Exceptions occur when your application is unable to interpret a user's action. A typical method for notifying the user about an exception is to use cues.

The user can also request information from the system by activating help messages, which are a type of cue.

Audible Cues

An *audible cue* is a sound generated by the computer to draw the user's attention to, or provide feedback about, an event or state of the computer. Audible cues can be anything from a beep to a recorded or computer–generated message, and occur when the user attempts to perform an invalid action or when an event or state of the operating environment requires the user's attention. For example, the system may beep when the user tries to choose a menu item that is not available.

If your application will be used with computer hardware that has advanced audio capabilities, you can specify more elaborate and informative cues, such as speech synthesis. Audible cues are limited only by the available hardware and the usefulness of the cue. However, audible cues should not be intrusive or distracting, and the user should always be able to specify that the audible cues be turned off.

Some users may be unable to hear audible cues due to disabilities or environmental constraints. Therefore, audible cues should always be redundant to or replaceable by visual cues. For example, use an audible in conjunction with a flashing screen.

Visual Cues

A *visual cue* is a change in the appearance of an application's elements. Visual cues allow users to see the results of their interaction with the interface immediately. There are two types of visual cues: *graphical cues* and *textual cues*. For example, if the user places inappropriate information in a text–entry field, the shading of the text–entry field can change to alert the user that the information is unacceptable for that entry field. The shading in the text–entry field is a graphical cue. A textual cue is a more elaborate visual cue that includes messages that provide additional information.

This section discusses the following:

- Graphical cues
  - Pointers
  - Cursors
  - Emphasis
- Textual cues
  - Messages
  - Help

Graphical Cues

Both the pointing device (such as a mouse) and the keyboard have graphical cues. The graphical cue for the pointing device is the pointer; the graphical cue for the keyboard is the cursor. Pointers inform users about the destination of their pointing device actions; cursors inform users about the destination of their keyboard input. Graphical cues also indicate user actions and the states of controls. Such cues are called *emphasis*.

Windows and objects also have graphical cues. For example, if an object or window other than the one the user is working with requires attention, the appearance of the object's icon or the window icon could change.

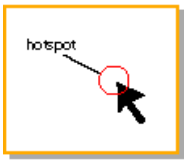
Pointers

The pointer is a graphical cue, usually in the shape of an arrow, that the user can move with a pointing device. The shape of the pointer provides the user with an important visual cue that indicates the function of the area, element, or control in which the pointer is currently located. The shape of a pointer can also reflect the state of a user interaction or of an input device.

Hot Spot

Each pointer shape has a hot spot, which is the actual position on the pointer where input device actions occur (see Pointer Hot Spot). The pointer can change shape to indicate a change in the function of the current area, but the location of the hot spot should not change on the screen.

Figure 1 Pointer Hot Spot



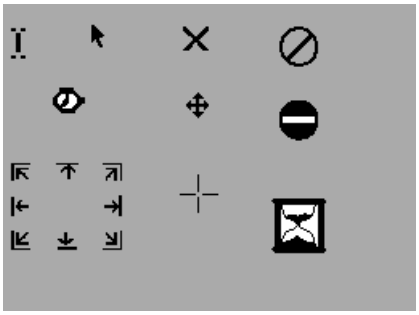
Pointer Shape

Pointer shape can indicate the function of the area, element, or control in which the pointer is located. For example, a pointer can change to a *cannot pointer* to indicate that the user cannot complete an action at the current screen location. Predefined Pointer Shapes shows some predefined pointer shapes used in Motif.

You can create new pointer shapes for functions that are not associated with a predefined pointer shape. The shape of the pointer should be associated with its purpose and be easy to distinguish. The shape should not create visual clutter, and the hot spot should be easy to locate. Also, use a predefined shape to symbolize only a function that the shape was designed to represent.

Your application can provide a palette of tools to give users easy access to graphically represented actions. When the user chooses a tool from a palette, the pointer shape changes, and pointer interaction is determined by the chosen tool. For example, choosing a line–drawing tool (such as a pencil) causes a line to be drawn as the user presses a mouse button and moves the mouse.

Figure 2 Predefined Pointer Shapes



For more information, see the Direct Manipulation, Mouse (Device), Palette Area (Area), Pointer, Pointer (Predefined), and Selection reference pages.

Cursors

The cursor is a graphical cue that shows where keyboard interaction occurs. Typically only one active cursor appears on the screen at a time.

Cursor Movement

The user can move pointers freely on the screen; however, the user can move the cursor only to valid cursor positions either on or between elements (for example, to menu bar items and push buttons, but not ordinarily to labels or headings). When the user clicks the pointing device, the cursor appears at the indicated location, if possible. The cursor never affects the pointer location. When the user presses a key, information passes to the operating system, and if appropriate, a response appears or an action occurs at the cursor's position.

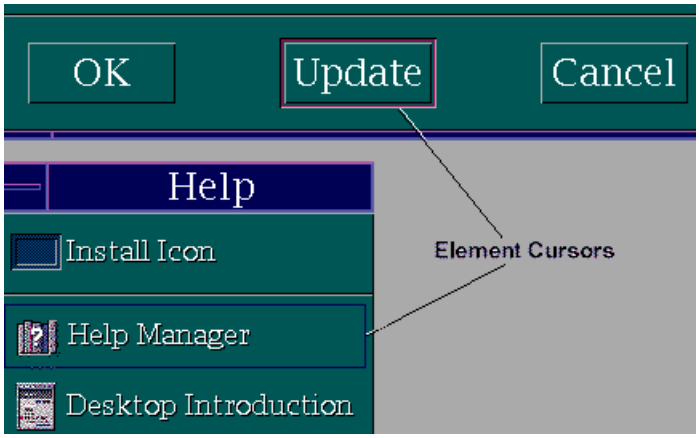
The appearance of the cursor changes, depending on its location. Therefore, cursors are sometimes categorized by their appearance and location. The way the user moves the cursor depends on the type of input device being used. See Input Devices for information about input devices.

Types of Cursors

There are three types of cursors: *element cursors*, *text cursors*, and *graphics cursors*. The following list describes each of these cursors:

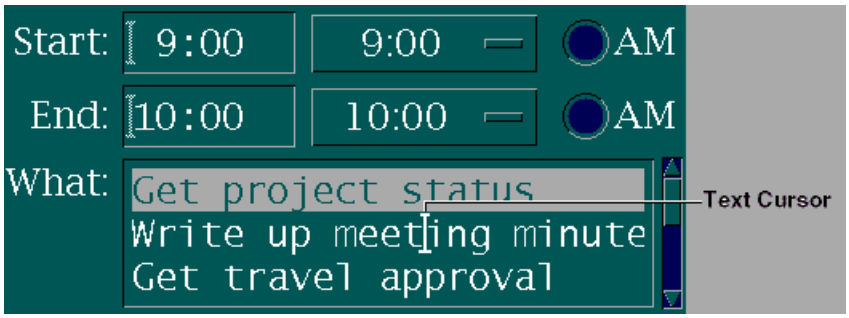
Element cursor

An element cursor appears on a control or on an element within a control that the user can select, activate, or manipulate in some way. For example, when an element cursor appears on a push button, pressing `Enter` activates the push button.



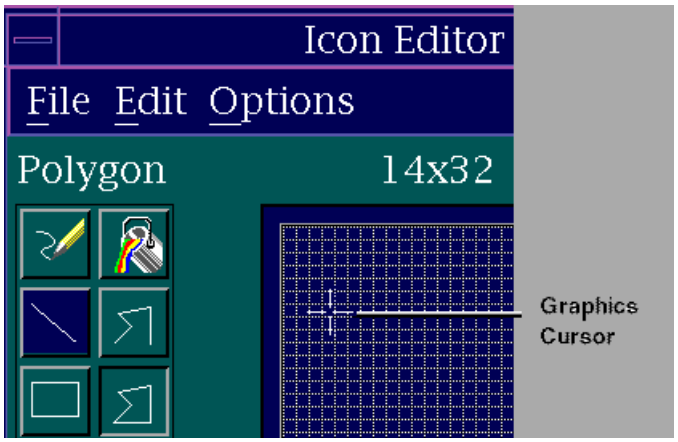
Text cursor

A text cursor is placed between characters within editable text and indicates where typed characters are.



Graphics cursor

A graphics cursor indicates an x,y location within a graphics area. Graphics cursors can be used for editing and positioning graphic elements.



For more information, see the Cursor, Input Focus, Keyboard (Device), Selection, and Text−Entry Field (Control) reference pages.

Emphasis

Emphasis is a graphical cue that distinguishes one element or group of elements from another and conveys information to a user. You typically use emphasis to indicate that an element or group of elements is:

- In use
- Selected
- Unavailable
- The source or target of an action
- The location that receives user input
- Ready to be activated
- The element or elements the user most recently interacted with

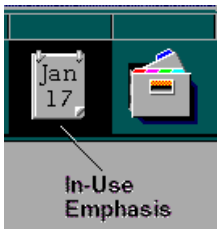
You can use emphasis to display other kinds of information appropriate for a specific application.

The type of emphasis you use should be determined by the interaction the user has with an element. More than one type of emphasis can be valid on an element at the same time; therefore the user can see different emphases simultaneously.

In−Use Emphasis

In−use emphasis indicates that at least one view is currently open on an object; the user can still interact with the object (see In−Use Emphasis).

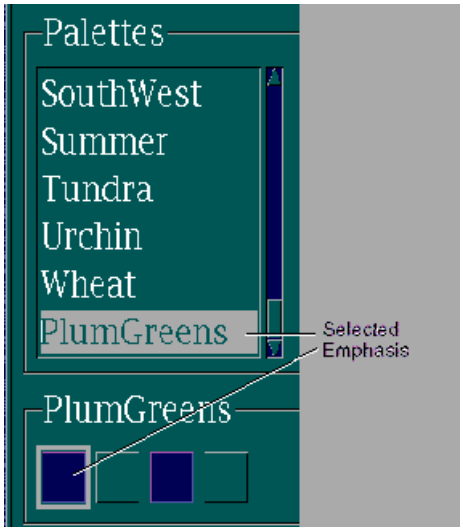
Figure 3 In−Use Emphasis



Selected Emphasis

Selected emphasis indicates that the user has selected an element (see Selected Emphasis). For more information about selection, see Selection .

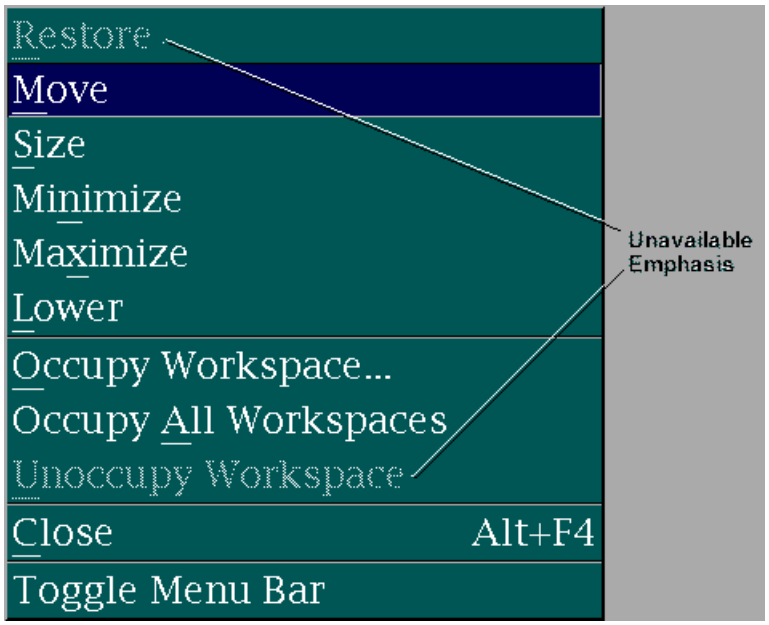
Figure 4 Selected Emphasis



Unavailable Emphasis

Unavailable emphasis indicates that the user cannot use a control at that time (see Unavailable Emphasis). A control with unavailable emphasis often appears dimmed compared to surrounding elements.

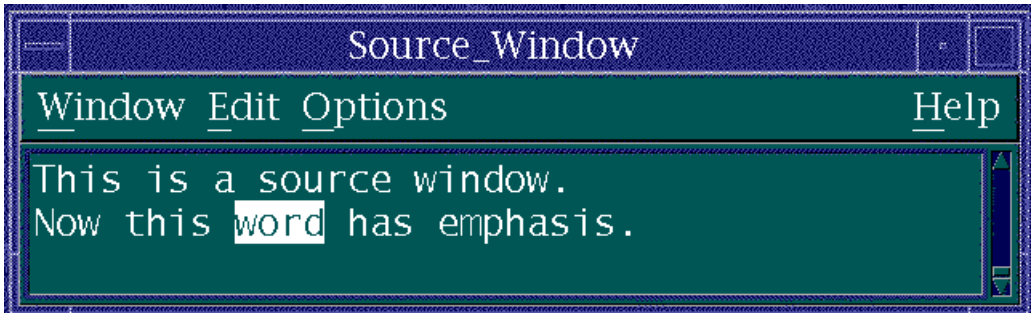
Figure 5 Unavailable Emphasis



Source Emphasis

Source emphasis indicates that an element is the source of an action, for example, the source of the element that the user is copying (see Source Emphasis). Source emphasis remains in effect until the action is completed.

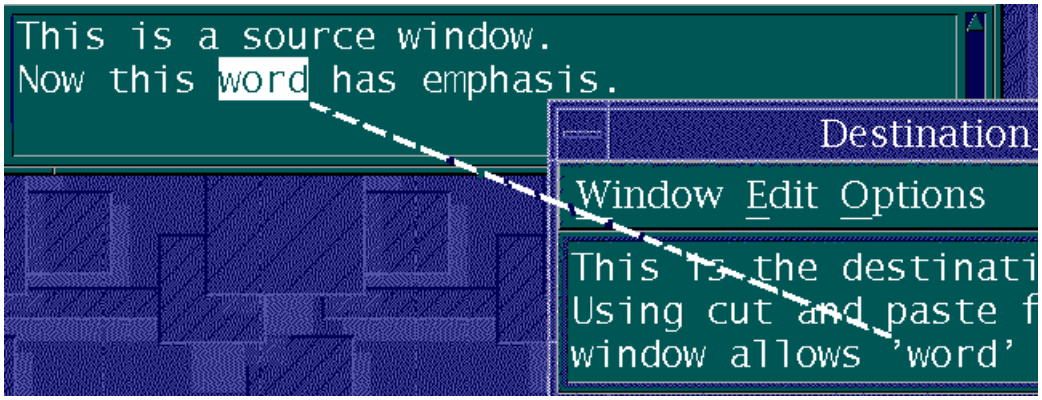
Figure 6 Source Emphasis



Target Emphasis

Target emphasis indicates the element that is the receiver of an operation (see Target Emphasis). When the user drags an item over an element on which it can be dropped, target emphasis is shown on the element. Target emphasis can indicate where the source element will be placed in the target element.

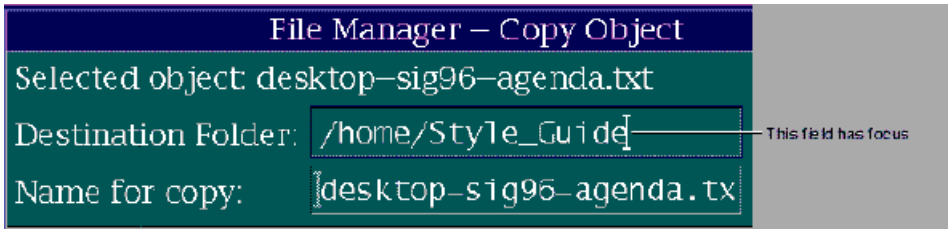
Figure 7 Target Emphasis



Focus Emphasis

Focus emphasis indicates which window or element will receive user input (see Focus Emphasis). For more information about focus emphasis, see Interacting with the Interface .

Figure 8 Focus Emphasis



Ready Emphasis

Ready emphasis, also known as armed emphasis, indicates that a control will be activated when the user completes the current action (see Ready Emphasis on the Update Push Button). For example, when the user presses the SELECT button while the pointer is on a push button, the push button displays ready emphasis to show that when the user releases the SELECT button, the push button will be activated.

Figure 9 Ready Emphasis on the Update Push Button

Often, when the user presses a push button, the push button appears to be pushed in to indicate ready emphasis.

Interacted Emphasis

Interacted emphasis indicates the control with which the user last interacted and which can be edited. It indicates which control is to be modified by various keyboard and menu–driven actions that paste data.

For more information, see the Copy To (Dialog Choice), Input Focus, and Move To (Dialog Choice) reference pages.

Textual Cues

When the user needs more information than can be conveyed with an audible or graphical cue, you can use a textual cue. A textual cue consists of a word or words that describe the current situation. You can display textual cues in various ways, including as information in a status area, as information in a message window (or messages), and as help information.

Messages

Messages provide detailed information to the user. They are appropriate when the information is particularly important or urgent. Messages can both describe a problem and explain how to correct the situation.

Besides text, messages can contain push buttons that help the user decide how to continue and graphics to indicate the message type. Some message windows contain groups of controls that allow the user to make more extensive corrections from the message window.

A message should clearly indicate why the message appeared and what actions can be taken. You can always provide access to help information in any kind of message to further help the user. Messages should be written so that a user unfamiliar with the operating system can understand the message and know how to proceed.

There are five types of messages:

- Information
- Warning
- Question
- Error
- In–progress

Information Message

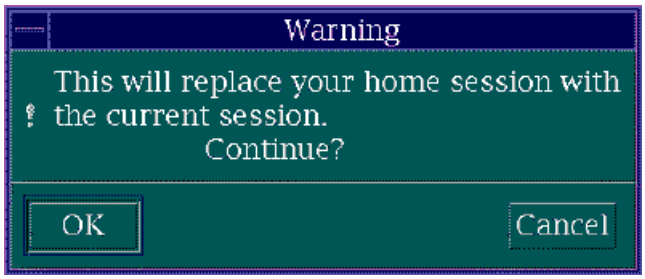
Use an information message when a situation occurs that the user may need to know about; for example, indicating that the user has received new mail (see Information Message).

Figure 10 Information Message

Warning Message

Use a warning message when the user should be aware of an existing situation but can continue working (see Warning Message). You can provide other actions to allow the user to modify the original request or to cancel it. For example, you can include a text–entry field that allows the user to enter a different file name if the file name originally chosen is a preexisting file name.

Figure 11 Warning Message

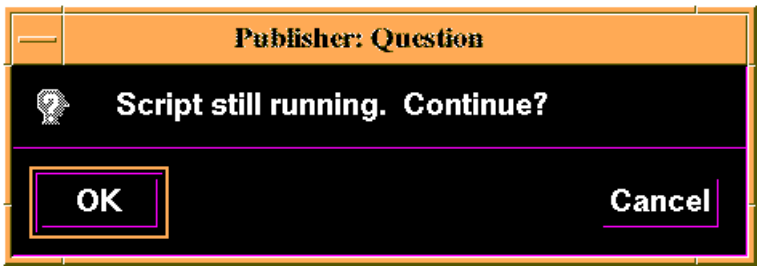


Question Message

Use a question message when a situation arises that does not require the user's immediate attention (see Question Message).

A question message can contain controls, such as a text–entry field, that allow the user to attempt to correct data that is invalid or incorrect. For example, a question message could contain a text–entry field to allow the user to indicate a new printer name if the current printer is out of paper, thus providing an alternate method for completing the task.

Figure 12 Question Message



Error Message

Use an error message when a situation requires the user's immediate attention (see Error Message).

An error message can contain controls, such as a text–entry field, that allow the user to attempt to correct data that is invalid or incorrect. For example, an error message could contain a text–entry field to allow the user to indicate a new printer name if the requested printer is not available.

Figure 13 Error Message

In–Progress Message

Use an in–progress message to provide feedback to the user during long processes (typically 15 seconds or more). An in–progress message indicates progress toward the completion of a process, such as copying or sorting a group of objects or files (see In–Progress Message).

An in–progress message can indicate a specific amount of time. For example, it could contain a digital clock that displays the time remaining in a process. An in–progress message can also indicate a relative amount of time.

Figure 14 In–Progress Message

For more information, see the Action Message, Gauge (Control), Information and Message Areas (Area), Information Message, In–Progress Message, and Message reference pages.

Help

Your application should provide information to the user about how to use the application and how to recover from exceptions. This information is known as help information.

Ideally, an application's help facilities should work together with its methods and mechanisms for exception handling to provide the user with all of the information needed to solve any problem.

Help information describes your application and user interaction. Help information might include information on how to perform a task or reminders for users who already know how to perform the task but need to know more detailed information. The information you provide should be clear, concise, and written in everyday language. In addition, the user needs to be able to access and leave help information easily.

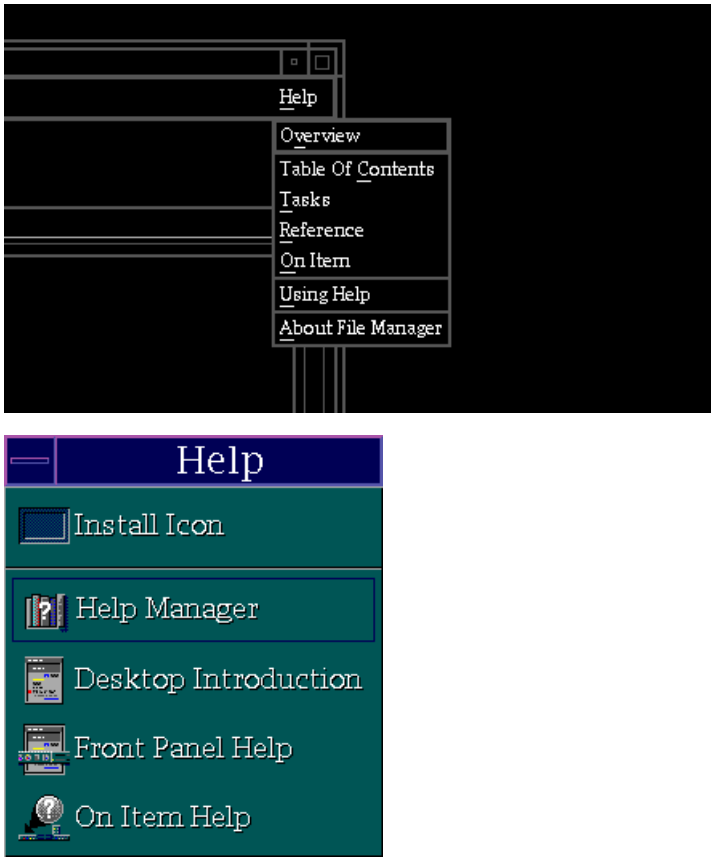
Types of Help Information

Your application should provide several types of help information, including information about:

- The contents of a window and the tasks the user can perform in the window
- Any element, including the element's purpose and ways to interact with the element
- Key assignments on the keyboard
- Using the help facility

Your application should also provide an index of all of the topics for which help is available. Types of Help Information lists types of help information in various help menus.

Figure 15 Types of Help Information



For more information, see the Context–Sensitive Help, Help (Menu/Action Choice), and Keyboard (Device) reference pages.

Specifying Attributes and Navigating

This chapter describes the navigation model for mouse and keyboard users. *Navigation* refers to how the user moves pointers and cursors within your interface. This chapter also discusses the activation of controls. *Activation* refers to using controls to perform actions, such as initiating an action choice.

Input devices have different actions, depending on which part of the interface the user is interacting with. Due to the inherent flexibility in mouse manipulation, users with a mouse and keyboard can access windows and controls more easily than users with just a keyboard.

This chapter describes:

- How users know which element they can currently interact with
- How users navigate within the interface
- How users activate controls and change values
- Shortcuts to help users navigate, select, activate, and change the value of elements

Introducing the Window Manager

Applications that interact with the user via windows must present information in a clear manner. Similarly designed windows help the user to understand the interface better. The system requires that an application manage the windows in the interface. That application is called the *window manager*. The window manager is a specific application that manages the windows in an interface. The window manager follows the same guidelines for input, navigation, selection, and activation as described in this book.

The window manager supports multiple applications, each with one or more primary windows. It also supports secondary windows and associates each secondary window with a primary window or another secondary window.

You can design the window manager to allow users to configure it, as well as other elements, including key bindings, menu contents, and default window controls. The window manager supplies the default settings. Users can toggle between their configured settings and the default settings.

Supporting and Designing Windows

A typical interface has several applications in operation simultaneously, and each application has a primary window. Applications can also have secondary windows to communicate context−specific interactions to users.

For more information, see the Primary Window and Secondary Window reference pages.

Interacting with the Interface

Several windows may be open simultaneously, but input from the keyboard is directed to only one of them at a given time. This window is called the *active window*.

Similarly, the active window may contain many controls, but keyboard input is directed to only one of those controls at a given time. The control to which keyboard input is directed is said to have the *input focus*, also called keyboard focus, or just focus.

The control with input focus generally displays focus emphasis, indicated by a cursor on or within the control. For example, a push button displays an element cursor on it when it has focus, a list box displays an element cursor on some item within the list when it has focus, and a text −entry field displays a text cursor at the current insertion point within the text when it has focus.

Some controls may display a graphical cue similar to a cursor even when the control does not have focus. In that case, when the control has focus the graphical cue changes in some way. For example, the text cursor may be visible even in a text−entry field that does not have focus, but might flash on and off when the field does have focus.

Stacking Order

Stacking order determines what parts of windows or controls are visible when they overlap. The user can choose whether to have the stacking order change as the focus changes. Changes in the stacking order are especially important at the window level. There are two different stacking policies: manual stacking order and automatic stacking order (see Manual and Automatic Stacking Order).

Figure 1 Manual and Automatic Stacking Order

Manual Stacking Order

Manual stacking order refers to a window that does not have to be on top of all other windows to be active, meaning the user can interact with a window without changing the stacking order of the window hierarchy. With manual stacking order, the user must perform a specific action to surface an active or inactive window to the top of the stacking order.

Automatic Stacking Order

Automatic stacking order refers to an active window that the interface automatically surfaces to the top of the stacking order when it receives input focus. The user does not need to explicitly surface the window.

Within a window, stacking order is less important since controls or elements generally do not overlap one another. An important exception is within areas devoted to graphics. In this case, you can use an automatic stacking policy; but most commonly, selected graphics elements change

their stacking order by specific actions, typically chosen from a menu.

Focus Policy

Only one window and only one control within that window can have input focus at any given time. Determining which window or control will receive input focus is dependent on the focus policy that is in effect. A *focus policy* is a specific policy for moving the focus among windows and controls. There are two focus policies: explicit focus and implicit focus. The user can choose which focus policy to use, depending on the task and environment.

Explicit Focus

When using an explicit focus policy, sometimes called click–to–type, the user must explicitly indicate which window or control receives the input focus by clicking a mouse button or by using navigation keys on the keyboard. When a new window is created, it often receives the input focus automatically.

Remember that input focus determines only the window or control affected by keyboard operations. Mouse–based operations (clicking on a button for example) affect the window and control under the pointer, regardless of the current input focus. In fact, when an explicit focus policy is in effect, most mouse–based operations move the focus to the window and control that contain the pointer.

Implicit Focus

When an implicit focus policy is in effect, keyboard events are automatically sent to the window and control where the pointer is located. Implicit focus is sometimes referred to as pointer focus, focus–follows–mouse, or real estate–driven focus.

Different parts of the interface can have different focus policies. For example, you can use an implicit focus policy to determine which window has focus (that is, the one containing the pointer); you can use an explicit focus policy to determine which control within the window has focus (for example, the user has to click on a text–entry field to move focus to it). Mouse users generally do not have to be concerned with the current focus policy because the mouse button action generally passes to the control under the pointer.

Navigating Within the Interface provides more information about navigation within particular parts of the interface.

Summary of Stacking Order and Focus Policies

The user can choose either manual or automatic stacking order when using implicit or explicit focus. Summary of Stacking Order and Focus Policies summarizes how stacking order and focus policies work together within the interface at the window level.

Table 1 Summary of Stacking Order and Focus Policies

If you have...	And...	Then...
Automatic stacking order	Implicit focus	User surfaces the window by moving the pointer into it (after a delay or when the pointer stops moving).
	Explicit focus	User must click on the window to surface it.
	Manual stacking order	Implicit focus
		User must click on the window to make it

	active; but clicking on the window does not make it surface. Clicking in the window frame surfaces the window.
Manual stacking order	

For more information, see the Active Window, Cursor, Input Focus, Secondary Window, Stacking Order, and Window Navigation reference pages.

Navigating Within the Interface

This section discusses how the user navigates within and between the elements of your interface when an explicit focus policy is in effect. Users can navigate in the following ways:

- Between windows (window navigation)
- Within windows
  - Between tab groups in windows (tab group navigation)
  - Between controls within tab groups (control navigation)
  - Within controls (internal navigation)
- Between and within menus (menu navigation)

Navigating Between and Within Window Families illustrates navigating between and within window families.

For more information on mouse and keyboard navigation, see Keyboard Model and Key Bindings and Mouse Techniques .

See also the Cursor, Input Focus, Keyboard (Device), Mouse (Device), and Pointer reference pages.

Figure 1 Navigating Between and Within Window Families

Navigating Between Windows

Each primary window may have one or more related secondary windows. Together, a primary window and its related secondary windows are called a window family. Users can navigate within a window family and between window families.

Window navigation for mouse users differs according to which focus policy is in effect. For environments with implicit focus, the input focus is wherever the pointer is located. For environments with explicit focus, users explicitly indicate where the input focus is by moving the pointer to the appropriate window or control and clicking a mouse button or by using the following keys to specify which window or control has focus:

`Alt Esc`

Moves the focus forward from one window family to the next.

`Shift Alt Esc`

Moves the focus backward from one window family to the next.

`Alt F6`

Moves the focus forward among windows within a window family.

`Shift Alt F6`

Moves the focus backward among windows within a window family.

When you use a window icon box, the window icon box is a separate primary window. When you do not use a window icon box, each window icon that appears on the desktop is a separate window family, and so `Alt Esc` and `Shift Alt Esc` navigate to window icons as well as to open windows.

Using a mouse and explicit focus, the user can move between windows by moving the pointer to the window to receive focus and clicking the SELECT button. With implicit focus, the window under the pointer automatically becomes the active window.

For more information, see the Window Navigation reference page.

Navigating Within a Window

Using implicit focus within a window, the user can navigate to a control by moving the pointer to the control. With explicit focus, the user can navigate to a control by moving the mouse to it and clicking on it or by using the keyboard.

The following sections describe how to use the keyboard to navigate within a window. The user can navigate between groups of controls (tab groups) and between individual controls. For example, the user might navigate between a radio box and a combination box (groups of controls) and between radio buttons within a radio box (individual controls). Also, the user can navigate within controls, for example, within the text–entry field in the combination box to move between characters in the field.

The following sections describe:
- Navigating between tab groups
- Navigating between controls
- Navigating within controls
- Focus–only navigation

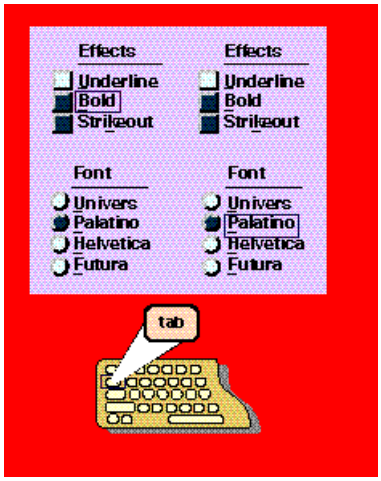
Navigating Between Tab Groups

A *tab group* is a control or group of controls that the user can navigate to by pressing `Tab` (to move forward), `Shift Tab` (to move backward), or by moving the pointer to a tab group and pressing the SELECT button. Within some controls, multiline text in particular, `Tab` is interpreted by the control itself (for example, in text it may insert a `Tab` character). In those cases, the user can press `Ctrl Tab` to move focus forward to the next tab group and `Ctrl Shift Tab` to move backward.

Tab groups fall into two categories:
- Tab groups that contain a group of other controls (for example, a radio box that contains a group of radio buttons)
- Tab groups that consist of a single control (for example, a text field)

Within a tab group that contains a group of controls, the user must navigate among those controls (see Navigating Between Controls). Within a tab group that consists of a single control, such as a text–entry field, the user must navigate within the control (see Navigating Within Controls). Navigating Between Tab Groups with the Tab Key illustrates navigating between tab groups.

Figure 2 Navigating Between Tab Groups with the Tab Key



Navigating Between Controls

The user can move focus between controls within a tab group (or within a window without distinct tab groups) by using directional keys or by moving the pointer to a specific control and pressing the SELECT button.

You can treat a push button as an individual control in a tab group or as an individual tab group. The user can then navigate among the push buttons by using directional keys, by pressing `Tab`, or both, depending on how you design the push buttons.

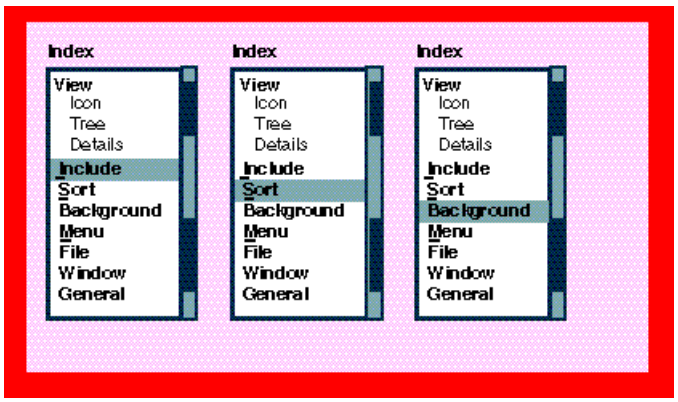
For more information, see the Push Button (Control) reference page.

Navigating Within Controls

Keyboard users use directional keys to move within a control such as a text–entry field or list box (see Using a Directional Key to Navigate Within a Control). Mouse users move the pointer and press the SELECT button to move within a control.

Even when you use an implicit focus policy within a window, you can still use an explicit focus policy within an individual control. For example, when you use an implicit focus policy, a text–entry field has focus only while the pointer is within the text–entry field. The user must still click the SELECT button or use keyboard navigation to move the text cursor; the cursor does not move when the mouse is moved within the text –entry field.

Figure 3 Using a Directional Key to Navigate Within a Control



Focus–Only Navigation

When you use an explicit focus policy within a window, a mouse user can move focus to a control or element by clicking the SELECT button on it. However, this sometimes has another effect; for example, it could activate a push button.

Focus–only navigation allows for moving focus to a control without causing side effects. With focus–only navigation, the user can click `Ctrl` SELECT on a control to bring focus to it but not activate it.

Within some controls, `Ctrl` SELECT may already be defined as an action. In this case, focus–only navigation is disabled and the control's definition remains in effect.

For more information, see the Control Navigation, Internal Navigation, and Tab Group reference pages.

Navigating Within and Between Menus

Mouse users click or press the SELECT button or the MENU button to display menus and navigate among them. Menus, other than menu bars and tear–off menus, are temporarily displayed (see Menu Navigation While Pressing the SELECT button). The user can display pop–up menus by using the MENU button. The user can display pull–down, option, or cascaded menus by pressing the SELECT button or the MENU button on a cascading choice.

Pop–up menus and cascaded menus are spring–loaded –– that is, when the user makes a noncascading choice within the menu, the menu automatically disappears. These menus may also be spring sensitive. A menu becomes spring sensitive when it appears as a result of pressing an appropriate mouse button, or when an appropriate mouse button is pressed in a menu that is already displayed.

In a spring–sensitive menu, an active cursor follows the pointer as the user moves the pointer through the menu. The following may occur in a spring–sensitive menu:

- When the user moves the pointer over a cascading choice, a corresponding cascaded menu immediately appears; if the user moves the pointer off the cascading choice (but not into the cascaded menu), the cascaded menu disappears.
- If the user releases a mouse button over a noncascading choice within the menu, that choice is activated (or toggled), and the menu disappears.

When a menu is spring sensitive, you should use an implicit focus policy within the menu. When a menu is not spring sensitive, you should use an explicit focus policy within the menu. The user may move the cursor within the menu only by pressing an appropriate mouse button within the menu (which makes it spring sensitive) or by using keyboard navigation.

When a menu appears as a result of a keyboard operation, the menu is not spring sensitive. In addition, if a spring–loaded menu remains displayed when a mouse button is released (for example, if it is released on a cascading choice), the menu stops being spring sensitive.

Menus displayed by mouse users may or may not be spring sensitive. Menus displayed by keyboard–only users are never spring sensitive and always use an explicit focus policy.

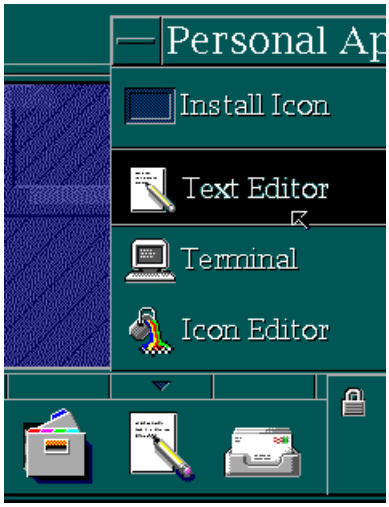
Keyboard–only users navigate menus by using the directional keys and special keys. The user presses `F10` (or `Shift Menu`) to move to menu bars and presses `Menu` (or `Shift F10`) to display pop–up menus. Once the cursor is within a menu or menu system, the user navigates by pressing directional keys.

Some lists, like menus, may be displayed only temporarily. In particular, drop–down lists and drop–down combination boxes contain a list button and cascading choice from which a spring–loaded list may be temporarily displayed.

Like menus, spring–loaded lists may or may not be spring sensitive, depending on the following:

- If a list is displayed as the result of the user pressing the SELECT button over a list button, the list is spring sensitive. Releasing the SELECT button on an item in the list selects it and removes the list.
- If the list is displayed as the result of a keyboard operation or remains displayed when the user releases the SELECT button over the list button, the list, though displayed, is not spring sensitive. The user can use keyboard navigation to move the cursor through the list.

Figure 4 Menu Navigation While Pressing the SELECT button



For more information, see the Menu (Control), Menu Guidelines, and Spring–Loaded (Control Type) reference pages.

Activating Controls

When the user clicks the SELECT button on a choice that can be activated, an action occurs. When the user presses the SELECT button on a push button, the push button appears with ready emphasis to indicate that releasing the SELECT button will activate the push button.

Choices that support activation include action choices, dialog choices, and cascading choices.

For more information, see the Action (Choice Type), Cascading (Choice Type), and Dialog (Choice Type) reference pages.

Changing Values in Controls

There are several ways the user can change the value that is managed by a control, depending on the type of control. Controls that allow the user to change values include check boxes, radio buttons, spin boxes, and sliders.

In a check box, the user can click the SELECT button while the pointer is on the control or press `Spacebar` while the focus is on the control to change the value of the check box. For example, if the user clicks the SELECT button while the pointer is on a check box that represents the choice Boldface, the check box changes from its current state and cycles through the states available: boldface on, boldface off, and possibly a special indeterminate state.

In a radio box, clicking on a radio button that is off turns it on and turns off any other buttons in the box that were on.

In a spin box, the user changes the value by clicking the SELECT button while the pointer is on an arrow. For example, if a spin box represents the days of the week, the user can click the SELECT button while the pointer is on the up arrow button to change to the next day of the week.

In a slider, the user changes the value represented by the slider by directly manipulating the slider arm, that is, by pressing the directional keys or by pressing the SELECT button while the pointer is on the slider arm and then moving the pointer in the direction of the desired value. For example, if a slider represents decibels for sound output, the user can press the `[rarr ]` directional key or press the SELECT button and move the slider to the right to increase the volume of the output.

Using Shortcuts

To speed user actions, you can provide your interface with shortcuts such as first−letter cursor navigation, mnemonics, default actions, and shortcut keys. The following sections discuss these shortcuts.

First−Letter Cursor Navigation

You can use first−letter cursor navigation in list boxes, containers, and similar controls. In first−letter cursor navigation, the user navigates to and selects an item in a list by inputting (from the keyboard) the first character of the item. Inputting the character again allows the user to navigate to and select the next item in the list beginning with that character (see First−Letter Cursor Navigation in an Index List).

Figure 1 First−Letter Cursor Navigation in an Index List

For more information, see the First−Letter Cursor Navigation reference page.

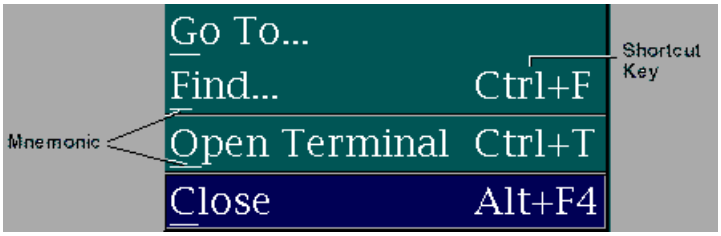
Mnemonics

A mnemonic is a readily recognized character that the user can type to move the cursor quickly from one place in a window to another and, in some cases, to activate or change the value of a specified control. A mnemonic is usually represented as a character from the Control label and is underlined in the label to indicate its function as a mnemonic (see Mnemonics).

The user presses `Alt` and the mnemonic key to activate action, cascading, and dialog choices in buttons and cascading choices in menu bars. Also, the user presses `Alt` and the mnemonic key for a label associated with a tab group to navigate to the tab group. The user presses only the mnemonic key when in a menu or tab group to activate or toggle a choice in the same menu or tab group.

For example, the user may need to navigate to a specific choice within the File menu. To do so, the user presses `Alt F`, which displays the File menu and highlights the first item within it. Then, the user presses `S` to activate the Save choice.

Figure 2 Mnemonics



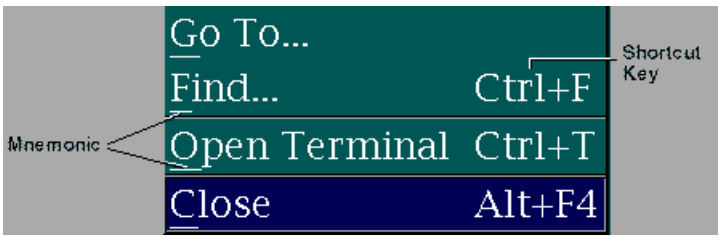
For more information, see the Mnemonic reference page.

Shortcut Keys

A shortcut key, or accelerator, is a key along with a modifier that the user can press to activate a choice (see Shortcut Keys). You should provide shortcut keys for choices that the user frequently chooses.

A shortcut key usually appears next to the choice to which it pertains so that the user can learn to associate the shortcut key with the choice. You might want to provide a mechanism that allows users to turn off the display of the shortcut keys. Keyboard Model and Key Bindings lists predefined shortcut keys.

Figure 3 Shortcut Keys



For more information, see the Shortcut Key reference page.

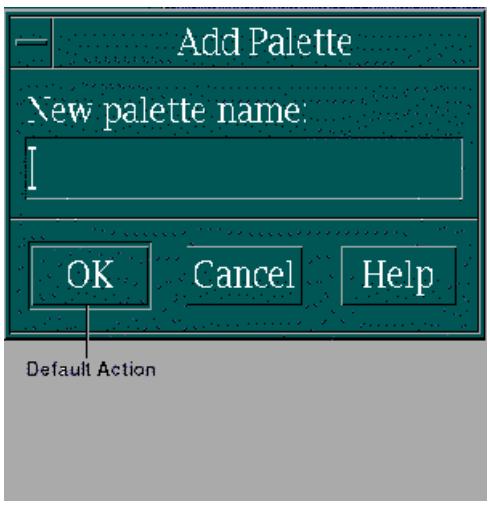
Default Action

Within a window, especially a secondary window used for a message, there is often an action that the user may want to perform at any given time. Your application can have this action be the default action of the window; the user can then invoke it (when the window has focus) simply by pressing `Enter`, `Return`, or `keypadEnter`. If `Enter` is used for other purposes, such as when the focus is in multiline text, use `Ctrl Enter`.

In general, the default action is the action choice of a push button displayed in the window. You should display that push button with default emphasis.

Assigning a default action allows the user to invoke an action without first navigating to (then activating) the desired choice. Default actions are also useful for inhibiting the user from making potentially harmful choices. Default Action illustrates the default action for a text–entry dialog.

Figure 4 Default Action



For more information, see the Default Action reference page.

Selection

The act of selection allows the user to choose one or more values from a set of values or to pick one or more elements (usually data elements or objects) from a given set of elements. The user then applies some action to the value. For example, a user might select a paragraph from within text, copy it elsewhere, check it for spelling, or delete it.

This chapter introduces the selection process and discusses the following topics:

- Selection models
- Mouse–based selection techniques
- Keyboard–based selection techniques
- Mouse–based selection modes
- Keyboard–based selection modes
- Complex scopes

Selection Models

There are many factors that affect how a user can perform a selection. For example, depending on what control the user is currently working in, a selection may be limited to just the selection of characters or to graphic objects. Users may also be limited in how they can select elements. For example, a user may be able to select only one item at a time or multiple items. How a user selects items is controlled by the *selection model.* A selection model describes how selection works in a selection scope. The following sections describe selection and the factors that affect selection.

Selection Scopes and Controls

A user performs selections within the confines of controls. The provision of many controls enables the support of selections. For example, although the apparent purpose of containers is to group items, the reason that these items are grouped is so that a user can select and act on them. Controls limit what elements may be selected and how they are to be selected. The following are examples of controls and their selection limits:

Text−entry field

Supports the selection of characters; that is, each character is a data element that can be selected.

List box

Supports the selection of list elements. List elements may be either data elements or objects.

Container

Supports the selection of the objects in the container.

Selection Model Types

How a user performs a selection depends on the selection model. For example, some selection models support the selection of only one item at a time; others support the selection of multiple elements at one time.

A selection model consists of the following:

Selection techniques

Control how a user makes a selection.

Selection modes

Control how the selection technique interprets the requested selection action.

Selection policies

Control how selection techniques work.

Persistent and primary selections

Control how a selection performed in one scope affects selections in other scopes.

The following sections briefly describe these selection model components. Refer to the Selection Models reference page for a description of the selection models themselves.

Selection Techniques

Selection techniques are the multiple ways that the user can interpret how to make a selection. For example, in a list, the user selects a list element by clicking on it. In text, the user selects a character by pressing the SELECT button, moving the mouse to highlight the character, then releasing the SELECT button. A number of factors affect how a user selects items, including the selection scope and the layout of the items to be selected. For example, if a user needs to choose multiple items from a list where the items are not next to each other, then the user selects the items individually. However, if the list items are next to each other, the user can select the list items as a group.

There are many reasons for having different selection techniques. Some techniques result from different preferences and some techniques result

from user needs.

Refer to **Mouse–Based Selection Techniques** and Keyboard–Based Selection Techniques for a detailed description of selection techniques for mouse and keyboard–based users, respectively.

See the Selection Techniques reference page for guidelines on selection modes.

Selection Modes

Selection modes control how selection techniques work. For example, a user can select a list item by clicking the SELECT button on that item. However, what actually happens to that item depends on the selection mode in effect.

Select mode and *toggle mode* are the two selection modes available for the mouse user. When the user selects a list item and select mode is enabled, that item is selected and all other items are deselected. When toggle mode is enabled, the result of the selection depends on the current state of the item. If the selected item is already selected, it becomes deselected; otherwise it is selected. In other words, the state of the item is *''toggled.''*

Select mode is for users who prefer to select items, perform actions on those items, then repeat the procedure with an entirely new selection. Toggle mode is for users who prefer to select items, perform actions on those items, then add or remove items from the selection to perform other actions.

Refer to Mouse–Based Selection Modes and Keyboard–Based Selection Modes for a detailed description of selection modes for mouse and keyboard–based users, respectively.

See the Selection Modes reference page for guidelines on selection modes.

Selection Policies

Selection policies affect how a user can perform a selection, such as how a user can change a selection, whether an element's selection state can be toggled, and so on. For example, when a user selects a background color from a list of colors, it makes sense for the user to be able to choose only one color at a time. In some applications, it is even possible to have a no selection option.

Selection policies also control whether selections of groups of items should be contiguous or noncontiguous. Contiguous elements are any two elements that, when selected, also have all the elements between them selected. Noncontiguous elements are elements that, when selected, do not have the elements between them selected. Selecting contiguous elements is useful when the selection is for ordered sets of items, such as alphabetized lists. Selecting noncontiguous elements is useful when the selection is for nonordered sets of items, such as graphic data.

There are a number of selection policies that you should consider when choosing selection models. These policies fall under the general categories of policies that define the selection models, policies for modifying selections, and area and range policies.

Policies that define selection models

> These policies include the count, deselection, and contiguity policies. These policies limit how users can select and deselect elements and determine whether discontiguous selections are allowed.

Policies for modifying selections

> These policies include the adjustment, toggling, and toggle removal policies. These policies affect how users can change selections.

Area and range policies

> These policies affect how users can change area and range selections. Some of these policies depend on whether or not areas or ranges are dense. A dense area or range is one that has a relatively high population of elements that the user can select. These policies include the area inclusion, technique initiation, and end–point inclusion policies.

Refer to Selection Policies for a detailed description of the available selection policies.

Persistent and Primary Selections

Whether or not a selection retains its selection state while a user performs other selections depends on whether the selection scope supports *persistent selection* or *primary selection*. In a selection scope that supports persistent selection, a selection is independent of any selection in any other selection scope. For example, a user can select text in a text control and a list item in a list control, and both selections are independent. In a

selection scope that supports primary selection, existing selections in other selection scopes that support primary selection are deselected. In other words, only one selection is allowed.

You must provide primary selections for primary transfers. A primary transfer is one that allows a user to transfer a current selection from a source directly to a destination without using dragging or the clipboard as an intermediary. To achieve this, only one primary selection can exist at any given time so that when a user selects an item for primary transfer, then clicks the TRANSFER button, the contents of the primary selection are transferred to wherever the user clicks the TRANSFER button.

For more information on persistent and primary selection, see the Primary Transfer and Selection reference pages.

Mouse–Based Selection Techniques

Not all mouse–based selection techniques are available for all controls. There are techniques for selecting or deselecting individual elements or groups of elements. The group techniques range from basic group selection techniques (such as range selection) to more advanced techniques (such as margin selection). This section describes the different types of individual and group selection techniques. It also describes adjustment techniques, which allow users to change an identified selection by either adding or removing elements from it (thus enlarging or shrinking the region).

Mouse–based selection depends on the select and toggle modes. Select mode is more common for basic selection. The discussion in this section assumes that select mode is in use. For more information on selection modes, see the Selection Modes reference page.

Individual Selection Techniques

When select mode is active, *individual selection techniques* allow the user to select an element. All other elements are deselected, whether they were originally selected or not. There are two individual selection techniques: point and browse.

Point Technique

With the *point technique*, the user clicks the SELECT button on the desired element (see Point Technique).

In select mode, if a user tries to use the point technique to select text, or in the background of a graphics scope, this technique simply deselects everything. Because the pointer is not on an element that can be selected, this results in a selection that does not contain anything.

Figure 1 Point Technique

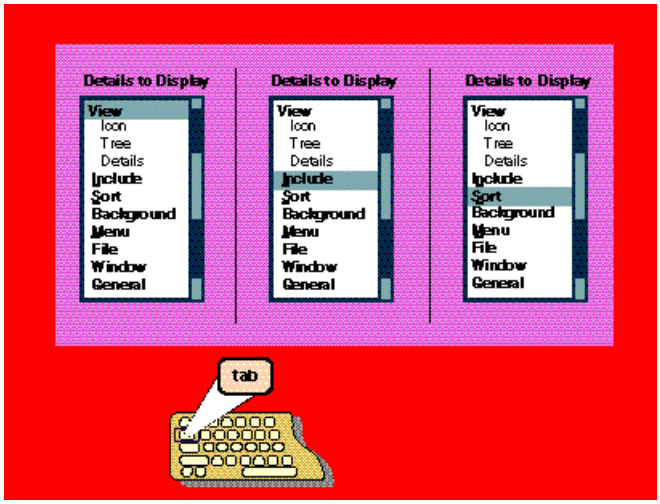
For more information, see the Point Technique reference page.

Browse Technique

With the *browse technique*, the user presses the SELECT button within the scope, moves the pointer (browses) until it is on the desired element, then releases the button. As the pointer moves over the elements that can be selected, selection emphasis is moved from one element to the next until the final selection is made at the release of the SELECT button (see Browse Technique).

Because the elements that can be selected are emphasized as the pointer moves over them, the browse technique is a way to identify these elements.

Figure 2 Browse Technique



For more information, see the Browse Technique reference page.

Group Selection Techniques

When select mode is active, *group selection techniques* allow the user to select a group of elements at one time. Group selection techniques include:

Range technique

User selects a group of ordered elements that are within a range.

Area technique

User selects a group of contiguous elements.

Touch swipe technique

User selects a group of noncontiguous elements by touching them with the pointer.

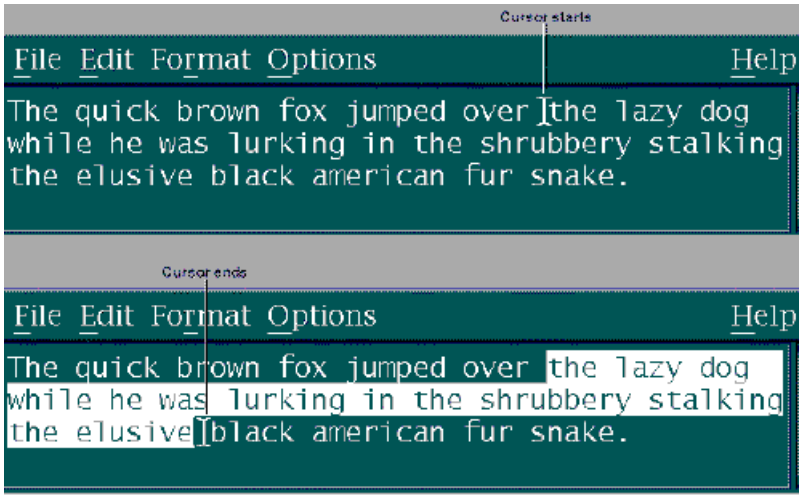
The following sections describe these techniques.

Range Technique

There are two range techniques: range swipe and range click. Both techniques act on a group of ordered elements.

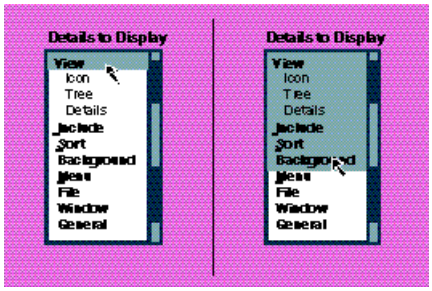
With the *range swipe technique*, the user presses the SELECT button at the beginning of the range of elements to be selected, then moves the pointer to the end of the range and releases the button. All the elements within this range are then shown with selected emphasis (see Range Swipe Technique in Text).

Figure 3 Range Swipe Technique in Text



With the *range click technique*, the user clicks the SELECT button at the beginning of the range of elements to be selected and `Shift` SELECT at the end of the range. The elements within this range are then shown with selected emphasis (see Range Click Technique in a Menu). The range click technique is useful in text for defining the beginning and ending point of a selection. People with repetitive stress injuries who find it painful to move the mouse while pressing a button will find the range click technique helpful. It is also useful when selecting a large range of elements that does not fit on one screen. The user can click on the beginning point, then scroll to the end of the range and click on the ending point.

Figure 4 Range Click Technique in a Menu



For more information, see the Range Click Technique and the Range Swipe Technique reference pages.

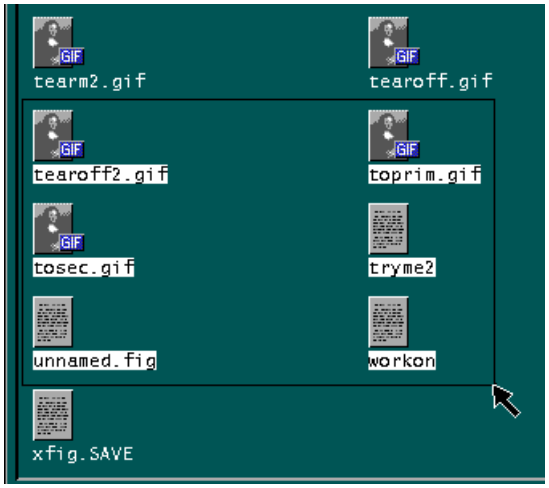
Area Technique

There are two area techniques: area swipe and area click. They have the same relationship to each other as the range click and range swipe techniques. Both area techniques act on a group of contiguous elements (see Area Technique).

With the *area swipe technique*, the user presses the SELECT button on the location where the boundary is to begin and moves the pointer, creating a framing rectangle (or marquee) between the two corners (some implementations might have a boundary shape other than a rectangle). The area swipe technique is useful in containers and graphics scopes.

With the *area click technique*, the user clicks the SELECT button at one corner of an area, then clicks `Shift` SELECT at the diagonally opposite corner of the area, encompassing the elements in a rectangle. The elements within this area are shown with selected emphasis.

Figure 5 Area Technique



For more information, see the Area Click Technique and Area Swipe Technique reference pages.

Touch Swipe Technique

The *touch swipe technique* allows a user to select noncontiguous elements one at a time by *''touching''* each desired element. To perform a touch swipe selection, the user presses the SELECT button on the screen background and moves the cursor. As the cursor moves across the screen, each *''touched''* element becomes selected and is shown with selected emphasis. The touch swipe technique is useful in containers and graphics scopes.

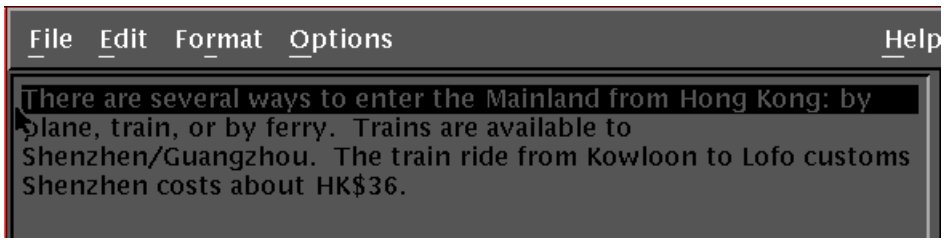
For more information, see the Touch Swipe Technique reference page.

Advanced Group Selection Techniques

There are two advanced selection techniques: margin and multilevel. These techniques enable the selection of groups of elements without directly interacting with the elements themselves.

The *margin technique* is available for selection scopes that have margins, such as text. When margin selection is in effect, the user performs a selection technique, such as the point technique, at the margin of the scope of selection (see Margin Technique). This results in the selection of a related group of elements within that selection scope. For example, in text the left margin may contain a margin element for each line. A user could use the point technique and click the SELECT button at the left margin of a line to select that margin element, which in turn would select all the characters on that line. When combining the margin technique with the range swipe technique, pressing the SELECT button at the left margin and swiping down the lines in the range would select the lines from the margin and the swiped margin elements, which in turn would select all the characters on those lines.

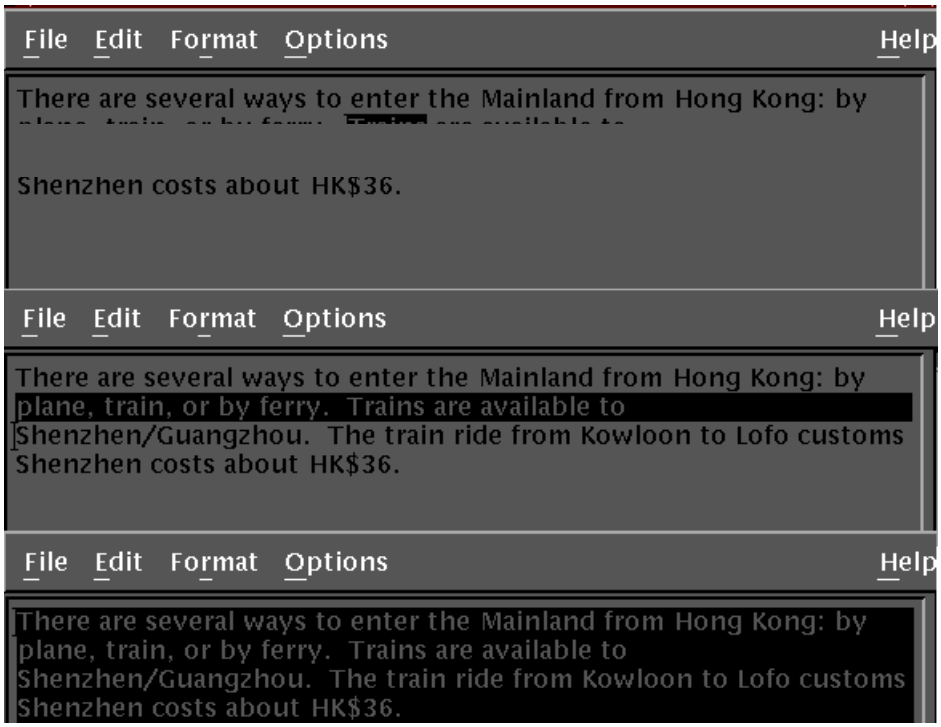
Figure 6 Margin Technique



The *multilevel technique* is available for selecting groups of elements that have some sort of hierarchical structure. The user can click either

multiple times on text or adjacent to an element. With the first click, that text or element is selected. With each subsequent click, higher–level groups of adjacent elements are selected (see Multilevel Technique). For example, a SELECT double–click (two clicks) would select a word, and subsequent clicks would select lines, paragraphs, sections, and so on. The user can combine this technique with the range techniques to select a hierarchical range of text or elements quickly. For example, in text SELECT double–press (two presses) and move would select a range of text one word at a time.

Figure 7 Multilevel Technique

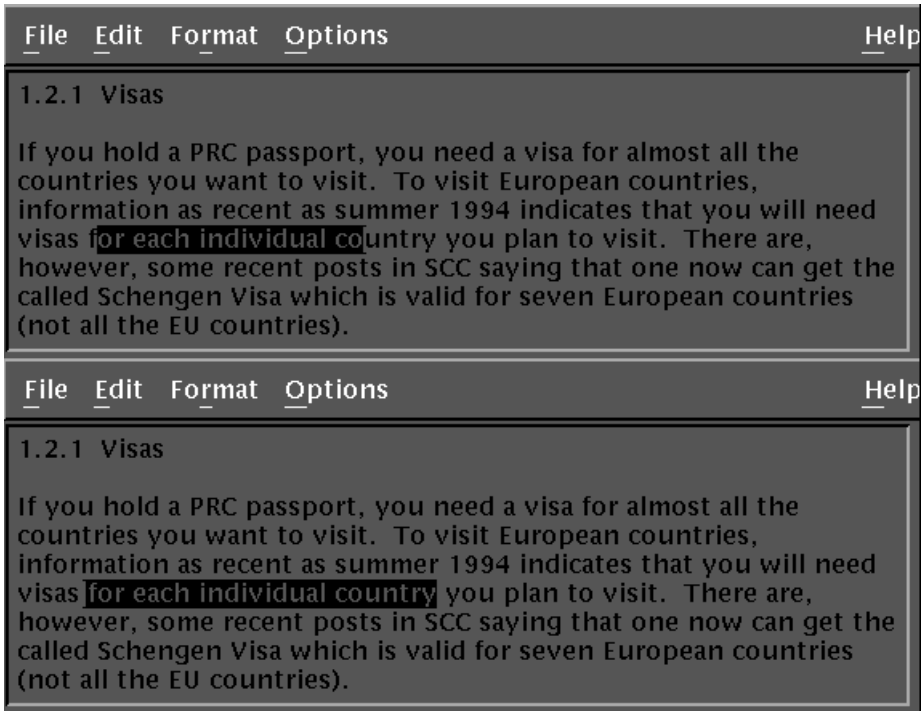


For more information, see the Margin Selection Techniques and Multilevel Selection Technique reference pages.

Adjustment Selection Techniques

After an individual or group element has been identified for selection, a user can change the existing selection by either enlarging or shrinking the region of selected elements (see Selection Adjustment). The *adjustment technique* involves using the ADJUST button, which the user can perform with `Shift` SELECT or TRANSFER. (Refer to Keyboard Model and Key Bindings for a description of mouse models.) There are two adjustment techniques, adjust click and adjust swipe. These have the same relationship as the range and area techniques described earlier.

Figure 8 Selection Adjustment



Adjust Click Technique

The *adjust click technique* either enlarges or reduces a range or area, depending on the location of the point relative to the selection. The user adjusts the selection by clicking the ADJUST button at the point where the user wants the selected range or area to be extended or reduced. The affected elements are then shown with or without emphasis, as relevant. For example, if a user has already selected an area of elements, then wants to reduce the number of selected elements, the user can click the ADJUST button on the point within the existing selection area. The elements still contained within the area remain unemphasized. However, the elements falling outside the new area are deemphasized. The range or area click techniques are the equivalent of initially using the point technique to select an element, then moving the mouse to another point and using the adjust click technique.

The effect an adjustment has on a selection depends on the various selection policies (in particular, the adjust policy) in effect. Refer to the Selection Policies reference page for a description of the various selection policies.

With the touch swipe technique in select mode, clicking the ADJUST button on a particular element adds the element to the selection.

For more information, see the Area Adjust Click Technique reference page.

Adjust Swipe Technique

The *adjust swipe technique* is similar to adjust click, except that after pressing the ADJUST button, the user can move the mouse to continue adjusting the selected region before releasing the button.

The user can also use the adjust swipe technique with advanced selection techniques. For example, after selecting some text, the user can double−press the ADJUST button or move to extend the selection one word at a time.

For more information, see the Area Adjust Swipe Technique reference page.

Keyboard–Based Selection Techniques

As with mouse–based selection, depending on the selection mode that is defined, a particular selection technique can perform either a selection or a deselection. The available keyboard–based techniques are similar to mouse–based selection. This section describes the different types of individual and group selection techniques for keyboard users.

How keyboard–based selection works depends on two modes: normal and add. Normal mode is the keyboard equivalent of the mouse–based select mode. In this mode, various techniques determine the region whose elements can be selected. If there are any elements within the region, those elements are selected and all other elements are deselected. Normal mode has three variants: standard normal mode, text normal mode, and graphics normal mode, which correspond to the type of cursor used.

Add mode is the keyboard equivalent of the mouse–based toggle mode. In this mode, various techniques determine the region whose elements can be toggled. Specified elements have their selection states toggled, and the selection states of all other elements are unaffected, unless selection policies are in effect that change this behavior.

Individual Techniques

There are two individual selection techniques: point and browse. The normal and add modes affect their operations. The following sections describe point and browse techniques.

Point Technique

To perform the point technique, the user navigates to the element and presses `Select`, `Space`, or `Ctrl Space` on the element. (The exception to using just `Space` is in text, where a space is inserted instead.) In normal mode, if the cursor is on an element, that element is selected and all other elements are deselected. In add mode, if the cursor is on an element, the state of the element is toggled, and all the other elements are unaffected, unless selection policies specify otherwise.

For more information, see the Point Technique reference page.

Browse Technique

To perform the browse technique, the user navigates to the element and selects it. This technique is available only with standard normal mode and text normal mode.

For more information, see the Browse Technique and Selection Modes reference pages.

Group Techniques

In normal mode, there are two kinds of group techniques for selecting a group of elements, as follows:

Range and area swipe techniques

> User selects a group of contiguous elements within a range or area by keyboard swiping.

Range and area click techniques

> User selects a group of contiguous elements that are within a range or area at the endpoints of a region.

The following sections describe these techniques.

Range and Area Swipe Techniques

The range swipe and area swipe techniques use keyboard swiping to select a group of contiguous elements. Keyboard swiping applies when the cursor is not within the bounds of the current selection region. With either selection technique, the user presses a sequence of `Shift` directional keys (swipe) to select or toggle the range or area determined by the initial and final cursor location. These techniques are useful in text.

For more information, see the Area Swipe Technique and Range Swipe Technique reference pages.

Range and Area Click Techniques

Range click and area click techniques select a group of contiguous elements at region endpoints. With either selection technique, the user first presses `Select`, `Space` (except in text), or `Ctrl Space`, then navigates elsewhere via a sequence of directional keys to the desired element. The user then presses `Shift Select`, `Shift Space` (except in text) or `Ctrl Shift Space` to select or toggle the range or area determined by the initial and final cursor location. These techniques are useful in text.

For more information, see the Area Click Technique and Range Click Technique reference pages.

Advanced Group Techniques

Keyboard–based selection has advanced group techniques functionally equivalent to the mouse–based multilevel and margin techniques. The user can use the range or area swipe techniques with navigation operations to navigate over groups of elements. For example, in text a user can press `Ctrl [rarr ]` to navigate one word to the right and then press `Shift Ctrl [rarr ]` to select that word.

If the selection scope allows keyboard navigation to margin elements, margin selection from the keyboard can be used.

Adjustment Techniques

After an individual or group element has been identified for selection, a user can change the existing selection by either enlarging or shrinking the region of selected elements. The adjustment technique involves using the ADJUST button, which the user can access with `Shift Select`, `Shift Space` (except in text) or `Ctrl Shift Space`. There are two adjustment techniques: adjust click and adjust swipe. These have the same relationship as the range and area techniques described previously.

Adjust Click Technique

The adjust click technique either enlarges or reduces a range or area. Either way, the user adjusts the selection by pressing `Shift Select`, `Shift Space` (except in text) or `Ctrl Shift Space` (equivalent to clicking the ADJUST button in the mouse–based adjust technique). If the user uses the touch technique, the adjust click technique adds the element that the cursor is on to the selection.

The various click–style group techniques are the equivalent of initially using the point technique to select an element, then navigating to another point and using the adjust click technique.

For more information, see the Area Adjust Click Technique reference page.

Adjust Swipe Technique

The adjust swipe technique either enlarges or reduces the selected region. This technique applies only when the cursor is within the current swipe selection. When the cursor is not within the current selection, then the selection is considered to be a new selection. The user adjusts the selection by typing a sequence of directional keys to adjust the range and area selections.

This action is the equivalent of pressing the ADJUST button at the initial location, moving the mouse, and releasing the button at the final location.

For more information, see the Area Adjust Swipe Technique reference page.

Selecting and Deselecting All Elements

A keyboard–only user can use standard shortcut keys to select or deselect all elements, as long as the selection scope uses a count policy that allows all elements to be selected. (Refer to the Selection Policies reference page for a description of selection policies.) The user presses `Ctrl /` to select all the elements within the scope and `Ctrl \` to deselect all items in the scope. However, if the cursor is visible, is on a selected element, and normal mode is in use, the element with the cursor on it remains selected.

Mouse–Based Selection Modes

Selection scopes can be in either select mode or toggle mode. The mode determines whether identified elements are to be selected or deselected. When the user selects an item, and select mode is enabled, that item is selected and all other items are deselected. When the user selects an item and the toggle mode is enabled, the result of the selection depends on the current state of the item. If the selected item was already selected, it becomes deselected; otherwise it is selected. In other words, the state of the item is *''toggled.''* This latter action may behave differently depending on the selection policies (refer to the Selection Policies reference page).

Selection Techniques and Toggle Mode

Toggle mode affects selections made with individual and group selection techniques. How the selections are affected depends on the current selection states of the elements. For example, when the user uses the point technique to select an individual list element that is not already selected, then that element is selected. However, if that element was already selected, then this selection action would deselect it.

How toggle mode affects selections made with group selection techniques depends on the existing selection state of all of the elements in the group. In toggle mode, when the user uses the range swipe technique to select a group of list elements that are not already selected, all the elements become selected. However, if all the elements were already selected, then this selection action would deselect those elements.

In summary, when a user uses one of the individual selection techniques to select an element, and that element is not yet selected, then the element becomes selected. However, if the user selects an element that is already selected, that element becomes deselected. When the user selects a group of elements, and all the elements in that group are already selected, they become deselected. When the user selects a group of elements, and all the elements in that group are not already selected, they become selected. When the user selects a group of elements, and some of the elements are already selected and some are not, then the resulting selection states depend on the toggle policy that is in place for that selection technique, as described in the Selection Policies reference page.

Augmented Toggling

If a selection scope supports only select mode, a user cannot perform discontiguous selections by using the area or range techniques by default. This is because to make a discontiguous selection, toggle mode is needed. However, a user can access toggle mode by augmenting a selection technique with `Ctrl` when using a selection technique. For example, in a list in select mode, a user can use the range swipe technique to select an element, then press the `Ctrl` key and use the range swipe technique to add discontiguous elements, or use the `Ctrl` key to remove some elements.

Augmented toggling can also apply to advanced group techniques. For example, the user can double–click `Ctrl` SELECT on text to toggle the selection state of a word (using the multilevel selection technique).

The user must use `Ctrl` SELECT when accessing toggle mode. If the user does not, then `Ctrl` SELECT is active for focus–only navigation. In this case, `Ctrl` SELECT moves the cursor to either the element or to where the mouse button was pressed, without affecting the current selection.

Adjusted Toggling

The adjustment technique adjusts the selected region, depending on the selection mode last used for that scope. After using a selection technique in a mode that toggles elements, subsequent use of the adjustment technique changes the region of toggled elements. For example, in a list in select mode, a user can press `Ctrl` and use the range swipe technique to remove some selected elements, then use ADJUST click to enlarge the region of the removed elements.

After using augmented toggling (which is the act of holding down `Ctrl` to access toggle mode), the user can use either ADJUST or `Ctrl` ADJUST to adjust toggling.

Although a `Ctrl` SELECT click in the background does not toggle anything, subsequent adjustment techniques, such as ADJUST click, will use toggling and adjust the region to be toggled.

Choosing a Toggle Mode

When choosing a selection mode that supports discontiguous selection, you can choose to provide either toggle mode or select mode with

augmented toggling implemented as described in the previous section. When choosing between these two options, consider the following:

Using toggle mode only

> Use this mode when the user chooses a set of elements, performs an action on them, and then wants to add and/or remove a few elements from that selection before performing another set of actions on them.

Using select mode with augmented toggling

> Use this mode when the user chooses a set of elements, performs an action on them, and then starts over, selecting a completely new set of elements before performing another set of actions. In this model, select mode is the default model, but toggling is available with augmentation.

Keyboard–Based Selection Modes

Keyboard–based selection modes are specific to keyboard–based users. The three normal modes include standard normal mode, text normal mode, graphics normal mode, and add mode.

Standard Normal Mode

In standard normal mode, navigation affects the selection state. When the user selects an element, all the other elements are deselected. In standard normal mode, the browse technique always applies during navigation. The point technique achieves the same effect. However, because navigating to a point automatically selects it, it is usually unnecessary to use the point technique in this instance. There are exceptions, such as when after mouse–based focus–only navigation, the cursor lands on an unselected element.

Similarly, the range and area techniques are not available under standard normal mode. However, implementations can have range swipe, area swipe (or both), and adjust click and adjust swipe techniques available in standard normal mode.

You may use the standard normal mode in conjunction with both the element and text cursors. For an element cursor, a user can navigate in a list control and use the range swipe technique, which is a sequence of `Shift [rarr ]` keys, to select an element.

For text cursors, it is more complicated because the cursor is between characters and not on them. In this case, navigation deselects all characters and there is no selection at the cursor position. The user then must use adjustment techniques to select characters. For example, to select text in standard normal mode, a user can navigate to the beginning of a word and then use the range swipe technique, which is a sequence of `Shift Ctrl [rarr ]` keys, to select words.

Text Normal Mode

Text normal mode is used only with text cursors and is the recommended mode for text. It works like standard normal mode, except that it also supports the use of the range and area click techniques. For example, a user can navigate to the beginning of a word, press `Ctrl Space`, type a sequence of directional keys to navigate to the desired words, then press `Ctrl Shift Space` to select those words.

Graphics Normal Mode

In graphics normal mode, navigation does not affect the selection state. Implementations use this mode with graphics cursors. When this mode is used with the point technique, a user can select an element if the cursor is on it. When the user uses the range, area, and touch techniques with graphics normal mode, these techniques select elements as either ranges, areas, or touched elements. The user can also use the adjustment techniques adjust click and adjust swipe when in this mode.

Add Mode

In add mode, navigation does not affect selection. When a user toggles the elements in a specified region with any of a number of selection techniques, the elements beyond that region are unaffected. However, various selection policies can affect general add mode behavior, including how many elements can be selected at a time, whether the selected elements need to be contiguous, and so on.

Add mode also supports adjustment techniques. As with the mouse, the effect of keyboard adjustment depends on the mode that was last used in the scope. After selecting elements, the user can use adjustment techniques to adjust the region of elements selected; after toggling elements, the user can use adjustment techniques to adjust the region of toggled elements. The various add mode policies can affect the adjustment techniques. An example that illustrates add mode is a list with various selected elements.

When a selection scope is in add mode and the cursor is in the background (in other words, not on an element), if a user presses `Select` or `Space` (except in text) or `Ctrl Space` nothing is toggled. However, if the user subsequently uses an adjustment technique (such as adjust click), the technique uses toggling and adjusts the region to be toggled.

Switching Between Normal and Add Modes

If a selection scope supports only normal mode, the following limitations occur:

- In standard or text normal mode, the user cannot make a discontiguous selection.

- In graphics normal mode, the user cannot generally remove elements from a selection.

If the selection scope also supports discontiguous selection, but usually uses normal mode, the user can press `Shift F8` to switch into add mode and enable discontiguous selections. (Pressing `Shift F8` again returns the user to normal mode and vice versa.) An example of where this would be useful is in text that supports discontiguous selections. In this case, a user can press `Select`, then navigate; press `Shift Select` to select one paragraph; press `Shift F8`, then navigate to another paragraph; press `Select`, then navigate; then press `Shift Select` to add that discontiguous paragraph to the selection.

In a scope that can be edited, an operation on selected items (such as copying to a clipboard or deselecting) switches the mode back to normal.

Complex Scopes

This section describes the behavior supported within selection scopes that contain data elements and objects, elements that the user can activate and toggle, text with objects and elements that the user can activate and toggle, and active text regions.

Data Elements and Objects

In general, selection scopes contain data elements, objects, or both. The selection behavior of data elements and objects are similar. Selecting either a data element or an object selects or deselects it. The differences between the two lie in their default actions. The default action for an object is for a view of the object to be opened (or, in the case of audio objects, to be played). The default action for a data element is application specific.

Elements That the User Can Activate and Toggle

Selection scopes can contain elements that the user can activate and toggle, such as push buttons or check boxes. Although a user can select these elements, the result is no selection. Selecting such elements actually affects the element itself (such as activating it) instead of affecting its selection state. A user cannot select such elements via individual selection techniques. However, the user may be able to use a group or adjustment technique to indirectly select these elements.

Text with Objects and Elements That the User Can Activate and Toggle

Text scopes can include objects and elements that the user can activate and toggle, interspersed with characters. However, because they are contained within text, these objects and elements are treated like characters with respect to formatting. As a result, when using navigation, the text cursor stops immediately before and after these noncharacter elements. The text cursor changes to an element cursor in between these noncharacter elements. In multilevel navigation, each noncharacter element is treated like a word.

Active Text Regions

Textual scopes that include only character elements can treat a contiguous group of characters as an active text region, which is specially highlighted and which acts like an object or element that the user can activate.

When a user selects the characters in the active region with the standard press, move, and release of the select button starting in an active region, the operation always performs the usual selection of text. However, clicking the SELECT button in an active region activates that region. Performing this same operation in an object region selects the text in the entire object region. Double–clicking SELECT displays the corresponding object. Also, when the cursor is in an active region, pressing `Ctrl Enter` (or `keypadEnter`) performs the default action, which is as follows:

- In an active region, the selection invokes the corresponding action.
- In an object region, the selection displays the corresponding object.

In either case, if the user presses `Enter`, a new line is inserted. If the user presses `Select` or `Ctrl Space` (with or without `Shift`), text is selected.

Data Transfer

The user interacts with elements, such as objects, data, and menu items, to perform tasks. This chapter describes the basic interactions the user performs and discusses the techniques for accomplishing those interactions, including drag and drop and clipboard transfer.

Transferring Data

Data transfer refers to how a user manipulates data, including:

- Moving elements
- Copying elements
- Creating elements
- Deleting elements
- Sharing elements
- Linking elements

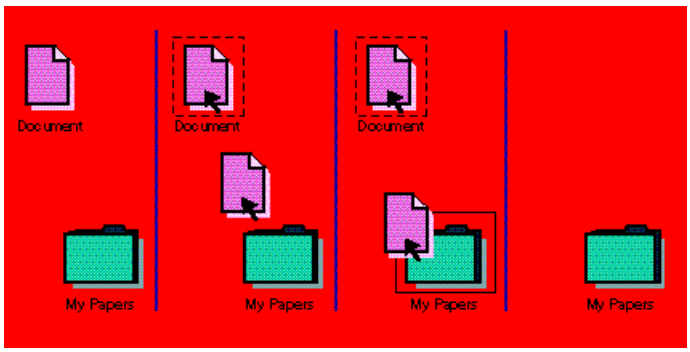
This section describes these data transfer interactions. Using Transfer Techniques describes the transfer techniques.

Moving Elements

A user moves elements to change their location. For example, the user can move a file to a new directory, move a graphic to a different position in a piece of artwork, or move text to a different section in a letter.

The user moves an object by pressing the SELECT button, dragging the object under the pointer, and releasing the SELECT button as shown in Moving an Element .

Figure 1 Moving an Element

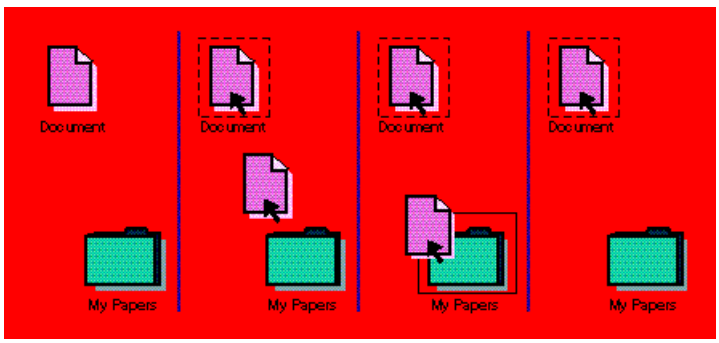


Copying Elements

A user copies an element in order to work with a duplicate while keeping the original element intact. For example, the user might copy a file into a different directory (or folder) as a backup of the original, copy a graphic from one piece of art into another as a logo, or copy text from one letter into another letter.

Copying an Element illustrates copying an object from one folder to another folder.

Figure 2 Copying an Element

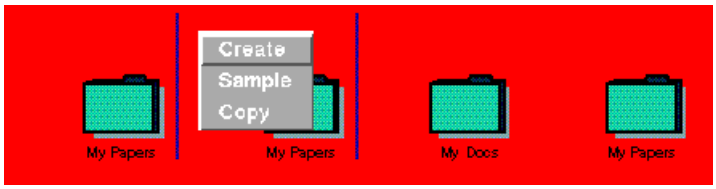


Creating Elements

A user can create new elements from existing elements. This is different from copying elements in that new information is generated at the time of creation. For example, a user can create a new memo from an existing memo and use new, system–generated information, such as the time and date, or add a name and address to the heading of the memo. A user can create an object by using the original object as a template or by requesting a new object from an application.

Creating a New Element illustrates creating a new folder element.

Figure 3 Creating a New Element

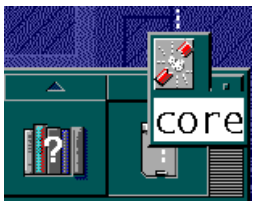


Deleting Elements

Deleting elements removes unwanted elements. For example, a user can drag a document to a trash can icon to delete it or press the `Delete` key to delete a character in the text of the document.

Deleting an Element illustrates deleting an element.

Figure 4 Deleting an Element



Sharing Elements

A user can share an element to make it accessible from multiple locations. For example, the user can share a spreadsheet in a composite document. When the user simply copies the spreadsheet, any changes to the original are not reflected in the copy. When the user specifies the spreadsheet as a shared element, any changes in the original are reflected in the shared version.

Linking Elements

Linked elements have established relationships between them. The exact functions of the links can vary. For example, your application may let the user create a link in the target element to a selected source object or to a region of selected data in the source. Within the target, you can represent the link as an icon. The user can then use the link to view the current contents of the source object or the selected region. Alternately, the application can immediately display the link in the target element as a shared view of the object contents or the selected region.

You should associate selected data in the source object with related data (which is often of a different type) in the target data. Changes to the source data are reflected in the target data (and possibly vice versa). For example, when the user links a graph object to a spreadsheet object, any changes made to the graph will cause changes to the spreadsheet data.

Create a hypertext link in the target to information selected in the source. This link will allow the user to navigate from one to the other. For example, a user might create a link within a letter to a graph that is in a different file. Any changes made to the graph within the letter are also made to the graph at its source.

Using Transfer Techniques

Depending on the number of intervening steps the user must take to complete a transfer operation, transfer techniques span two categories: direct manipulation and indirect manipulation.

Direct manipulation

Allows the user to perform actions on elements by interacting directly with the elements. Dragging an element with a mouse and dropping it onto another element is one form of direct manipulation.

Direct manipulation is similar to interacting with things in the real world; when users need to throw something away, they simply pick it up and put it in a trash can. Similarly, users can drag a file and drop it on a trash can icon in the interface. Direct manipulation is usually more efficient than performing the equivalent actions through navigating menu items.

Indirect manipulation

Allows the user to interact with an element through controls (push buttons, dialog boxes, and so forth) and menus.

By using a keyboard exclusively, a user can obtain results equivalent to those available through direct manipulation.

Direct and indirect manipulation represent two ends of a range. In reality, manipulation techniques often fall somewhere between direct and indirect manipulation. For example, using a pop−up menu is more direct than using other kinds of menus, but it is less direct than dragging an element.

Transfer techniques fall into the following range, from direct to indirect manipulation:

1.Drag and drop
2.Primary transfer
3.Quick transfer
4.Clipboard transfer

Drag and Drop

Drag and drop provides a quick and simple method of transferring data. The technique is called drag and drop because it involves moving an element from one place (dragging) and leaving it at another (dropping). The user can drag and drop single or multiple elements.

The result of a drag and drop depends on the relationship between the source element and the target element.

Source element

A source element is usually an item that contains information to be transferred.

Target element

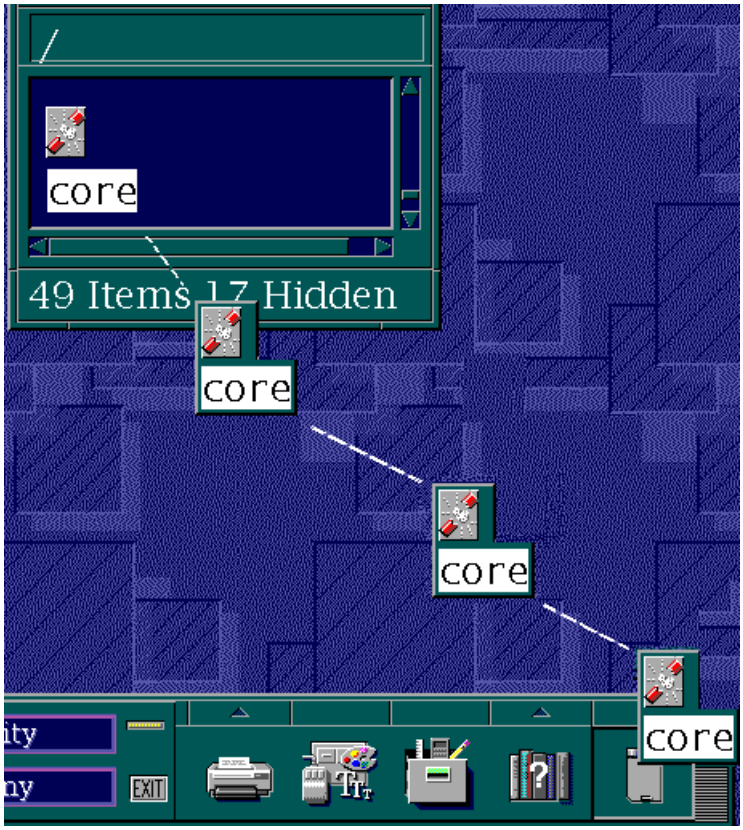
A target element is usually an item that the user is transferring information to.

Within a collection (set of elements), the user can either drag a selected subset of elements, a single unselected element, or the entire collection depending on the following:

- In text−like collections, initiating a drag in a selected region drags the text selection (including all pieces if it is discontiguous).
- In list−like and graphics−like collections, initiating a drag on a selected element drags the entire selection.
- In list−like and graphics−like collections, initiating a drag on an unselected element drags just that element. If the collection contains a selection, the selection must not be affected (except if the drop occurs in the same collection).
- In list−like and graphics−like collections, initiating a drag in the background of a contiguous selected region drags the selection.

Drag and Drop illustrates dragging a file into a trash can.

Figure 1 Drag and Drop



For more information, see the Data Transfer, Direct Manipulation, and Drag and Drop Transfer reference pages.

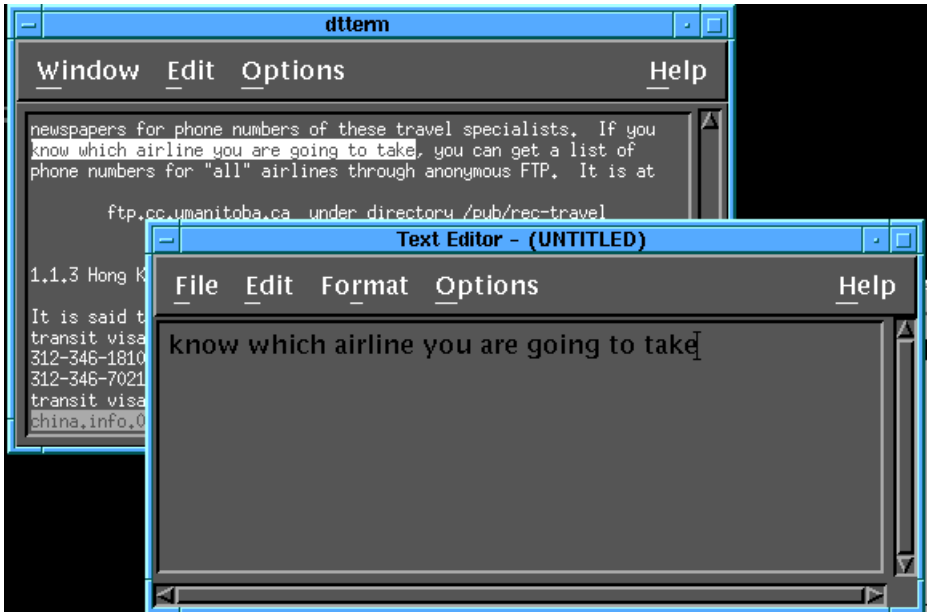
Primary Transfer

Primary transfer allows the user to transfer a primary selection from a source directly to a destination without dragging it and without using a storage mechanism, such as a clipboard. There are three primary transfer operations: primary copy, primary link, and primary move.

For example, a user can select a paragraph in help text, move the mouse pointer to the target area in another window, and transfer the paragraph by pressing the TRANSFER button.

Primary Transfer illustrates a primary transfer.

Figure 2 Primary Transfer



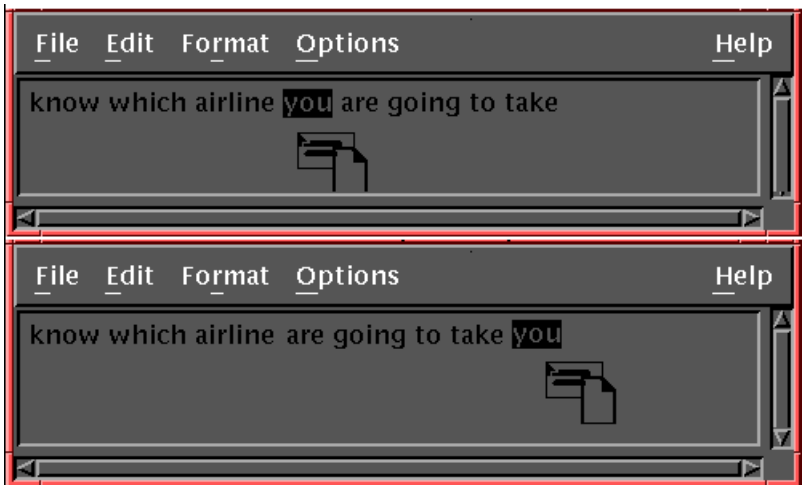
For more information, see the Primary Transfer reference page.

Quick Transfer

Quick transfer allows the user to make a temporary selection that does not affect the current selection within the scope of selection. The user can then immediately copy, move, or link that selection to the insertion point in the control in which the user is interacting. There are three quick transfer operations: quick copy, quick cut (or quick move), and quick link.

Quick Transfer illustrates a quick transfer.

Figure 3 Quick Transfer



For more information, see the Quick Transfer reference page.

Clipboard Transfer

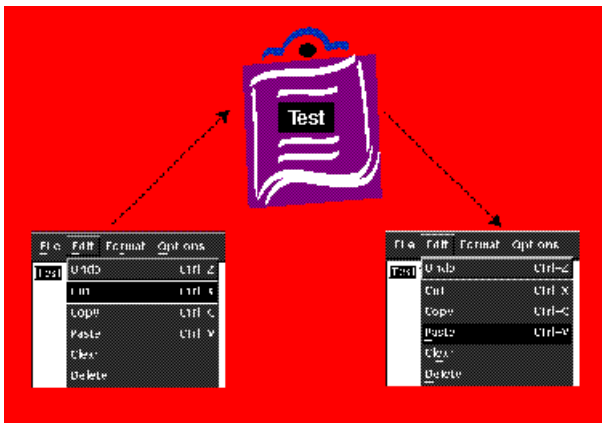
The *clipboard* is an area of storage provided by the operating system to hold data temporarily. A user can cut, copy, and paste elements to and from the clipboard. For example, the clipboard can hold a single line of text or an entire document, a single data record or an entire database, a single line segment or an entire graphic.

The user can perform the clipboard transfer operations cut, copy, and paste from the Edit menu. Standard keyboard bindings must also be available in every editable data collection.

Except when necessary to prevent the corruption of data, you should not restrict the user from placing any elements or parts of elements onto the clipboard.

Clipboard Transfer illustrates a clipboard transfer.

Figure 4 Clipboard Transfer



For more information, see Keyboard Model and Key Bindings and the Clipboard and Cut, Copy, Paste reference pages.

Fundamental Design Principles

The primary goal of the interface developer is to create a design that is easy to use and consistent across applications. A user who learns one application with a Motif interface should be able to learn to use other such applications quickly. The interface should also:

- Increase productivity
- Increase user satisfaction
- Reduce user errors

This chapter discusses Motif user interface design principles, which are based on principles of human behavior, the experience of users and researchers, and the results of usability testing.

Occasionally you must choose among design principles. Factors outside your control may lead you to favor one principle over another. For example, cost, performance, and usability concerns sometimes conflict. You must balance these concerns according to their effect on the user and the user's tasks.

The following sections discuss the categories of general design principles:

- Placing the user in control of the interface
- Reducing the user's memory load
- Promoting consistency

Placing the User in Control of the Interface

Users should be able to interact with the computer and feel that they are in control of their tasks. The following design principles address user control:

- Avoid program–driven sequences that prompt the user through fixed steps.
  For example, instead of providing fixed steps in which the user can choose only from a set series of choices and consequences, let the user move throughout the interface, choosing from a range of choices that can lead to numerous subsequent choices and consequences.
- Provide alternatives.
  Let the user take alternative courses of action; do not limit a user's capabilities by imposing any preconceived notions of the *''correct''* sequence for accomplishing a task.
- Design with a *''no user errors''* philosophy.
  Use informational messages or help information to represent errors as application limitations. Avoid representing events as errors that could be attributed to the user.

Adopting the User's Perspective

Effective design starts with adopting the user's point of view. Application developers tend to see an application as the implementation of functions; users see an application in terms of the tasks they need to perform.

Good design is rooted in an understanding of the user's tasks. Well–designed applications solve users' problems, make their work easier, and offer them new capabilities. The following design principles address the user's perspective:

- Involve users in the design process.
  Input from users can help determine both appropriate functions and the methods for presenting them. Involve users as early as possible in the design process because as the design progresses, the possibilities for change decrease.

  You do not need a working prototype to involve users. You can involve users while you write specifications. At this stage, you can watch users work in order to understand the environment in which your application will be used. Talk to users about their work, their current tools, and their goals for new tools. For example, if you are designing an application to create and display charts and graphs during meetings, attend meetings at customer sites, see how charts and graphs are used, and interview meeting participants to learn what they would like in a new tool. Once you have a working prototype of your application, invite users to test it to see if your interface meets the goals you established for it.
- Use your application in real situations.
  Using an application provides critical insights into user interface problems. Before you create the interface for your application, use similar applications, even competitive applications, to help you understand the user's tasks. Observing the user's reaction to various interfaces will also provide important feedback. Acquiring experience with the application may be difficult and time consuming, but it is a worthwhile exercise.

  When you use your own application, you become a user yourself. However, be aware that not all users think like you. You may not represent the user for which the application was designed.

Providing Modes

A *mode* is a state of the application in which only certain actions are available to the user. That is, modes restrict the user's options and limit an application's response to the user's actions.

Modes require additional learning, slow down users, and create confusion when the same user action causes different results in different modes. However, modes are useful in a number of situations. Modes extend the capabilities of input devices by letting several actions be accomplished with the same technique, key, or button; and they help an experienced user perform a series of actions quickly. Modes are also useful when there are a set of steps that must be performed in sequence, or if the user does not have permission to perform certain actions.

Provide modes when:

- The user must make a specific set of choices
- The user must press modifier keys, such as `Select`

The following sections contain more information about using modes.

Modes as Choices

You can establish a mode to take effect when the user makes a choice. A tool can represent that choice.

Tools are especially useful in drawing or imaging programs. For example, when a user chooses an eraser tool (activating eraser mode), presses the SELECT button, and moves the pointer within the drawing area, the area of the drawing over which the pointer passes is erased.

The following design principles address mode choice:

- Choose a clearly understood text or graphic label for the mode choice and provide appropriate help.
- Provide an appropriate visual (or audible) cue to help the user determine the current mode.
  Usually, applications change the pointer or cursor within the control to indicate the current mode. Consider the following when designing visual cues:
  - Change the pointer to reflect the tool that the user has chosen. For example, when the user chooses an eraser tool, change the pointer to include an eraser graphic.
  - In text controls, provide an I–beam cursor when the control is in insert mode and a block cursor in replace mode.
  - In list boxes, provide a box element cursor in normal mode and a dashed element cursor in add mode.

  Remember that users may be unaware of the mode they are in unless there is a clear visual cue. Without such a cue, users may be upset when an action leads to an unintended result or no result at all.
- Make sure that the interface indicates to the user how to exit from a mode and that it does so consistently.
  Note that users can exit from a mode in two ways: automatically after completing some action or by picking another mode. For example, in a drawing program, the user may choose rectangle mode to draw a rectangle. If the application exits from the rectangle mode automatically, the user must choose the rectangle mode again to draw another rectangle. If the application does not exit automatically, the user can draw another rectangle immediately or choose another mode.

  Whether to have your application exit from modes automatically depends on the function of the application. Consider basing your decision on a thorough task analysis. In many cases, users appreciate the opportunity to configure the exit mode themselves. Whatever your decision, design all of the tools in a toolbox to exit in the same manner.

  When the application does not exit from a tool mode automatically, design the first tool in a toolbox to display a pointer graphic and establish a standard selection–based mode (that is, one in which the SELECT button is used for selection). For example, in a drawing program, the standard mode could be used to select part of a drawing.
- Keep the scope of the mode narrow and allow the user to interact with other parts of the application while the mode is in effect.
  For example, when the user chooses an eraser tool within the drawing area of a drawing program, interactions with the remainder of the interface should behave as usual.
- In most situations, do not define a mode that automatically executes an action on selected elements.
  For example, do not define a delete mode that automatically deletes every file the user chooses from a list until the user exits from the mode. Instead, avoid using a mode entirely by allowing the user to first select the files from the list and then choose a delete action.
- Do not use a mode when you can use standard techniques.
  For example, do not design a text control that assigns standard printing characters to navigation and editing functions while in an edit mode, such as D for deleting the next character or B for moving the cursor back one character. Instead, use standard nonprinting characters, such as `[larr ]` to go back one character or `Delete` to delete the next character; or, use a modifier key with a character key such as `Ctrl B` to go back a character.

Modes with Modifier Keys

Within a control, pressing a modifier key can have different effects. For example, in a scope of selection that uses an extended selection policy, augmentation with `Ctrl` switches the scope from select mode to toggle mode. That is, ordinarily, pressing the SELECT button and then moving the pointer selects a group of elements, deselecting the remaining elements in the scope. If `Ctrl` is held down, however, the selection state of the elements is toggled, and the remaining elements in the scope are unaffected.

The following design principles address using augmentation with modifier keys to define a mode:

- Design the augmented mode to be a clear variant of the unaugmented mode.
- When appropriate, use augmentation that is consistent with its use in standard Motif controls.
  For example, when using the keyboard for navigating or for changing values, the user can press `Ctrl` to increase the appropriate increment, such as changing the scrolling increment from a paragraph to a page.
- Use augmentation with tools only if you can augment a number of tools similarly.
  For example, in a drawing program, once the user chooses a tool, provide a modifier key that determines whether to fill a drawn shape or to switch colors. Users often find it easier, however, to have additional tools rather than to use augmentation with tools.
- Be cautious about simultaneously supporting multiple augmentations, that is, allowing different combinations of modifiers to have different effects.
- Use visual cues to indicate the current mode.
  Providing a visual cue with modifier keys is not as important as when users choose a mode because users usually remember when they are pressing modifier keys. However, if combinations of modifier keys have different effects, a visual cue might help convey this information.

Making Actions Reversible

The results of a user's actions should be reversible. The following design principles address reversible actions:

- Let users learn by exploring: let them try actions to see results and undo actions if the results are not what they expected.
  For example, if a user merges two lists, then decides that the merged list is not appropriate, the user should be able to choose Undo (or press `Undo`) to return the two lists to their original format.
- Design your application so that the user can redo any action that has been undone.
  An action should not cause irreversible consequences. Thus, continuing the previous example, if the user that just undid the work done to the

lists needed the merged and sorted lists after all, choosing Redo to restore the previous work is preferable to re−creating the work.

- Inform the user if an action cannot be reversed and give the user an opportunity to choose some other action.
  Your application might display a message describing the outcome of a particular action and indicating alternative actions. For example, if the user attempts to erase a diskette, your application should display a message that tells the user that if the action continues, the user will not be able to retrieve the information on the diskette. Always inform the user *before* the action takes place, not after. The message should also indicate alternative actions that the user can take, such as stopping the action or replacing the diskette in question with another diskette.
- Provide as many reversible actions as possible.
  Determine which actions can or cannot be undone. For example, consider which interface actions users might need to undo and provide them as reversible actions whenever possible. Also, provide support to undo as many actions in a series of actions as possible.

For more information on making actions reversible, see the Undo, Redo, Repeat (Action Choices) reference page.

Providing Immediate Feedback

Immediate feedback lets the user assess whether an action has resulted in the desired response. Feedback elements include in−progress messages, information messages, status area, and pointer shape. The following design principles address feedback:

- Make the results of the user's actions immediately obvious. If the results are not as expected, the user should be able to choose an alternative action at once.
  For example, when a user is using a group selection technique such as area technique, show selected emphasis on each element as it becomes included in the selection area. Do not make the user wait until the entire selection operation has completed to see whether the intended elements were included.
- If the results of the user's actions cannot be made obvious immediately (for example, if a network delay interferes), provide the user with this information.
  If the user's action cannot be processed immediately, let the user continue to interact with the application, if possible. If the user's action must be processed before the user can continue within the application, provide feedback and let the user continue to interact with other parts of the interface.

For information on feedback elements, see the In−Progress Message, Information Message, Pointer (Predefined), and Status Area (Area) reference pages.

Keeping Interfaces Flexible

If an interface is complicated, users become confused. Flexibility within an interface can accommodate different levels of experience, different styles of interaction, and different user needs. The following design principles address interface flexibility:

- Provide different ways for users to access application functions and accomplish their tasks.
  For example, provide access to a function through a pull−down menu, direct manipulation of an element, a mnemonic key press, or a shortcut key. However, keep in mind that too much variety can cause confusion.
- Design your application so users can configure the interface.
  For example, allow users to reformat the interface by moving elements such as push buttons and menu items, defining macros, and changing input device bindings.

  Letting users change settings and select personal preferences enhances their sense of control and encourages them to take an active role in understanding an application's functions.
- Provide a way for users to proceed at a comfortable pace, learning as much as necessary to accomplish the task at hand.
  For example, allow the user to access related help topics on text modes while working in a text−entry field.
- Provide a way for users to go beyond the basic level of knowledge required for frequently used features.
  For example, let the user expand windows by choosing the More choice for additional choices and elements.
- Design the application to encourage exploration so that, as a user's expertise increases, the user is able to discover and use the application's more advanced features.
  For example, the application might initially display simplified menus that contain only those choices that a novice or casual user would use. As the user becomes more experienced, provide a mechanism for allowing the user to display complete menus of all the application's choices.
- Rely on visual cues and avoid making users type extensively or remember details.
- Provide hidden mechanisms, such as shortcut keys and condensed sequences of steps. Be sure that they are easy to discover and learn.
- Provide ways to remove some visual cues.
  For example, an experienced user should be able to turn off the display of certain kinds of information that are not required.
- Ensure that the interface serves the needs of its primary users first, but design it to be flexible enough to accommodate a range of users.

For additional information about designing for users with disabilities, see **Designing for Accessibility** .

Reducing the User's Memory Load

The following sections provide information about how to design an interface that does not tax the user's memory.

Relying on Recognition

The user should never have to rely on memory for something an application can *''remember.''* The following design principles address recognition:

- Present alternatives and let a user choose from among them.
  People are better at recognition than at recall, so provide visual alternatives. For example, provide lists of items, such as choices in a menu. The user can recognize choices in a menu without having to recall commands or their syntax.
- Provide reminders to help a user keep track of the task at hand.
  For example, provide visual cues, such as emphasis, gauges, or textual cues. Emphasis reminds users that they are interacting with an element. Gauges remind users that a process is under way. Textual cues tell users what to remember.

Using Real–World Metaphors

When interface elements resemble their real–world counterparts, users transfer knowledge gained in other areas to the computer environment and learn to use applications more quickly. For example, in the real world files are stored in folders and new mail is put into a mailbox; therefore, design your interface elements to resemble their real–world counterparts.

To ensure that metaphors are correctly interpreted, remember that they are evaluated by individual users. For example, an application designed for children should have metaphors that a child can relate to.

When an element represents an abstract notion that does not have a real–world counterpart, provide a representation of the element that helps the user visualize and remember relationships.

Using Progressive Disclosure

Design the interface so that the most necessary and commonly used functions are prominently featured and readily accessible. Consider hiding more sophisticated or less frequently used functions from immediate view. For example, a user should be able to find help immediately upon looking at a window, but you can place the choices for changing the presentation of data in a cascaded menu under the View menu.

Employing Visual Clarity

Using good visual design principles ensures clear visual communication and improved ergonomics. Well–designed interfaces reduce clutter and prioritize visual elements. For more information about the visual presentation of the interface, see Visual Presentation Principles .

Promoting Consistency

Consistency helps a user transfer knowledge from one application to another. The following design principles address consistent user interfaces:

- Develop paradigms that provide for identical implementation of common functions throughout, and across, applications.
  For example, the Motif guidelines specify that a user should be able to use the same technique for editing text, regardless of where the text appears.
- Ensure that controls are consistent.
  Determine whether making one control consistent within an interface will affect the consistency of other controls. For example, user interface controls might be consistent in shape, location, or color, while others might be consistent in interaction techniques.

  Remember that consistency is a means to an end −− ease of learning, ease of use, and reduction of errors −− rather than an end itself. Sometimes it is impractical or impossible to be completely consistent. In that case, make consistency compromises based on your knowledge of the user.

Sustaining the Context of the User's Tasks

Users can become confused if the interface changes constantly while they work. The following design principles address continuity:

- Provide cues that help the user relate an effect to its cause.
  For example, displaying the cannot pointer over a target element when the user attempts to drop a source element on it indicates a problem with the target element.
- Design your application to maintain useful points of reference while the user works on a task.
  For example, when the user adds objects to a container, the appearance of the container's window should remain the same while the appearance of the window's contents changes.
- Allow the user to complete a step or a series of related steps without having to alternate between input devices.
  For example, the user should not have to use a pointing device to scroll text while editing that text from the keyboard. The text should scroll automatically when the cursor reaches a boundary; the keyboard should have its own scrolling mechanisms, such as keys that move the cursor up or down in the window.

Maintaining Continuity Within and Among Applications

A new application or a new version of an existing application should build on the user's knowledge. Do not discount the user's experiences with other user interfaces, such as those provided in prior versions of an application or those generally accepted as industry standards. When designing your application, design the application to behave in a manner similar to previous versions or to other applications if it is part of a suite of applications. In addition, maintain visual consistency throughout your application and any related applications.

Designing Consistent Behavior

Provide consistent behavior within your application and between applications. Observe the following design principles:

- Be cautious about changing the behavior of an application or control from one version of an application to the next.
- Test a new behavior to make sure that the benefits outweigh the drawbacks of forcing a user to relearn the behavior of an application or control.
- Consider allowing users to choose either old or new behavior for an application or control.
  If an old behavior is recognized by many users, consider supporting the old behavior. Although backward compatibility is desirable, a poorly implemented application could result in a mixed model that is not easily learned and used.
- Determine whether to continue providing consistency in an existing poor design.
  Though consistency is important, it may not be appropriate to continue using a poorly designed interface. Weigh the needs of experienced users against the need for updated design.

Maintaining Visual Consistency

Maintain visual consistency throughout your application and between applications, if possible. Pay attention to visual elements such as the typeface and color. Observe the following design principles:

- Choose an appropriate font and show differences in levels of information by using different typefaces within that font.
  For example, when using the Times Roman font, show differences by making the typeface bold, italic, a different size, or underlined. You can use an additional font when there is a genuine need for more contrast in the information. However, vary the fonts only when the information is significantly different or when the choices of typefaces within an available font are limited.

  Design all of one type of element to have the same font and typeface. Pay particular attention to labels, choices, and text fields. For example, design all labels to use one font and typeface.

  When choosing a set of fonts to be displayed together, choose fonts that have noticeable contrast. The contrast can be of structure (serif or sans serif), weight, form, size, or any combination of these. Do not choose two serif fonts that are similar in size and weight to indicate different information.

Limit the number of fonts used in an interface to two. Choosing three fonts that work well together is difficult.

- Provide clear visual communication.
  The user should be able to clearly distinguish one unrelated element from another. For example, if two unrelated elements appear visually similar (such as both are the same color), the similar color might lead the user to believe that a relationship exists between the elements, even though it does not.

Visual Presentation Principles

You must design the elements of your interface to convey a coherent interface image. An *interface image* is composed of all the elements on the screen. If your interface does not accurately convey the function of its elements, users will try to create their own context for the tasks. However, their context might not correspond to the functions of your interface.

To develop an application interface that integrates function and design, you must employ visual presentation principles throughout the design process.

Design Methodology

Basic design methodology for creating the interface image incorporates two concepts:

- Understanding and exploiting the limits and potentials of the operating environment
  When making design decisions, be aware of rendering technologies that affect screen resolution, speed, and color and gray−scale support.
- Understanding human visual capabilities
  You must understand how to visually support the selection and grouping of elements. For example, knowledge of users' visual capabilities will determine how and where you will use patterns and anomalies in icon design.

Visual Variables

This section introduces visual design variables. The information presented is abbreviated due to the scope of this book.

The visual design variables available to you comprise the visual *''language''* you use to create your application's interface. Some variables, such as texture, have limited availability on today's systems, and are not addressed. However, the variables described can provide a wide variety of elements to create an infinite number of unique visual designs. Visual design variables are:

- Color
- Shape
- Size
- Contrast
- Anomaly

Color

You can use color in a wide variety of ways due to increasingly sophisticated technology. This section discusses only a few prominent design principles.

The computer display image is formed of pixels radiating light toward the user. As a consequence, such an image covers a wider range of contrasts than reflective images like printed pages and paintings. Computer display images have ranges from black to bright and luminous white. In addition, the color contrasts that are possible are much stronger than in traditional reflective media.

Used judiciously, color contributes greatly to the appeal and usability of an interface. Subtle contrasts are possible, but the misuse of color can have jarring effects. Also, remember that some users may be unable to distinguish colors due to disabilities. For more information about designing for users with visual disabilities, see Designing for Accessibility .

If you do not have a graphic designer available for consultation, consider the following principles for using color:

- Think of color in terms of hue, saturation, and value.
  Color is three dimensional. The color of a pixel on the screen is specified by three numbers: a level of red, green, and blue (RGB). A more intuitive, useful representation of color uses three defining dimensions: hue, saturation, and value (HSV). Developers often offer HSV versions of their color settings as a standard. The following list provides a brief description of these three dimensions:

  Hue

  Along the hue dimension, colors on a color wheel (or in a rainbow) vary from red to orange to yellow to green.

  Saturation

  Along the saturation dimension, color varies from gray to colorful. A greenish gray and a vivid green have the same hue (green) but different levels of saturation. The greenish gray has a low saturation; it is green *''watered down''* with gray.

  Value

  Along the value dimension, colors vary from dark to light. A dark red and a light red (pink) have the same hue (red) but different values (a dark one and a light one).

Designing with Color lists HSV principles for designing background and foreground elements.

Table 1 Designing with Color

Element Position	Hue	Saturation	Value
Background elements (background, window client areas)	No restriction	Low	Medium
Foreground elements (icons, text)	No restriction	Medium to high	Low and high (higher contrast)

When designing your application, plan to support a variety of hardware configurations. Take into account whether color is available on the system (that is, black and white and gray−scale design), and the number of colors available on different systems. Design your application so that colors can be viewed in black−and−white or greyscale without a loss of clarity. Remember that the same set of colors may look different depending on video hardware specifications. Always evaluate color palettes with appropriate video hardware when available.

Map the colors of an interface (or any image) into a color space. Such a visualization method helps define color relationships between different categories of interface elements. By establishing color relationships, you control the relative prominence of these elements and offer users interfaces that are more usable and make better use of the rich colors now available on state–of–the–art displays.

When specifying color for an application interface, consider the color resource (the HSV color space, for instance) as a *''credit''* of color. That is, once you use a color for a certain purpose, do not use it for another purpose. *''Spend''* your color resources judiciously.

Shape

The shape of interface elements can convey meaning. For example, an organic, rounded shape appears more inviting than a pointed shape.

Introducing different shapes provides visual interest and gains attention. Because of today's system implementation techniques, you can incorporate a variety of shapes in icons. Note that selected emphasis can emphasize different icon shapes.

A change in the shape of an element can also convey meaning. For instance, a trash can might change shape to show that it contains trash or a folder icon might open when an object is dragged over it to indicate that the object can be dropped there.

Size

Element size is relative; that is, the terms *''large''* and *''small''* are relative to what you are comparing. Elements in the interface can vary in size depending on different factors, such as meaning. You can also reflect relative status of elements by contrasts in the size of elements, as described in Contrast .

When designing your application's interface, keep in mind the various elements that will be required. For example, consider the appearance and size of window icons, the variety of monitors (including size and resolution), and the proportion of elements as they appear on the screen.

Contrast

You can use color, shape, and size alone or in concert to express the concept of contrast. Contrast ranges far beyond simple opposites, for example, mild or severe, vague or obvious, simple or complex.

Use contrast to differentiate elements. For example, two elements can be similar in some aspects and different in others. Their differences become emphasized when the elements are presented with contrast. In other words, one element might look small by itself, but it might appear large compared to small elements next to it. An element can have a similar shape to the elements around it and can have a contrasting color.

The following list describes contrast in conjunction with other design elements:

- Color contrast
  Some examples of color contrast would be light and dark, brilliant and dull, and warm and cool. Elements can be the same hue, but have contrasting saturation: one would be light, the other dark. The opposite is true as well; that is, elements can have the same saturation with different hues (light red, light blue, or light green) but still be perceived as a group because they all have the same saturation and value. Pastel elements in an image otherwise dominated by dark, saturated colors could be perceived as a group.
- Shape contrast
  Contrasting shapes are complicated because you can describe a shape in many ways. Geometric shapes contrast with organic shapes, but two geometric shapes can contrast if one is angular but the other is not. Other common cases of shape contrast include:
  - Curvilinear or rectilinear
  - Planar or linear
  - Mechanical or calligraphic
  - Symmetrical or asymmetrical
  - Simple or complex
  - Abstract or representational
  - Undistorted or distorted
- Size contrast
  Contrast of size is straightforward. That is, size contrasts such as big or small forms and long and short lines are familiar concepts and usually easily recognized.

Anomaly

An *anomaly* is a kind of contrast: the irregular introduced into a regular pattern. If a pattern of any kind is maintained and then broken, an

anomaly is created. In other words, there is contrast between anomaly and regularity because regularity is the observation of, and anomaly is the departure from, a certain kind of discipline or pattern. The element might be similar in many aspects to other elements, but an anomaly in one aspect gives it emphasis. Two examples are a set of elements that are all green with one yellow element or a set of rectangle elements with one circle.

In design, use an anomaly only when necessary. It must have a definite purpose, which can be one of the following:

- To attract attention
- To relieve monotony
- To transform or break up regularity

Visual Priority

Visual priority refers to the priority of elements, or what elements on the screen appear most important to the user. Because of their visual nature, some elements appear more important than others. The following guidelines ensure that what users perceive as most important is actually what is most important:

- Determine what information is important.
  User interviews and task analysis enable you to determine what information is important to users, the feedback they need, and the elements they look for.
- Determine the best combination of the visual variables to communicate important information.
  Visual priority changes as the user interacts with the system and moves from task to task. When an element becomes important to a task, it should become more noticeable than when it is not needed.

You can achieve visual priority with the visual variables described in **Visual Variables** . You can use visual variables alone or together. Visual Variable Priorities shows some of the ways visual variables convey priority:

Table 1 Visual Variable Priorities

Visual Variable	Priority
Color	If the background layer of less important elements is composed of neutral, low–contrast colors, adding saturation and contrast to the color of an element elevates it.
Shape	In an interface that is full of rectangles, changing the shape or orientation of an element makes it more noticeable and, therefore, more visually important. Also, adding animation to an interface element elevates it to the highest visual priority.
Size	The amount of space an element, or many elements of a type of element, occupies can be a size variable. When one type of element occupies more space than others, that type becomes the dominant element type.
Contrast	When one element is smaller or different from other elements, it is not necessarily de–emphasized. It can be an anomaly and therefore more apparent. For example, if you have a composition made up of large elements of white, black, and grey, and you incorporate one small dot of red, the eye is attracted to the small red dot because it is an anomaly of both size and color.

Visual Design Tasks

Though much of the interface image for an application is inherited from the operating environment, you can custom design a significant amount of the visual representations. The major visual design tasks include the design of static elements such as window contents, icons, and new controls as well as interactive elements such as visible cues and transitions. For more information about designing these elements, see Application Development Principles .

Application Development Principles

This chapter discusses designing user interfaces and includes:

- Developing a menu structure
- Creating windows
- Designing controls within windows
- Designing for user interaction
- Developing an object–oriented interface

Developing a Menu Structure

The menu structure uses a standard set of menu−bar items that leads to predefined pull−down menus. Choices in pull−down menus are categorized by the type of action they represent.

For more information, see the Menu Guidelines reference page.

Window Menu

The window menu provides choices that let the user change the size and location of the window and gain access to functions and settings specific to the window manager or workspace.

For more information, see the Window Menu reference page.

Menu−Bar Items

Description of Menu−Bar Items describes the standard menu−bar items.

Table 1 Description of Menu−Bar Items

Menu	Description
File	Contains choices that let the user identify a file to be used by the application or perform actions with the data it contains or represents. For example, the Open choice allows a user to choose a file to be invoked by the application; the Print choice allows the user to print the data the application contains. Generally, actions included in this menu are those that display, transfer, or store the information associated with the application.
Edit	Contains choices typically used when editing, such as Cut and Paste.
View	Contains choices that affect the arrangement and presentation of the view in the window such as Sort and Change View.
Options	Contains choices that enable the user to customize an application.
Help	Contains choices that let a user refer to the online help available for the various parts of the interface. It can also provide access to an online tutorial for a product or operating system.

Creating Windows

The following sections discuss:

- Choosing the parts of a window
- Displaying secondary windows in the correct location
- Choosing between modeless and modal secondary windows
- Conserving screen space

Choosing the Parts of a Window

The following principles address choosing the parts of a window:

- Provide a status area that displays the status of a running process so that the user can have that information in the window. For example, a window that displays a printer queue might have a status area to show when the queue is ready, stopped, or processing jobs.
- Provide an information area where the user can view general information about the window or elements within the window. Examples of what the information area could display are:
  - A brief description of the element that the cursor is on, for example, descriptions of the menu items as the user navigates through them
  - The position of the cursor, for example, Row 2, Column 3 when in a text−entry field
  - Messages that the user might not want in an information message, for example, *No misspelled words found*

For more information, see the Information and Message Areas (Area) and Status Area (Area) reference pages.

Displaying Secondary Windows in the Correct Location

Display secondary windows in the best possible location for the user. In some cases, this means that you need to let the user specify a default location for a particular secondary window, for example, a Find dialog window. The following principles address where to place a secondary window:

- If the user does not need to see the information in the existing window, display the secondary window centered on top of the existing window. Do not cover up the title bar or menu bar.
  This keeps the windows from spreading unnecessarily over the entire screen. The user should be able to see the context from which the secondary window came and also to use the menu bar to access other secondary windows if needed.

- If the user needs to see the information in the primary window, display the secondary window so that it does not cover the existing window. In many cases, you need to position the secondary window so that it does not cover any part of the primary window. You can calculate the best location by beginning to the right of the primary window and moving the secondary window in a clockwise direction around the existing window until you find enough available screen space to display the secondary window. Do not cover up the title bar of the primary window. The user should be able to see the context from which the secondary window came.

- If the secondary window contains information about a particular element within a window, display the secondary window so that it does not cover the following:
  - Any portion of the element
  - Any portion of an important related element, such as a label for the element that the secondary window is related to
  - The title bar or menu bar of the window that contains the particular element

  The user should be able to see the context from which the secondary window came.

For more information, see the Secondary Window reference page.

Choosing Between Modeless and Modal Secondary Windows

Modeless secondary windows let the user continue to work in the existing window from which the secondary window was opened. Modeless secondary windows are more flexible than modal windows because users have greater control in the interface when they can interact with many windows.

Modal secondary windows prevent the user from continuing to work in the existing window from which the secondary window was opened. Use modal windows only when absolutely necessary, for example when a communications program is downloading a file that cannot be modified until the download has completed. Even though users cannot interact with elements within the existing window, they should still be able to interact with the existing window itself.

As an alternative to providing modal secondary windows, use a modeless secondary window and make some of the choices in the primary window unavailable.

For more information, see the Secondary Window reference page.

Conserving Screen Space

The following principles address conserving screen space:

- Leave between one−eighth and one−half inch between the window contents and the window border.
- Plan grids for the known information in the windows.
  A grid allows you to create visually appealing windows that accurately reflect the intended hierarchy. You can design a grid system to accommodate any size, shape, or quantity of information.

  Grids promote consistency, allowing the user to anticipate placement of information from window to window.
- Use group boxes or separators to separate groups of related controls. For example, you can use group boxes or separators to show that several radio buttons belong together in one group.
  Use group boxes judiciously. Although using group boxes effectively groups related controls together, group boxes provide visual clutter that can distract the user, especially when you use group boxes within other group boxes. Using separators and the design layout guidelines in this book can help minimize the use of group boxes.
- Use standard typography.
  Coherent typography presents a more cohesive, organized presentation of the information and makes it easier for the user to read. For example, use a bold font for headings and a light font for labels.

  Line up similar elements and separate groups by placing a single colored line under the heading or by increasing the space between groups. Use indents to show sublevel information rather than nested group boxes. This saves screen space and keeps the windows organized.
- Design positive and negative space within the interface.
  In every composition there is positive and negative space. Positive space is the space taken up by elements such as icons and controls. Negative space is the space around the positive space. Software designers often think of this negative space as *''extra''* space. Although organizational techniques such as grid systems can help you compose a design that uses space economically, do not completely eliminate negative space because this space creates visual *''breathing room''* and enhances readability.

Designing Controls Within Windows

The following sections discuss:

- Aligning controls
- Sizing controls
- Grouping controls
- Placing and sizing push buttons
- Providing dialog and action choices
- Designing icons
- Using ellipses
- Creating new controls

Aligning Controls

When related controls are displayed next to one another within a window, align them into rows and columns. The following principles address aligning controls:

- Align controls horizontally with the baseline of the text or graphic.
- Align controls vertically with the most distinguishing features of the control. For example, align radio buttons and check boxes such that the radio button graphic or check box graphic is in alignment. To align text, use the left edge of the text in each control if the text is left–justified.
- Right align the labels with the left edge of the text–entry fields in cases where there are several text–entry fields. This makes it easier for the user to see which label goes with which field.

For more information, see the Label and Text–Entry Field (Control) reference pages.

Sizing Controls

The following principles address sizing controls:

- If the user can change the size of a window, size any controls that can be scrolled so that more of the control's contents are visible. Examples of controls that support scrolling are text–entry fields, list boxes, and containers.
- Size other controls when the user would expect the size to change or when it would be useful for accomplishing a particular task. For example, users might expect sliders to be sized larger when a window is made larger because it lets them set slider values more accurately. When sizing controls, remember to keep the controls' sizes in proportion to text size.
- There should be a minimum size for each control if controls are sized in a window. The minimum size should not be so small that it is useless to the user. When the window is sized smaller than the minimum control size, the control will be clipped. If clipping makes the application unusable, the window can display a message stating that the user should increase the window display size.
- When displaying graphics, bit maps, or icons, do not automatically resize the image so that the aspect ratio is incorrect. This results in unpredictable and often unreadable images.

Grouping Controls

Group and label controls within a window appropriately. Examples of controls that belong in a group are:

- Related radio buttons
- Push buttons at the bottom of a window
- Check boxes that have a common purpose

When grouping controls, separate individual groups from one another in the window. Some ways to separate the groups are to use:

- Blank window space
- Separator lines
- Group boxes

Placing and Sizing Push Buttons

The following principles address placing and sizing push buttons:

- Place push buttons that affect the entire window at the bottom of the window horizontally.
  Align push buttons on the left side of the window when the window is much wider than the area occupied by the push buttons. Place the push buttons so that there is the same amount of space between each push button.
- Align push buttons on both sides of the window (with equal amounts of space between each button) when the window is about the same size as the area occupied by the push buttons.
  In rare circumstances, placing all the buttons on one horizontal line causes the window to be wider than necessary to display all the elements in the window. In extreme cases, provide the user with two rows of buttons instead of making the window wider than is necessary.

  See the Push Button (Predefined) reference page for the correct ordering of the push buttons from left to right.
- Place push buttons that affect a particular control or group of controls within the window to the right of the control or group of controls. Align the push buttons vertically. If there are too many buttons to fit beside the affected controls, place the push buttons horizontally below the controls and visually separate the controls and their associated push buttons from the other controls in the window.
- Design all push buttons that are in a horizontal row to have the same height and all push buttons that are in a vertical column to have the same width.
- Design push buttons to be consistent in size, whenever possible. Consider the longest label necessary and make the buttons conform to that size. If it is not possible to make all of the push buttons the same size, size the push buttons relative to the length of their labels.

For more information, see the Push Button (Control) and Push Button (Predefined) reference pages.

Providing Dialog and Action Choices

Provide a dialog choice when the user must supply additional parameters through a secondary window before a process begins. Provide an action choice when the user's action begins immediately after the user activates the choice.

For example, when a user chooses the Print choice, it can be either a dialog choice or an action choice. A dialog choice asks for additional information such as page orientation, number of copies, and printer name. An action choice prints the document immediately, possibly displaying a secondary window to show an in–progress message.

Designing Icons

Icons are a prominent visual aspect of the interface and are what many people think of when they think of a graphical user interface. When you design your icon set, be aware that users need to work within a larger set –– the operating environment. Do not consider icons as *''advertisements''* for the interface. Design icons as functional, integrated parts of the visual hierarchy of the interface.

The following principles address icon design:

- Design your icons to work together as a family by using the same style for each icon and by conveying a consistent language of images throughout your icons.
  A family of icons builds on knowledge the user acquires when encountering different icons in the set.
- Do not design any icon that stands out from the rest of the icons, unless it is necessary for user understanding.
  An icon stands out if its use of color, style, or shape are different than those around it. This would be considered an anomaly and should be employed only where there is a genuine necessity. Color and font resources should never be hard coded.
- Limit the number of details shown in the icon.
  For most interfaces, there are between 70 and 90 pixels in which to display icon information. Because of the limited space, the amount of information that you can convey is also limited. Any extraneous details detract from your intended message.
- Design icons to be clear and readily understood.
  Strive to create icons that are easily recognized and memorable. One way to achieve this is to base your designs on metaphors from the user's environment and to create icons that are consistent with those metaphors.
- Simplify and refine the icon design to reduce clutter or visual noise in the overall interface image.
  Visual noise is annoying and distracts the user from important visual information. If visual noise occurs in the interface, the user must sift through the noise to find information. It is difficult to use an icon to convey a complex message because icons are small and are often created from simple shapes such as squares or rectangles. To convey your basic message clearly, limit details and simplify information when possible.
- Incorporate multicultural considerations in your icon design. For more information about international guidelines, see **International Design Guidelines** .

Using Ellipses

The following principles address when to use an ellipsis (...) after a choice label.

- A choice should have an ellipsis when the choice will not be carried out immediately and a secondary window will be presented to the user to further clarify the choice before the process begins.
- A choice should not have an ellipsis when the choice is carried out immediately after the user activates it, regardless of whether or not a secondary window is opened. Never display an ellipsis on a choice when activating it will not display a secondary window.

Creating New Controls

When creating new controls, remember that some controls, such as audio or video controls, have real−world metaphors that you can follow. As with other static elements, you need to be aware of the user's idea of how these elements should be presented.

Designing for User Interaction

User interactions take many forms and the interface often provides feedback or cues as users select or manipulate interface elements. The following sections discuss feedback and cues. See also Visual Presentation Principles for more information on presentation principles.

Feedback

Consider user interactions as part of an ongoing dialog between the user and the interface. Often, when people converse, the listener provides feedback to the user to indicate that the message is being received. Feedback is the application's return of a dialog (or other cue) to the user's input.

As in dialogs between people, users appreciate feedback in dialogs with the interface to help them understand what information the system has received. Therefore, design your interface to provide feedback that is subtle but consistent.

Also, provide transitions to help users follow the results of their interactions. Think of transitions as providing a *''storytelling''* aspect in the interface. In other words, use transitions to help users understand the flow from one step to the next. Transitions should be noticeable but subtle so that they are helpful to less experienced users and do not annoy experienced users.

An example of a subtle transition is graduated boxes (sometimes called zoom boxes or nested boxes) that appear when the user closes a window. They tell the user that the view has been closed and the data put away. If the window simply disappears, the user might think that the information had been deleted.

As with static elements, all the visual variables can play a role in communicating interaction cues and transitions, including size, shape, movement, and color. However, interactive cues and transitions can incorporate an additional important visual interface concept: movement. Users react instinctively to movement. For example, when a printer icon displays paper ejecting from the printer, the user knows that the print job was successfully spooled. Therefore, movement is one of the most powerful visual variables. To maintain that potency, use movement sparingly and only where absolutely necessary.

Cues

Unlike feedback, a cue may alert the user unexpectedly, for example, when a printer jam occurs.

Because graphical user interfaces are still mainly visual, visual cues are numerous. When designing a new visual cue or set of visual cues, avoid conflicts with preexisting cues. As with any type of cue, efficiency of a visual cue depends on its timing. Carefully choose when to display or remove a cue in relation to user action. You can intensify the effect of certain visual cues by combining them with appropriate audible cues.

For example, if a printer runs out of paper during printing, the audible cue attracts user attention while the blinking sheet of paper on the printer icon informs the user of the nature of the problem. The interface can indicate a paper jam by another visual cue, such as the paper on the printer icon changing color or shade.

For more information on cues, refer to Audible and Visual Interface Cues .

Developing an Object−Oriented Interface

The concept of basing an interface on objects is a step toward the goal of a user−centered design. Object−oriented interface design employs real −world metaphors to help users understand and use the functions in your application. The types of objects and the interactions required depend on the tasks to be performed. For example, a highway construction worker may need a map object with a topographical view; a secretary is more likely to need a telephone directory.

While some objects are specific to an application or task, other objects have a wider range. For example, the construction worker and the secretary may both need a calendar object. In an object−oriented environment, various objects can work together. Users can choose the objects they need and group those that are customized for the tasks they want to perform.

Object Types

You can classify objects according to the functions they perform. The Motif user interface classifies objects as follows:

- Container objects
- Data objects
- Device objects
- Composite objects

Container Objects

Many objects are capable of containing other objects. Before classifying an object as a container, take the user's perspective and determine what its primary use will be. A wastebasket object can contain other objects. Its containment behavior, however, is both temporary and secondary to its main purpose of disposing unwanted objects and data. The user is more likely to regard it as a device than a container.

A folder, on the other hand, does little to its contents other than contain them. A folder may have views that allow a user to see the contents, but the folder does not transform the contents. Therefore, a folder is generally a container.

When an object is dragged to a container that can hold it, the default result of the drop operation should be to move the source object into the container, removing it from its former location.

Data Objects

You can think of most objects as containing data. A data object is one that exists to allow direct access to information, either to be displayed or edited. A folder usually contains data in the form of its objects but its main purpose is containment. A note object, however, is meant to convey information and would therefore be a data object.

Device Objects

Device objects provide an interaction path between a user's objects (or data) and the functions that the user may want to perform on those elements. For example, a printer object can allow the user to print without menus, using only direct manipulation techniques such as dragging and dropping an object to the printer.

The default behavior for a device object should usually be to place a copy of the source object into the device or into a queue for the device, prompt the user for additional information, perform the specified operation on the copy of the source, and, unless the device is a drive, delete the copy. Exceptions include devices such as wastebaskets and shredders, which are intended to allow the user to discard data or objects, and which should allow the source to be moved instead of copied.

Composite Objects

A composite object is an object that is made up of other objects. It can easily be decomposed; that is, its control objects can be moved, copied, or deleted. It is distinguished from a simple object, which cannot be decomposed and is not made up of other objects. A composite object is composed of simple objects.

Choosing Between Simple and Composite Objects

Use the following suggestions and your understanding of objects along with the user's tasks, expectations, and needs to decide when to use a simple or composite object:

Use a simple object when:

- The user does not often need to decompose the object into its constituent parts.
- The number of composed views necessary to support a task is not so large as to be unmanageable. Six to eight views is a recommended maximum. For more information, see the View reference page.
- Each composed view will be perceived as a sensible view derived from the function and nature of the object.
- Each view presents the object's information as a whole or as filtered subsets.
- The user perceives the object as a single indivisible object, and not as a container or collection of many objects (for example, a report object rather than an investment portfolio object).

Use a composite or container object when:

- Information elements are not clearly related when presented to the user.
- There are clearly defined boundaries between the information elements that suggest separate logical simple objects.
- The information elements are not derivations of a simple object.
- The application relies heavily on containment or collection behavior; for example, the application refers to a folder that contains many different objects.
- The user will need to decompose the object and access (or manipulate) individual objects or the composite (or container) object.

Application–Oriented and Object–Oriented Development Principles

The following principles address the differences between application–oriented and object–oriented interface design.

Application–Oriented Interfaces

The main element of an application–oriented interface is the application itself. The user first starts the application, then calls files into the application to accomplish a task. The user must be aware of which application is running at any given time, which type of files the application supports, and how the operating system organizes data into files, directories, and storage media.

The tasks performed in an application–oriented environment generally have a rigid stricture; the user must complete steps in a predefined order from start to finish before going to another task. An application–oriented environment is best suited for users who do a small number of tasks repetitively. The application–oriented menu structure supports this type of interface.

Application–Oriented Menus

The application–oriented menu structure uses a standard set of menu–bar items that contain predefined pull–down menus. Choices in the pull –down menus are categorized by the type of action they represent as shown in **Application–Oriented Pull–Down Menus** .

Table 1 Application–Oriented Pull–Down Menus

Menu	Description
File	Contains choices that let the user call a file into the application or perform a set of actions on the file (or parts of it).
Selected	Contains choices that are determined by the scope of the actions they represent. Place choices that apply to selected objects or data in the view, but are not typical editing choices, in this menu.
Edit	Contains choices typically used when editing, such as Cut and Paste.
View	Contains choices that affect the arrangement and presentation of the view in the window, such as Sort and Change View.
Help	Contains choices that let the user refer to online help available for the various parts of the application.

Object–Oriented Interfaces

Object–oriented interfaces focus on the user's objects. The user decides what object to work with, then chooses actions to apply to the object, either by using menus or by direct manipulation. The user is not required to know what application is being used to work with the objects at any given time.

In an object–oriented environment, the user can decide how to structure tasks. In most cases, the user can choose the steps to be performed. The objects that need to be supplied should work together and with objects supplied by other applications. The object–oriented menu structure supports this type of interface.

Object–Oriented Menus

The object–oriented menu structure sorts choices into various menus, based on the objects or elements to which the choices apply. To satisfy the user's expectations and to make choices easy to find, be aware of the scope of each choice and place the choice in the appropriate menu. Object –Oriented Pull–Down Menus describes standard object–oriented menu choices.

Table 2 Object–Oriented Pull–Down Menus

Menu	Description
Object	Contains choices that affect the object whose view is in the window in which the menu appears. The Object menu also lets the user open additional windows that contain other views of the object. The same set of choices should also be available from the object's pop–up menu.
View	Contains choices that affect the way the object is presented in the current view and that let the user switch to a different view of the object within the same window. This menu allows the user to customize the view by sorting, filtering, or changing the format of its contents, or by changing the presentation of the menu–bar item.
Selected	Contains choices that affect the objects or data that are currently selected in the window or that affect the selection state of the contained objects. The contents of the Selected menu must change dynamically to reflect those choices that are appropriate to all members of the selected set.
Help	Contains choices that let the user refer to online help for various parts of the application.

Views

In an object–oriented user interface, applications no longer become the primary interface to the user. Instead, users work directly with objects, organizing and classifying them to suit their tasks. Views replace applications as the means of accessing or modifying information. For example, a newly installed spreadsheet processor would be accessible as a new view for every existing spreadsheet object.

There are three types of views: composed view, contents view, and properties view.

Composed View

A composed view shows an object's data in presentation (or final output) form. For example, a composed view for a word–processor document shows the formatted output as seen on a printer or previewer. Composed views usually apply to data objects.

Contents View

A contents view shows the contents of an object. For example, a contents view for a word–processor document shows the list of files that make it up. Contents views usually apply to container objects, such as in folders.

You can display contents views with various interpretations, including iconified views and detailed views. Do not be overly concerned about which category a particular view falls into. Instead, create views that convey information in a form that is meaningful to users.

Properties View

A properties view displays information about the characteristics or attributes of an object and optionally provides a way for the user to change them. For example, in a properties view for an object such as a document, the user could change the font, type size, or color of the document. A properties view should be provided for every type of object.

International Design Guidelines

This chapter describes those features of the interface that may require clarification or different guidelines when applied to various cultural environments.

This chapter first describes general guidelines for international design, including icon, symbol, and pointer design; data formats; multibyte character sets; screen text; and modularized software. Finally, the chapter presents guidelines for bidirectional language support.

General Guidelines for International Design

The following sections discuss using general Motif guidelines in an international environment.

Capitalization

The guidelines stated for capitalization in this manual are for the English language developer. When an application is developed in another language or translated to another language, the rules of that language must be followed. For example, in the German language all nouns are capitalized, no matter what their position in a phrase or sentence.

Column Headings

The space for a translated column head may need to be increased. You can create the additional space by increasing the number of rows in the heading.

Labels

The translation of a label may cause a field to be misaligned. Fields must be realigned after translation.

Labels sometimes reflect a localized or culturally sensitive format. Allow the translator to edit the descriptive text.

First–Letter Cursor Navigation

Allow a user to press the unaccented uppercase or lowercase character to access accented first letters when using first–letter cursor navigation.

For a description of first–letter cursor navigation, see First–Letter Cursor Navigation .

Shortcut Keys

Using `Alt (A – Z)` for shortcut keys can be awkward on non–United States keyboards since only a left–hand `Alt` key or a right–hand `Alt` key may be available for this function.

Sort Function

Each country has a different way of sorting and may have several ways of sorting within the country, such as in a telephone book or a dictionary. The Sort menu may have to allow selection of these criteria based on the requirements of the country or application. In addition, when lists are ordered alphabetically, as in spin boxes, the list must be reordered after translation.

Graphical Symbols

You may not always be able to design a graphical symbol (like an icon or a pointer) that adequately represents the same object or function in different countries. Cultural differences impact even seemingly universal symbols. For example, sending and receiving electronic mail is a common function, but representing that function with a picture of a mail box may be inappropriate because the appearance of mail boxes varies among countries.

Advantages of Graphical Symbols

When used correctly, graphical symbols offer the following advantages:

- They are language independent and do not need to be translated.
- They can replace computer terms that have no national–language equivalent.
- They may have more impact when used with warning text than ordinary text.

Guidelines for Using Graphical Symbols

Use the following guidelines when designing icons, symbols, or pointer shapes:

- Use an already existing international icon, if possible.
- Avoid using human body parts.
- If you include alphabetic characters, use uppercase letters.
- Avoid symbolically written material oriented for left–to–right text.
- Avoid using stars and crosses.
- Do not use humor or regional metaphors.
- Make your icons, symbols, and pointer shapes represent basic, concrete concepts. The less abstract the icon, the less explanatory documentation is needed.
- Check your icons and symbols for conflicts with existing icons or symbols for that function.
- Do not incorporate text in icons because the text will need to be translated. Translated text often expands and might no longer fit the icon.
- Test and retest your symbols and icons in context with real users.

Country–Specific Data Formats

Different countries require data to be represented in different ways. Besides the language differences, other data (such as currency) varies from country to country. The following list describes these data formats and includes an example of each:

- Thousands separators
  The comma, period, space, and apostrophe are examples of valid separators for units of thousands.

```
1,234,567
1.234.567
1234567
1’234’567
```

- Decimal separators
  The comma, period, and center dot are examples of valid separators for decimal fractions.

```
5,234
5.234
5·324
```

- Grouping separators
  Grouping may not be restricted to thousands separators.

```
400,001.00
40,0001,00
```

- Positive and negative values
  The symbols + (plus) and - (minus) can appear before or after the number. You can enclose negative numbers in parentheses in certain applications, such as spreadsheets.

```
234+
+234
234−
−234
(234)
```

- Currency
  The comma, period, and colon are examples of valid separators for currency. There can be one or no space between the currency symbol and the amount.

```
Kr. 3,50
Sch3.50
FIM 3:50
3F50
350 Pts
ESC3.50
```

- Date format
  Most countries use the Gregorian calendar but the dates may be formatted differently. Separators might be different or might not be used. The hyphen, comma, period, space, and slash are all valid separators for the day, month, and year. In numeric date formats, the month and day fields might be reversed and, in some cases, the year field might come first.

```
4/8/96 or 8/4/96 for August 4, 1996
961029 or 962910 for October 29, 1996
```

- Time format
  The colon, period, and space are valid separators for hours, minutes, and seconds. You can sometimes use the letter h to separate hours and minutes. Also, you may use either 12– or 24–hour notation. For 12–hour notation, a.m. or p.m. might appear after the time, separated by a space. You can also indicate the time zone for a given time.

```
1307
13:07
```

```
01 07
01h07
1.07 p.m
13:07:31.30 IST (Indian Standard Time)
```

- Telephone numbers

  Telephone numbers can contain blanks, commas, hyphens, periods, and brackets as valid separators. You can display telephone numbers in local, national, and international formats. Local formats vary. National formats might have an area code in parentheses. The international format does not use parentheses, but adds a + (plus) at the beginning of the number to indicate the country code.

```
(038) 473589 +44
(038) 473589
617.555.2199
(617) 555-2199
1 (617) 555-2199
(1) 617 555 2199
911
1-800-ORDERME
```

- Proper names and addresses

  Addresses can vary from two to six lines long and can include any character used in the locale's character set. The post code (zip code) can be in various positions in the address and can include alphabetic characters and separators as well as numbers.

```
Herr Dipl. Ing. Schmidt
Stolbergerstrasse 90
D-200 Hamburg 55
GER

Madame Dupont Claudette
17, Rue Louis Guerin
F-69626 Lyon-Villuerbanne
France
```

Internationalized Text Input

In some languages, users cannot type characters from the keyboard into a text control. Instead, the user must apply multiple key combinations to form the symbols that appear in the text–entry field. For these languages, you must provide a preedit area. The text is typed into the preedit area until the user indicates the text is complete; the text is then converted and sent to the text control.

You must address the following issues when providing a preedit area:

- Locating the preedit area
- Displaying the status
- Converting the preedit characters to final characters

Locating the Preedit Area

The location of the preedit area depends on a variety of software considerations. Ideally, preedit should occur in place in the text control being edited. The following list describes three preedit locating methods:

On–the–spot

> The input from the preedit area goes to the text control. This method is preferred because it is more direct for the user; however, the on–the–spot input method can be difficult to implement.

Over–the–spot

> The preedit spot is above, but separate from, the text control.

Off–the–spot

> A single preedit area is used for multiple text controls. The input from the preedit area goes to the text control that currently has the focus.

Displaying the Status

With the on–the–spot method, the text control may contain both unconverted and final characters. You should show the portions of the text that

have been converted by using a visual cue, such as a different font, color, or highlight.

Converting Preedit Characters to Final Characters

When the preedit text is converted into the final characters, there might not be enough information to unambiguously convert the text. If there is not enough information, the conversion either:

- Prompts for more preedit text
- Presents a list of possible choices
- Fails

Multibyte Character Sets

ASCII and extended ASCII character sets are single−byte character sets (SBCSs). That is, there are no more than 256 unique characters in an SBCS. The English language uses an SBCS. Some languages, like Japanese or Chinese, have a larger set of unique characters that constitute more than what a single byte can represent. These character sets are called multibyte character sets (MBCSs). The following sections address multibyte character set guidelines.

Mnemonics

When a user types a valid mnemonic, the cursor moves to the choice that the mnemonic is assigned to and the choice is automatically activated. This saves keystrokes for choices that are usually activated explicitly.

All the guidelines for mnemonics in an SBCS language apply to an MBCS language except for how the mnemonics appear. If all the letters in a choice are already assigned, or the choice consists of MBCS characters, you may choose another letter or a keyboard character, such as the comma (,). You should assign the same mnemonic to choices that appear many times throughout an application.

Applications translated from SBCS languages to MBCS languages can keep the SBCS mnemonics. Place the mnemonics in parentheses following the word.

Sort Order

For controls that specify choices be shown in a specific sort order, such as alphabetic order, the sort order may actually depend on the application. For example, in Japanese Kanji it may be better to sort by Japanese phonetic order or some other appropriate order, depending on the information in the field.

Screen Text

Well−written screen text makes an application easier for users to understand. It also makes translation easier. Use the following guidelines when writing screen text for translation:

- Write brief and simple sentences. They are easier to understand and translate.
- Write affirmative sentences. They are easier to understand than negative statements. For example, write *''Do you want to continue?''* rather than *''Don't you want to continue?''*
- Use active voice. It is easier for both users and translators to understand. For example, use *''Press the Help button''* rather than *''The Help button should be pressed.''*
- Use prepositions to clarify the relationships of nouns. Avoid stringing three or more nouns together.
- Use simple vocabulary. Avoid jargon unless it is part of your audience's working vocabulary.
- Allow space for translation. Text translated from English is likely to expand thirty to fifty percent or more in some languages.

Software Design

Designing software in modules makes translation easier. A modular application requires fewer files; therefore, fewer files must be translated. Use the following guidelines when designing software:

- Create separate modules for text, code, and input and output components that need to be changed to accommodate different markets. Also, place elements that need to be translated in separate files.
- Separate all user interface text from the code that presents it.
- Use standard registered data formats, such as ISO and IEEE.
- Use standard processing algorithms for all processing, storage, and interchange.

Use the internationalization tools on your system to create a different set of language−dependent text files for each market you are designing for.

Bidirectional Language Support

The Hebrew and Arabic languages are written from right to left. The English language is written from left to right. Consequently, bidirectionality is a necessity in any system intended for Arabic or Hebrew users.

You can design for most right–to–left languages in the same way. This chapter uses *''Arabic and Hebrew''* as a generic designator for the national language that a bidirectional application uses. Whenever special considerations apply to either Arabic or Hebrew only, they are mentioned explicitly.

Basic Rule for Bidirectional Applications

The basic rule for bidirectional applications is that you must display all pieces of data in the orientation that is correct for the user. Also, data input must be supported in the orientation that is natural for users.

National Language Use

Arabic and Hebrew applications generally must use the national language for titles, instructions, headings, prompts, and other window controls, with the following exceptions:

- English acronyms or terms are not commonly translated, for example, DOS, EXE, CICS.
- Key names must be identical to the keyboard, for example, `F1`, `Alt`, `Ctrl`, `Enter`.
- Key combinations must be displayed in English, for example, `Alt F2`.

Dynamic Language Selection

An application may allow dynamic language selection. When the user chooses a new language, the headings, messages, and commands should adhere to the new language. For right–to–left languages, the application interface should reflect these bidirectional language guidelines. When the application contains a mixture of right–to–left and left–to–right language elements, the left–to–right elements follow the unmodified guidelines provided in this book and the Style Guide Reference; the right–to–left elements follow the bidirectional language guidelines in this chapter.

If the application is multilingual, mention the language being used in the product information window. If the application allows dynamic language selection, mention the initial language in the product information window.

Orientation

Use a right–to–left screen orientation for Arabic and Hebrew applications. Thus the right side of a window is the *''begin''* side, and the left side is the *''end''* side. Most references in the guidelines for left–to–right languages are true for Arabic and Hebrew, with the understanding that the meaning of *''right''* and *''left''* are interchanged. (Note that in some cases, the meaning *''right''* and *''left''* must not be changed for Arabic and Hebrew.)

While bidirectional language interfaces seem to have exchanged the meaning of the words *''right''* and *''left,''* the physical right and left are still the same. Thus the following have the same effect for Arabic and Hebrew as for English:

- Cursor movement keys (including `[larr ]` and `[rarr ]`) must still move the cursor according to the direction of the arrow engraved on the key top. (This is also true for combinations of `[larr ]` and `[rarr ]` with `Shift`, `Ctrl`, or `Alt`.)
  Also, `Ctrl PageUp` must always scroll to the left and `Ctrl PageDown` to the right.
- Right and left buttons of a pointing device are not transposed.
- Right and left movement of a pointing device is not transposed.
- Graphics may be unaffected by the window orientation: for example, a map must still have the East on the right and the West on the left.

The appearance of an Arabic or Hebrew window is the mirror image of a corresponding English window, except that the position of the window menu button and the maximize and minimize buttons in the title bar does not change.

Orientation applies at multiple levels. An application has a general orientation for its windows. A window also has an orientation. If the window orientation is right to left, its elements are logically ordered from right to left and from top to bottom.

In a bidirectional environment, the user chooses the orientation of the screen and windows according to use, standards, or preferences.

Guidelines for Right–to–Left Screens and Windows

Use the following guidelines for right–to–left screens and windows:

- Tile, cascade, or superimpose windows with the older primary window on the right and the newer secondary window on the left.
- Order minimized (iconified) windows from right to left, beginning from the right side of the screen.
- For a right–to–left window, prioritize the positioning of associated secondary windows (like help windows) as follows: to the left of, to the right of, above, or below the application window.
  Note that an Arabic or Hebrew application can include some left–to–right windows.

  You can also use a left–to–right window on a right–to–left screen and vice versa. This means that English left–to–right applications can be used on a right–to–left screen and Arabic and Hebrew (right–to–left) applications can be used on a left–to–right screen.
- Within the same window, text and graphics may have differing orientations. For example, a geographic map may have left–to–right graphics (as for an English application), but the Arabic and Hebrew captions may have right to left graphics.
- Text–entry fields are a special case: for each field, an orientation is defined, which by default is right to left within right–to–left windows. However, you should define fields that are to receive numeric data or English text as left to right. Within a text–entry field, the keying or cursor direction may be right to left or left to right, according to the data entered: Arabic or Hebrew text, or English and numbers.

For more information, see **Right–to–Left Text–Entry Fields** .

Right–to–Left Text–Entry Fields

In a bidirectional application, each field has an orientation that is defined by the application. By default, text–entry fields assume the same orientation as the encompassing window. According to the expected contents of the field, the application must define text entry as right to left (Arabic or Hebrew textual data) or left to right (numeric data or English textual data).

Right–to–Left Text–Entry Field Differences

Right–to–left text–entry fields differ from left–to–right text–entry fields in the following ways:

- Text is right–aligned in the text–entry field.
- At initial entry, the cursor is located at the rightmost position.
- For each entry, the cursor movement is initiated as right to left: in replace mode, the text cursor moves left to the next entry position; in insert mode, the cursor and all characters to the left of the cursor are shifted one position to the left.
- If scrolling is available in the text–entry field, the rightmost field positions are initially visible. When the cursor moves to the left, the information scrolls to show more characters on the left side. In summary, the beginning of the information is on the right side, its end on the left side.

When the user must type text in an orientation opposite to the field orientation, such as typing numbers within a right–to–left field, use special bidirectional functions such as Push and End Push.

During a push operation, the cursor does not move (like a calculator interface). The cursor stands on the character at the boundary between the surrounding right–to–left text (for example, a street name) and the Push segment (for example, a house number).

When the user presses End Push to end the operation, the cursor skips beyond the Push segment. The user can then continue entering right–to –left text.

Text Cursor Appearance

You must make it clear when the cursor is located at a Push boundary. The Push boundary is where the latest character typed in a push operation appears, assuming that arrow keys were not used to move the cursor. Typing text at the Push boundary does not always have the same calculator –interface effect as typing elsewhere, thus you should provide some visual feedback to help the user differentiate between the two.

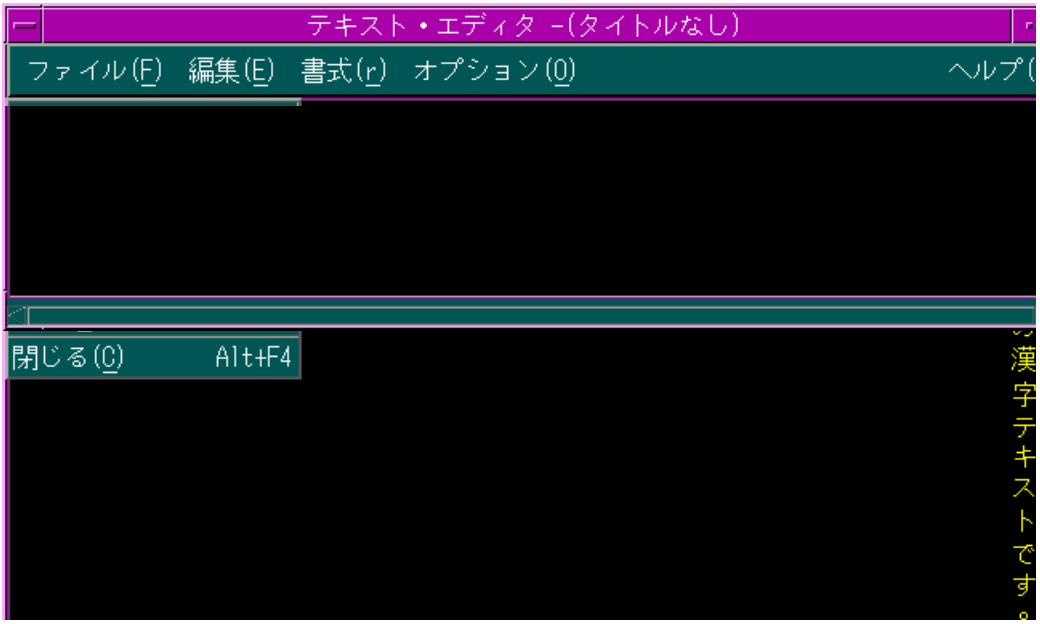
Text Cursor Position in Insert Mode

In general, the text cursor has the same appearance in insert mode as described for left–to–right text. However, in insert mode you must place the vertical bar representing the cursor on the proper side of the cursor position.

Vertical Language Support

Although Asian languages use both vertically and horizontally written text, vertical text is the more natural of the two. In vertical text, a line is a string of characters written from top to bottom, and each new line starts to the left of the previous line. Vertically Written Text (Japanese) shows an example of vertically written text.

Figure 1 Vertically Written Text (Japanese)



Character Alignment

Chinese, Japanese, and other vertically written Asian languages represent words as characters or combinations of characters. Unlike English, these characters should be equally spaced from each other. Similar to a grid, each character should be aligned on a vertical line called a center line.

Preedit Area

In vertically written text, the preedit string must be displayed vertically. You can do this with the on−the−spot method (see Locating the Preedit Area). On−the−spot character entry eases the implementation of an input method server for vertical writing.

Basic Rule for Vertical Applications

The basic rule for vertical language applications is that you must display all pieces of data in the orientation that is correct for the user. Also, data input must be supported in the orientation that is natural for users.

National Language Use

Chinese and Japanese applications generally must use the national language for titles, instructions, headings, prompts, and other window controls, with the following exceptions:

- English acronyms or terms are not commonly translated, for example, DOS, EXE, CICS.
- Key names must be identical to the keyboard; for example, `F1`, `Alt`, `Ctrl`, `Enter`.
- Key combinations must be displayed in English, for example, `Alt F2`.

Dynamic Language Selection

An application may allow dynamic language selection. When the user chooses a new language, the headings, messages, and commands should adhere to the new language. For vertical languages, the application interface should reflect these vertical language guidelines. When the application contains a mixture of vertical and left–to–right language elements, the left–to–right elements follow the unmodified guidelines provided in the Style Guide Reference; the vertical language elements follow these vertical language guidelines.

If the application is multilingual, mention the language being used in the product information window. If the application allows dynamic language selection, mention the initial language in the product information window.

Orientation

Use a vertical screen orientation for Chinese and Japanese applications. Thus the top of a window is the "begin" side, and the bottom is the "end" side. Lines of text begin with the rightmost line and continue leftward. Most references in the guidelines for left–to–right languages are true for Chinese and Japanese, with the understanding that the meaning of "top" and "right" (and "bottom" and "left") are interchanged for client areas, preedit areas, and some labels, text–display fields, and text–entry fields.

While vertical language interfaces seem to have exchanged the meaning of the words "top," "bottom," "left," and "right," the physical right and left are still the same. Thus the following have the same effect for Chinese and Japanese as for English:

- Cursor movement keys (including `[larr ]` and `[rarr ]`) must still move the cursor according to the direction of the arrow engraved on the key top. (This is also true for combinations of `[larr ]` and `[rarr ]` with `Shift`, `Ctrl`, or `Alt`.)
  Also, `Ctrl PageUp` must always scroll to the left and `Ctrl PageDown` to the right.
- Right and left buttons of a pointing device are not transposed.
- Right and left movement of a pointing device is not transposed.
- Graphics may be unaffected by the window orientation: for example, a map must still have the East on the right and the West on the left.
- Pull–down menus, pop–up menus, push buttons, and most dialog boxes still conform to left–to right language guidelines.

The appearance of a Chinese or Japanese window is the same as a corresponding English window, except that the orientation of the text in a client area is vertical instead of left–to–right.

In a vertical language environment, the user chooses the orientation of the screen and windows according to use, standards, or preferences.

Guidelines for Vertical Screens and Windows

Use the following guidelines for vertical screens and windows:

- When including English text in vertically written text, display the English text rotated clockwise by 90 degrees.
- You can use a left–to–right window on a vertical screen and vice versa. This means that English left–to–right applications can be used on a vertical screen and Chinese and Japanese (vertical) applications can be used on a left–to–right screen.
- The format of text data to be exchanged between clients is the same as in horizontally written text. Since the text buffer representation of characters is the same in both horizontal and vertical writing, you can treat the text data in the same way in both writing styles.
- Expand text selections vertically. Vertical motion of the pointer is interpreted as column expansion, and horizontal motion of the pointer is interpreted as line expansion.
- Within the same window, text and graphics may have differing orientations. For example, a geographic map may have left–to–right graphics (as for an English application), but the Chinese and Japanese captions may have vertical graphics.

- Text–entry fields are a special case: for each field, an orientation is defined, which by default is vertical within vertical windows. However, you should define fields that are to receive numeric data or English text as left to right. Within a text–entry field, the keying or cursor direction may be vertical or left to right, according to the data entered: Chinese or Japanese text, or English text and numbers.

  For more information, see Vertical Text–Entry Fields .

Vertical Text–Entry Fields

In a vertical application, each field has an orientation that is defined by the application. By default, text–entry fields assume the same orientation as the client area's text field. According to the expected contents of the field, the application must define text entry as vertical (Chinese or Japanese textual data) or left to right (numeric data or English textual data).

Vertical Text–Entry Field Differences

Vertical text–entry fields differ from left–to–right text–entry fields in the following ways:

- Text is top–aligned in the text–entry field.
- At initial entry, the cursor is located at the topmost position.
- For each entry, the cursor movement is initiated as top to bottom: in replace mode, the text cursor moves down to the next entry position; in insert mode, the cursor and all characters under the cursor are shifted one position down.
- If scrolling is available in the text–entry field, the topmost field positions are initially visible. When the cursor moves down, the information scrolls to show more characters below. In summary, the beginning of the information is on the top, its end on the bottom.

In most Asian language applications, the user must enter English letters to generate an equivalent Asian character. For example, in Chinese, the user enters a character by keying in a phonetic "word" (called PinYin) that is based on the English alphabet. Since many characters can match a phonetic approximation, it is essential to display all of the matching characters in the application's status area or preedit area. The user should be able to locate the correct character and select it quickly.

Text Cursor Position in Insert Mode

In general, the text cursor has the same appearance as described for left–to–right text except that it is rotated 90 degrees clockwise.

Designing for Accessibility

Accessibility means enabling people with disabilities to participate in substantial life activities, including the use of services, products, and information.

Removing barriers to access often results in benefits for many people −− not only those with disabilities. For example, curb−cut ramps benefit wheelchair users as well as people on bicycles and those pushing shopping carts or baby carriages.

Designing accessible software has similar benefits for a wide range of users. Solutions that use the keyboard instead of the mouse aid in keyboard −intensive tasks. Visual cues help hearing−impaired users as well as those in noisy offices or using portable computers in public places.

U.S. Government statistics show that there is a growing market for accessible computer products. Approximately 40 million Americans have a disability of some type, and as the population ages, more and more of the population will develop age−related disabilities (25% by age 55; 50% by age 65).

Not only is providing access the right thing to do, section 508 of the Federal Rehabilitation Act calls for it to be required in federal contracts. In the commercial sector, the Americans with Disabilities Act (ADA) calls for similar considerations.

Designing for accessibility also helps your customers meet their current and emerging requirements for accessible products.

Standard Accessibility Guidelines

Many users with disabilities can use applications without any adjunct assistive software or hardware; others may use additional technology such as screen readers or speech–recognition software. The guidelines in this chapter specify standardized methods of interaction that allow users with disabilities to access applications either directly or through assistive software and hardware.

Types of Disability

Like any computer user, users with disabilities vary in age, computer experience, interests, and education. When barriers are removed, the computer gives them a tool to compete with users without disabilities on an equal basis. Users with disabilities may be engineers, artists, scientists, designers, lawyers, administrative assistants, and software engineers. The common thread among the people in these diverse professions is that computers play an important role in their daily work.

The following sections discuss how to write interfaces for users who have the following disabilities:

- Physical
- Visual
- Hearing
- Language, cognitive, and other

Physical Disabilities

Physical disabilities can be the result of congenital conditions, accidents, or excessive muscular strain. Examples include spinal cord injuries, degenerative nerve diseases, stroke, and typing–based repetitive stress.

The following design principles address creating applications for those users with physical disabilities:

- Use new technology to aid in developing your application
  Adaptive hardware and software capable of optical character recognition, word prediction, eye scanning, and synthesized speech can give users with physical disabilities alternatives to keyboard and mouse interaction.
- Provide keyboard access to all application features.
  While physical capabilities vary greatly, users all need keyboard access to the controls, features, and information in applications. Providing comprehensive keyboard access is essential to ensure that the user who cannot use a mouse can productively use your application.
- Follow the key–mapping guidelines in this chapter and throughout the style guide.
  Consistent use of these mappings provides applications that are easier to use and increases the effectiveness of alternate input/output technology, such as speech control and screen readers for blind individuals.

For more information, see Selection , Keyboard Model and Key Bindings , and the Keyboard (Device) reference page.

Visual Disabilities

Effects of visual disabilities range from reduced visual acuity, requiring reading glasses, to people who need large–size displays, to the completely blind who require screen–reading software that allows them to navigate and to hear what is on the screen. As with physical disabilities, it is crucial that your application provide complete keyboard access.

The following design principles address creating applications for those users with visual disabilities:

- Do not hard code font sizes and type faces.
  Allow users to configure all of the fonts in your application, including text in text–entry fields, menus, labels, and messages.
- Do not hard code your application colors.
  In addition to being difficult to interpret, some background and text color combinations can result in text that is difficult to read. Users should always have the capability to override default colors so that they can choose the colors that work best for them.
- Never use color as the only source of information.
  Interpreting information that depends upon color (for example, red=stop, green=go) is particularly difficult for those who are color blind or unable to see differences among some colors.
- Do not hard code graphic attributes such as line, border, and shadow thickness.
  Allow users to change the attributes of window controls for sizing and separation. A window border, shadow, or separator that is too thin or thick can be difficult to identify or may distort the user's focus.
- Provide meaningful names for controls.
  If you use nonstandard (for example, custom) controls, or standard controls without text (for example, push buttons or a palette with graphics), then you must give each control a meaningful name. Rather than calling an eraser graphic *''control5,''* for example, call it *''eraser.''* Users who rely on screen reading and character recognition devices will not understand the control's purpose unless it has a meaningful name.

For more information, see Visual Cues and the Control, Label, and Text–Entry Field (Control) reference pages.

Hearing Disabilities

People with hearing disabilities cannot hear sound output at typical volumes, if at all. In some cases, users can hear audible cues only at certain

frequencies or volumes.

The following design principles address creating applications for those with hearing disabilities:

- Do not overuse or rely exclusively on audible cues.
  Never assume that a user can hear an auditory notice. Even users who are not hearing impaired are often unable to use sound because they are in a noisy office or using a portable computer in an area where sound is prohibited.

  Sounds unaccompanied by visual notification, such as a beep indicating that a print job has completed, are of no value to hearing−impaired users or others who are not using sound. While such sounds can be valuable, never create a design that works on the assumption that they will be heard.
- Where appropriate, allow users to choose between audible or visual cues.
  There are times when either an audible cue or a visual cue is more appropriate. For example, hearing−impaired users and anybody using a system in a public area would benefit from the option of choosing to see rather than to hear a notice. Such notices can take the form of a changing icon or a message in an information area window.

  On the other hand, in the printer example it would be intrusive for most users to see a warning window every time a printout is ready. Therefore an audible cue is more appropriate.
- Allow users to configure the frequency and volume of audible cues.
  Every user has different hearing abilities. Allowing the user to change the frequency and volume of an audible cue helps to ensure that the user will both hear the cue and distinguish it from other audible cues.

For more information, see Audible Cues .

Language, Cognitive, and Other Disabilities

The wide range of disabilities is too broad to cover extensively. However, users with language, cognitive, and other disabilities can benefit from an application that:

- Allows flexible limits for task completion
- Minimizes the use of transitory signals
- Reduces the user's memory requirements

The guidelines outlined for physical, visual, and hearing disabilities also typically benefit users with cognitive, language, and other disabilities.

Existing Keyboard Access Features

When designing Motif applications, be aware of existing system–level keymappings that access features used in the X Windows server. These server features, known as AccessX, provide basic workstation accessibility, typically used by people with mobility impairments. AccessX became a supported part of the X Windows server in Version X11/R6.

The built–in server–level access features include:

StickyKeys

Provides locking or latching of modifier keys (for example, `Shift`, `Control`) so that they can be used without simultaneously pressing the keys being modified. This allows single–finger operation of multiple key combinations.

RepeatKeys

Delays the onset of key repeat, allowing a user with limited coordination time to release keys before multiple characters are sent.

SlowKeys

Requires a key to be held down for a set period before keypress acceptance. This allows a user with limited coordination to accidentally press keys without sending too many keypress events.

MouseKeys

An alternative to using the mouse, provides keyboard–based explicit control of cursor movement and all mouse button press/release events.

ToggleKeys

Indicates a locking key state with a tone when pressed, for example, `Caps Lock`.

BounceKeys

Requires a delay between keystrokes before accepting the next keypress so a user with tremors can accidentally press the same key without sending too many keypress events.

To avoid conflicts between Motif applications and current and future access tools, use **AccessX Key Mappings**, which lists the key mappings that have been reserved for AccessX.

Table 1 AccessX Key Mappings

Keyboard Mapping	Reserved For
5 consecutive clicks of `shift`	On/Off for StickyKeys
`Shift` held down 8 seconds	On/Off for SlowKeys and RepeatKeys
6 consecutive clicks of `Ctrl`	On/Off for screen–reader numeric keypad functions.
6 consecutive clicks of `Alt`	Reserved for future AccessX use.

Resources for More Information on Accessibility

For more information about software accessibility, consult the following organizations, conferences, and books:

Organizations

```
Clearinghouse on Computer Accommodation (COCA)
18th & F Streets, NW
Room 1213
Washington, DC  20405
(202) 501-4906
```

A central clearinghouse of information on technology and accessibility. COCA documentation covers products, government resources, user requirements, legal requirements, and much more.

```
Sensory Access Foundation
385 Sherman Avenue, Suite 2
Palo Alto, CA  94306
(415) 329-0430
```

A nonprofit organization that consults on application of technology "to increase options for visually and hearing impaired persons." Publishes newsletters on adaptive technology.

```
Special Needs Project
3463 State Street
Santa Barbara, CA  93105
(805) 683-9633
```

Vendor of books on a wide variety of disability issues for professionals and families.

```
Trace Research and Development Center
S-151 Waisman Center
1500 Highland Avenue
Madison, WI  53528
(608) 262-6966
```

A central source for the current information on assistive technologies as well as a major research and evaluation center. Trace distributes databases and papers on adaptive technology and resources.

Conferences

```
CSUN
Conference on Technology and Persons with Disabilities
Every spring in Los Angeles, California
(818) 885-2578
```

```
Closing the Gap
Conference on Microcomputer Technology in Special Education and Rehabilitation
Every fall in Minneapolis, Minnesota
(612) 248-3294
```

Common Desktop Environment Guidelines

The Common Desktop Environment (CDE) is a graphical user interface for UNIX(TM) in its variants (AIX, Digital UNIX, HP/UX, Solaris, UnixWare, and so on). CDE brings unparalleled ease of use to UNIX and is being adopted as a standard operating environment by many companies in the UNIX workstation market.

Advantages of a Common User Interface

The CDE interface provides many advantages to both end users and application developers. Among these advantages are:

- An easy−to−use interface, which enables the user to learn the system quickly and to use it efficiently.
- Consistency between UNIX platforms, which enables the user to move from one computer to another easily. It also enables programmers to write a single application that can be compiled for each platform, significantly reducing development effort.
- Similar consistency with the Microsoft Windows and IBM OS/2 environments, which enables the user to move easily between these environments and CDE.
- Several built−in productivity applications, which enable the desktop user to be productive before buying application software.
- Desktop specifications that have been submitted to the X/Open standards organization, ensuring that CDE is *''open''* and will not tie the user to proprietary solutions.

Relationship of CDE to Motif

The CDE user interface follows the Motif guidelines presented in this book. Motif, however, does not define a desktop, only the basic behaviors for applications and widgets. This chapter defines the guidelines that allow an application to integrate well with the desktop. Thus, to write a desktop–conforming application, you should follow the guidelines in this chapter as well as the previous chapters.

The guidelines in the following sections are specific to CDE and may not necessarily apply to a Motif interface or a Motif application running under another window manager (for example mwm or twm). The sections include information on:

- Visual design
- Application design guidelines
- Window and session control
- Application messages
- Drag and drop

CDE Visual Design

CDE is a visually rich environment. This section provides information on designing icons and other visuals consistent with the desktop style. It discusses some of the design philosophy behind the desktop and contains useful hints to help you successfully create icons for the desktop environment.

In most graphical user interfaces (GUIs), color is applied in a localized and specific manner, either in individual icons or specific control areas, such as window borders or title bars. In many other GUIs, color is pervasive, as virtually everything is drawn with colors, with a notable absence of black lines.

Most CDE icons are not color intensive, using grays instead. This keeps the number of colors on the desktop palette to a minimum and works well visually. Because the icons, being largely colorless, always appear in the context of colored backgrounds, they stand out more.

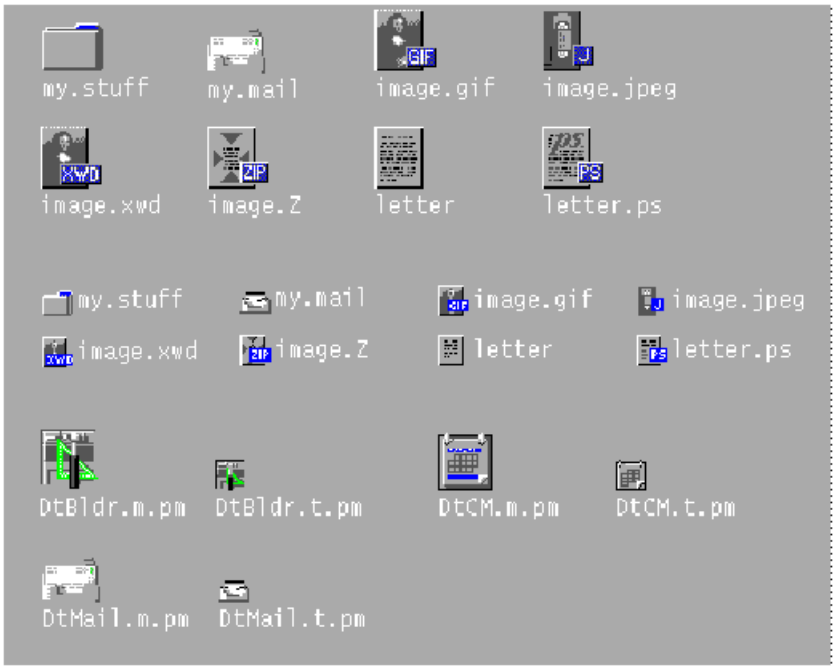
Using Icons in CDE

An icon in CDE is a specific graphical element that the user can move, copy, delete, or open. The following CDE applications use icons as a fundamental method for user interaction. (For information on the available applications that CDE includes, see the CDE User's Guide).

File Manager Icons

File Manager is the tool that provides for the presentation and organization of the user's file structure. The basic types of iconic objects displayed in File Manager are files, directories (folders), executables, and actions. These objects are referred to as documents, folders, and applications. File Manager displays the icons in two sizes, called Icon and Small Icon views in the Set View Properties dialog box. Icon is 32x32 pixels and Small Icon is 16x16 pixels. File Manager Icons at Sizes 32x32 and 16x16 shows both icon sizes.

Figure 1 File Manager Icons at Sizes 32x32 and 16x16



Documents, folders, and applications are represented by three different shapes. Documents are vertical rectangles meant to look like pieces of paper. Folders are horizontal rectangles with a tab to look like a file folder. Applications can be any shape and use the entire icon square. All objects in File Manager should indicate to the user that they can be manipulated, that is, dragged and dropped.

Application Manager Group Icons

The Application Manager is similar to File Manager, but its focus is on holding applications rather than documents. All network–accessible applications in the desktop are placed here in containers, called *application groups* (rather than folders).

Application Manager is like a "network store." This is the place where the user goes to find the latest applications available on the system. Application Manager Group Icons shows an example of the Application Manager's group icons.

Figure 2 Application Manager Group Icons

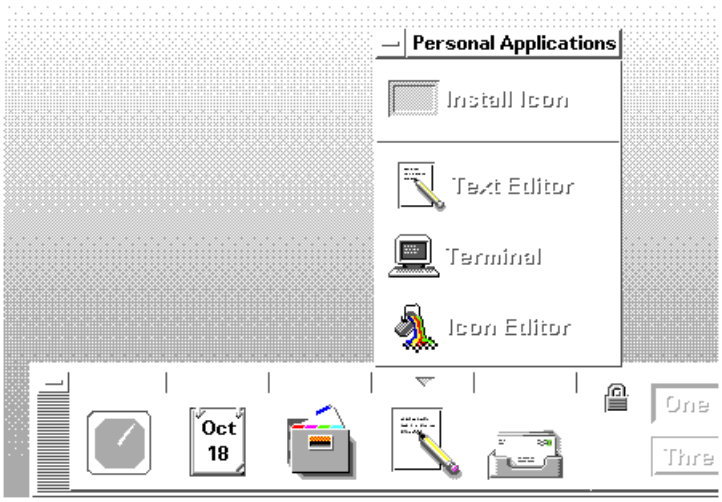
Application group icons are like folders in that they represent a collection of objects, in this case related objects. If your application requires support files or includes sample files, you can design your own application group icon that represents where a user can get the related files for your application.

Front Panel and Subpanel Icons

The Front Panel is the control panel for the desktop and usually appears at the bottom of the screen. Front Panel icons provide quick access to the user's most commonly used applications.

The Front Panel also has subpanels of icons that the user can access with the arrow buttons on the Front Panel. The subpanel is an extension of the Front Panel icon. For example, Front Panel and Subpanel Icons shows the Personal Applications subpanel open. The user can add applications to this subpanel by dropping them on the Install drop site. The user can choose to promote icons in the subpanel to the Front Panel via the pop–up menu.

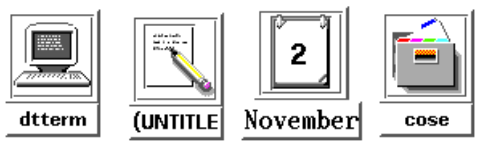
Figure 3 Front Panel and Subpanel Icons



Minimized Window Icons

Minimized window icons appear on the desktop when a window is minimized. The icon should represent the application that controls the minimized window, as shown in Minimized Window Icons . These icons are different from the icons used in the Front Panel in that they represent running applications, although they are the same size.

Figure 4 Minimized Window Icons



Other Graphic Elements

Other graphic elements include button graphics, tool bar graphics, and graphics used as labels. A tool palette in a paint program is an example; a document orientation button (landscape or portrait) in a printer dialog box is another. These are graphics that you create for use in your application and that are not used elsewhere. Other Graphics is an example of some other graphic elements (tool bar graphics).

Figure 5 Other Graphics

Using Color in CDE Icons

When designing icons for a CDE application, you must be aware of the available color palette and the dynamic mapping of colors.

Icon Color Palette

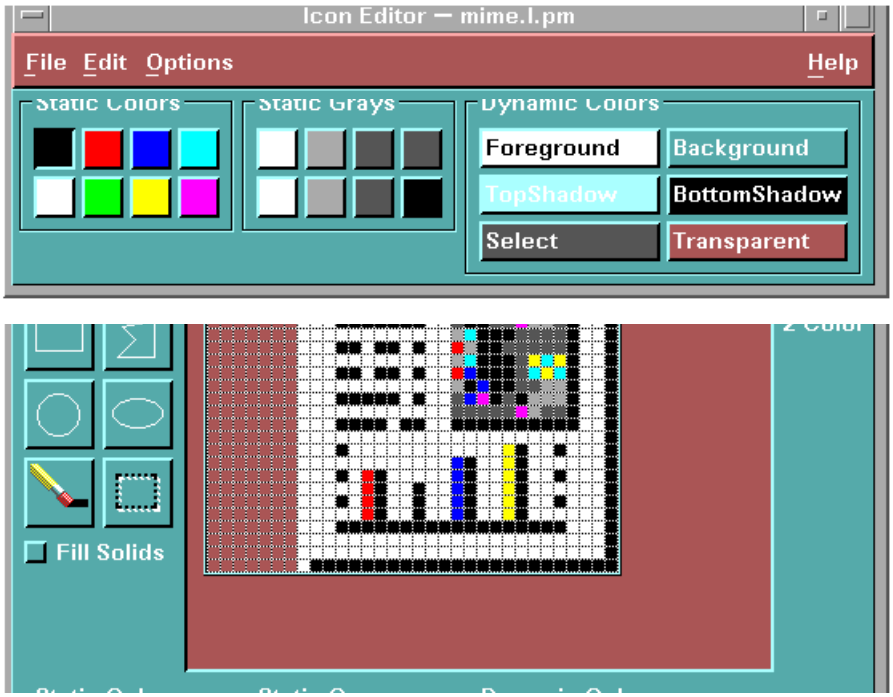
CDE icons use a palette of 22 colors:

- Eight static grays
- Eight static colors (white, black, red, blue, green, cyan, magenta, and yellow)
- Five dynamic colors for the Foreground, Background, TopShadow, BottomShadow, and Select areas
- A transparent "color" that allows the background to show through

These colors are the default colors in the Icon Editor, which is the recommended tool for creating desktop icons. Icon Editor Color Palette shows the Icon Editor and its color palette. This limited palette was chosen to maximize the attractiveness and readability of icons without using an unnecessary number of colors.

If you use more than the 22 icon colors, then your icons may experience color−flashing effects that can make the icon unreadable. The best way to ensure predictability of appearance of your icons is to use only the 22 colors in the desktop palette.

Figure 6 Icon Editor Color Palette



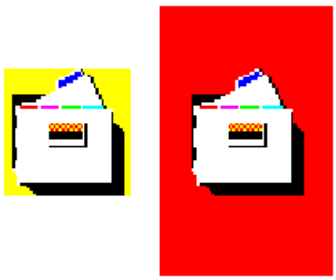
Role of Dynamic Colors

It is important to understand the limited role of the dynamic colors. These represent the colors used to display the user interface elements on which your icon appears. If your icon appears in File Manager, File Manager determines what the background color is. If the user changes the color palette in Style Manager, the colors in the user interface change to match, and the background color on which the icons are displayed changes.

In general, dynamic colors have little use in most icons. There are two ways they are used:

- If your icon does not fill the entire bounding box, then fill the unused area with the transparent color.
- You can draw a shadow under your icon. This is recommended only for Front Panel icons. Do not use this for File Manager icons. Dynamic Color Shadows shows dynamic color shadows.

Figure 7 Dynamic Color Shadows



Understanding CDE Design Philosophy and Helpful Hints

The philosophy behind the graphic language of the desktop is that users benefit if the computer world parallels the real world. This extends from the three–dimensional appearance of windows and controls, such as push buttons and menu bars, to the general appearance of icons.

Designing with Color

Design your icon primarily to use the eight static grays; use the eight static colors mostly as accents. The eight static colors are strong and can easily be overused. The static grays allow icons to blend gracefully with the already colorful desktop environment. You can dither the static colors with the static grays to tone the colors down for coverage of larger areas. You can also use the grays to smooth the edges of icons. This is sometimes referred to as "anti–aliasing."

Do not use dynamic colors in File Manager icons because the appearance of the icon will change when the user changes color palettes. Such a change could be inappropriate as well as unpredictable.

Using Different Icon Styles

Icons have various graphical styles. The most basic icon consists of a simple black outline. However, as color is added, the style resembles that of a coloring book, with color added within the black lines. This is most effective when done on white backgrounds.

CDE, with its pervasive use of colored and medium–value backgrounds, uses both lighter and darker shades to create fairly realistic images. You are encouraged to explore this style.

Another element of style is the point of view taken in portraying the object. CDE uses a head–on view, as shown in Examples of Three–Dimensional Icons in CDE, usually from slightly above if the object in question is a three–dimensional one, such as a printer. You should use a treatment that gives the icon a slight dimensional quality, as this reinforces the perception that the icon can be dragged and dropped.

Figure 8 Examples of Three–Dimensional Icons in CDE

Designing An Application Icon

The application icon is the most important icon you will design. This is the place for your product identity, as well as a clear indication to the user of what your application does. The application icon is what the user opens to run your application.

There are no shape conventions for application icons. They can fill the entire icon bounding box or they can be irregular in shape. It is recommended that your icon appear three–dimensional. Application Icons That CDE Uses shows some CDE icons. You can use these icons as templates when designing your own icons.

Figure 9 Application Icons That CDE Uses

Designing An Application Group Icon

The application group icon represents the container in which the user finds your application, as well as any other files you may choose to include, such as information or sample files. Design the icon in such a way that the user knows it is a container, such as a folder or box.

The concept that Application Manager uses is that of an icon based on accordion−style folders. This icon is large enough that you can stamp images on the front of it to indicate to the user what kinds of items are inside. Application Group Icons is an example of application group icons.

Figure 10 Application Group Icons

Designing Document Icons

A document icon should help the user understand what kind of data is stored in that document icon and what application is associated with the document. Document Icons shows a number of CDE document icons that you can use as templates when designing your own document icons.

Applications that support multiple file formats need different document icons for the different output formats. Rather than creating a distinctively different graphic for each format, you might use the same graphic for the basic file and add a tag to, or a small label in, the icon to delineate the format.

In the case of the document icon, the basic rectangle of the document is left−aligned in the icon square. If you use the tag approach, place the tag on the right side of the icon, half on and half off the basic icon, but not obliterating the descriptive graphic, as shown in Document Icons .

Figure 11 Document Icons

Understanding the Differences Among Platforms

The desktop is different from the application spaces you may be familiar with in the following ways:

- The desktop requires a 48x48 pixel size to accommodate higher resolution displays.
- The desktop has a different color space for icons. You may be able to reuse icons from other environments but if they have color in them, chances are some of the colors will need to be changed to map onto the desktop palette. The basic design should still work. See RGB Values for Common Desktop Environment Icon Colors for information on translating colors.
- Perhaps the most significant difference is that, in most cases, desktop icons appear against a background color other than white. This can make your icon appear unreadable if you simply copy it from another environment. You should test any icons from other environments before using them on the desktop.

Table 1 RGB Values for Common Desktop Environment Icon Colors

Color	RGB Values Decimal	RGB Values Hex	Grays	RGB Values Decimal	RGB Values Hex
Black	0,0,0	#00	Gray1	222,222,222	#de
White	255,255,255	#ff	Gray2	189,189,189	#bd
Red	255,0,0	#ff0000	Gray3	173,173,173	#ad
Green	0,255,0	#00ff00	Gray4	148,148,148	#94
Blue	0,0,255	#0000ff	Gray5	115,115,115	#73
Yellow	255,255,0	#ffff00	Gray6	99,99,99	#63
Cyan	0,255,255	#00ffff	Gray7	66,66,66	#42
Magenta	255,0,255	#ff00ff	Gray8	33,33,33	#33

Implementing Required Icons

This section discusses the details you need to know to create icons that display correctly in CDE, such as formats, resolutions, sizes, naming, and so on.

Icon Formats

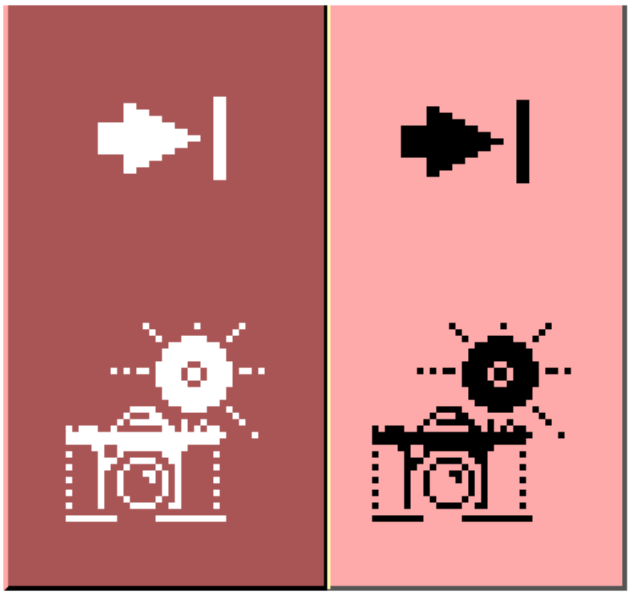
CDE runs in both color and monochrome modes, so you must create your icons in two formats: XBM (X bitmap) for monochrome and XPM (X pixmap) for color. The Icon Editor saves icon files to both formats.

On the desktop, buttons and palettes can use either the XBM or XPM formats. You should use the XPM format wherever possible for your button, palette, and tool bar graphics.

The XBM file format has only two colors: foreground and background. On the desktop, the foreground color is not fixed, but varies according to the background color. A color scheme with a dark gray background might cause any text or graphics to appear white. However, a color scheme with a light gray background will cause text and graphics to appear black.

Inverting the foreground color may have a strange effect on certain icons. For something simple, like an arrow shape, there is no adverse consequence. But for other images, the negative version created by inverting the foreground color might be illegible and, therefore, unusable. Monochrome (XBM) Bitmaps with Foreground Reversal Consequences shows the consequences of reversing the foreground color in monochrome mode.

Figure 12 Monochrome (XBM) Bitmaps with Foreground Reversal Consequences



Display Resolution

CDE accommodates three display resolutions: low resolution (640x480 pixels), medium resolution (800x600 pixels), and high resolution (mega −pixel). The size of the Front Panel and some of the icons change automatically, depending on the display resolution. For this reason, your application must provide different icon sizes.

Icon Sizes

CDE has three icon sizes: 16x16, 32x32, and 48x48, referred to as 16, 32, and 48. (They have suffixes of .t, .m, and .l, respectively.) If your application comes from the PC domain, then the size 16 and 32 icons are familiar sizes. Icon Sizes and Usage defines where each size is used.

Table 2 Icon Sizes and Usage

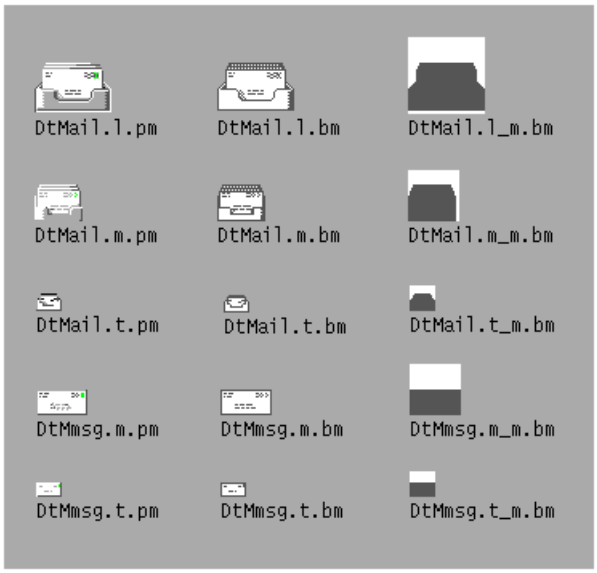
Component	Low Resolution	Medium Resolution	High Resolution
File Manager	32,16	32,16	32,16
Application Manager	32,16	32,16	32,16
Front Panel	32	48	48
Subpanels	16	32	32
Front Panel Controls	16	16	16
Minimized Window	32	48	48

Minimum Required Icon Set lists the icons you need to create for an application. A total of 16 icon files are needed, assuming one of each type and size. Minimum Required Set of Icons for Mailer shows an example set of icons.

Table 3 Minimum Required Icon Set

Type of Icon	16 Color	32 Color	48 Color	Mono. 16	Mono. 32	Mono. 48
Application icon	X	X	X	X	X	X
Document or file icon	X	X		X	X	
Application container icon	X	X		X	X	
Minimized windows			X			X

Figure 13 Minimum Required Set of Icons for Mailer



Icon Naming Conventions

The basic name for the icon must be no more than seven characters. The size and color suffixes are appended to the name, as shown in Icon Naming Conventions .

Table 4 Icon Naming Conventions

Size	COLOR	B&W	B&W Mask
48	*Iconame.l.pm*	*Iconame.l.bm*	*Iconame.l_m.bm*
32	*Iconame.m.pm*	*Iconame.m.bm*	*Iconame.m_m.bm*
24	*Iconame.s.pm*	*Iconame.s.bm*	*Iconame.s_m.bm*
16	*Iconame.t.pm*	*Iconame.t.bm*	*Iconame.t_m.bm*

The suffix *.pm* is for the XPM format. The suffix *.bm* is for the XBM format. The suffix *_m* refers to the mask for the black–and–white icon.

You do not have to provide icons in all these configurations. Minimum Required Icon Set lists the required icons. For example, the *.s* icons are typically used for tool bars, which your application may not have.

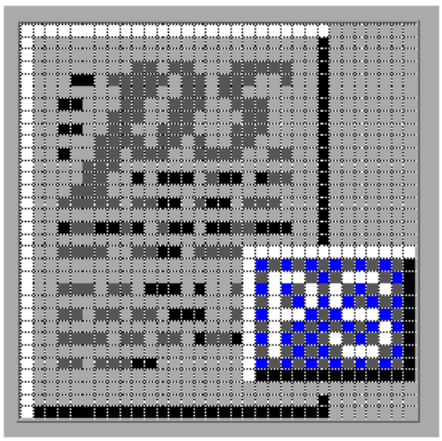
Icon Alignment

Depending on the graphic you use for your icon, the bits may not take up the entire space allocated for the icon. The recommended rules for where the empty space goes in a desktop icon are:

- For 16x16 and 32x32 icons, which are left–aligned, any empty bits are on the right side of the bounding box.
- For 48x48 icons, which are centered in the bounding box, any empty bits surround the centered image.

Left–Aligned 32x32 Icon with Tag is an example of a left–aligned 32x32 icon with a tag.

Figure 14 Left–Aligned 32x32 Icon with Tag



Optional Icon Sizes

There is no size 48 requirement for the document or application group icons because neither are expected to be used for a minimized window icon (you use the tool's icon instead) or in the Front Panel. A user might, however, promote one of these icons to the Front Panel.

Optional Front Panel Icon Style

The icons that appear by default in the Front Panel have a slightly different appearance from File Manager icons. These icons cannot be dragged and dropped and are thus given a more permanent look by appearing to be etched into the surface.

You do not have to provide size 48 icons with an etched appearance. Since you cannot determine if and when your icons will be used in the Front Panel, you should not design specifically for this usage; instead, design for File Manager usage, which is more common.

CDE Application Design Guidelines

Your application should present its components to the user in a logical and task–oriented manner. Menus should follow a common organization and naming convention to enable users to use the same rules and practices across the desktop. The following sections outline CDE application design and menu structure requirements.

General Guidelines

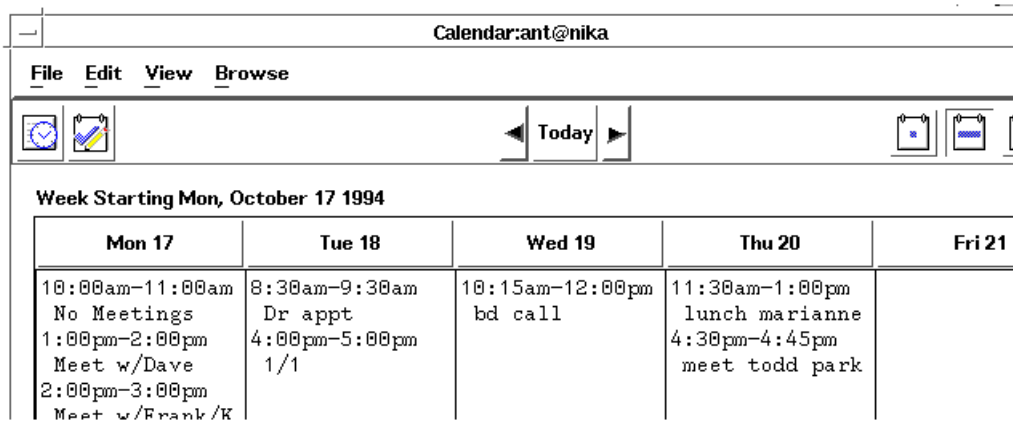
Consider the following guidelines when designing a CDE application:

- There should always be exactly one control within any window of your application that has the input focus if the window in which it resides has the input focus.
  If any window within your application has focus, some control within that window must have focus. The user should not have to explicitly set focus to a control within the window.
- When a text field within your application does not have input focus, do not display the text cursor within that field.
  Although use of inactive text cursors is allowed within Motif, it is better to hide the text cursor when removing focus rather than display the inactive text cursor. This makes it easier for the user to quickly scan the screen or window to determine which text field currently has focus.
- Your application should provide keyboard mnemonics for all buttons, menus, and menu items displayed within the application.
  Once the user becomes adept at using your application, keyboard mnemonics are a quick way to access functionality. Mnemonics also facilitate access to functionality from within keyboard–centric applications or windows. The user need not frequently switch between using the mouse and keyboard. Mnemonics should be provided pervasively throughout the user interface.
- Your application should provide shortcut keys (accelerators) for those functions that you expect the user to use frequently.
  Shortcut keys provide the user who has become expert at using your application a quick way to access application functionality without going through menus and dialog boxes.
- If your application does not use the values of global environment settings, such as multiclick timeout intervals, drag thresholds, window color settings, mouse left– or right–handedness, and so on, but instead uses its own values for these settings, then your application should provide one or more Options dialog boxes that allow the user to change the values for these settings.
  In general, you should not override the value of settings treated as global environment settings. The user controls these settings through the CDE Style Manager. If you choose to ignore these settings and specify your own settings, then your application will be inconsistent with other applications in CDE. If you nevertheless choose to provide your own values, then you must provide the user with a way to make your settings consistent with the rest of the desktop.

Tool Bars

Tool bars provide quick access to already user–accessible functions. Some common usages of tool bars include navigation, changing data views, accessing frequently used tools or editors, simplifying the number of steps to complete a common operation, and providing a fast path to frequently used menu items. Tool Bar on the CDE Calendar is an example of a tool bar on the CDE Calendar.

Figure 1 Tool Bar on the CDE Calendar



Tool Bar Design Issues

When designing your application and an associated tool bar, consider the following guidelines:

- Use tool bars only when they improve or enhance user access to common operations, such as in an application with several large menus.
- Present a natural organization of actions on the tool bar. Grouping items that are dissimilar can confuse the user if the item they are looking for is not in the proper context.
- Do not place too many items in the tool bar. The user should be able to find and use an item quickly. Keep the number of buttons to a minimum so that you do not increase the difficulty of using a tool bar.
- Do not use cryptic icons as they can confuse the user. Keep the pixmaps as simple as possible. Remember that all graphics must be international in scope. When designing a graphic to represent a command, such as Save, remember that the icon has to represent a verb, as opposed to a noun, like most other icons.

Tool Bar Components

You typically use the following Motif components when constructing a tool bar:

Tool bar container

The tool bar uses a container component to provide a layout mechanism for the drawn buttons that make up a tool bar. You may choose most any container for the tool bar, as long as it allows for the specified behavior. The tool bar container is placed directly under the menu bar and should be the same width as the window, as well as similar height to the menu bar.

Tool bar button

The Motif widget *DrawnButton* provides an appropriate medium for the graphic buttons in tool bars.

Pixmap

The pixmap for the drawn button is the graphic that conveys the functionality to be expected by pushing a particular button.

CDE Window and Session Control

Your application is presented to the user as a series of windows. Some of these windows present the main portion of the application. Others are dynamic, appearing to the user only when needed to accomplish certain tasks. All of these windows should contain menus, border decorations, and behavior styles appropriate to their function. The following sections describe the guidelines for designing your application's windows.

Window Control Guidelines

The fundamental user−visible characteristic of primary windows is that stacking, workspace placement, and minimization can be independent of other primary windows. Secondary window stacking, workspace placement, and minimization must be tied to the associated primary window.

Window Management Choices

You should provide the Close, Move, Lower, and Minimize choices as the minimum set of capabilities in primary windows. Provide Resize and Maximize choices as appropriate. Design your secondary windows so that resizing and maximizing are neither necessary nor appropriate. Most secondary windows should include only the Close, Move, and Lower choices. In extraordinary cases, you may provide the Resize and Maximize choices in a secondary window. Do not provide the Minimize choice in secondary windows (they are minimized with the associated primary window).

Windows that have a Close or Exit choice need to support the window management protocol for Close if there is a window menu. In the case of dialog boxes, the Close item on the window menu corresponds to the Cancel choice or dialog box dismissal with no further action taken.

Workspace Management Guidelines

CDE applications appear in one of several work areas called workspaces. A user may have several workspaces active on the desktop. Your application should behave in certain ways in relation to those workspaces.

For example, a spreadsheet application may have one or more secondary windows open that allow the user to change the properties of data cells in the main window. If the user moves the main window to a different workspace, the secondary properties windows should move with it.

On the other hand, a word processor may have several windows open, where each is used to edit a different document. In this case, when a user moves one of the windows to a different workspace, the other windows should remain where they are.

Session Management Support

When you design a desktop application, consider the following guidelines for session management:

- Applications should support Interclient Communications Conventions Manual (ICCCM) mechanisms for session management of their primary windows and key properties.
  The ICCCM defines important relationships and behaviors between applications and the window manager, including protocols for saving and restoring application states across invocations.
- Applications should support ICCCM mechanisms for session management of all associated windows (that is, secondary windows that may include help windows).
  Associated windows include multiple primary windows and secondary windows, such as online help windows.
- Applications should accept messages from the CDE Session Manager that inform them the user is logging out and should save their state at that time.

CDE Application Messages

From time to time, an application needs to present feedback to keep the user informed about the progress of ongoing activities and to alert the user to situations that require intervention. The CDE Motif interface provides many ways to provide such feedback to the user. This section describes the use of error messages, informational messages, and other message dialog boxes.

Error Messages

Use error messages when it is crucial to bring the information to the user's attention because the intended action cannot be carried out without user intervention. Use a Motif error dialog box to present application error messages. Keep in mind the basic three–part structure for error messages. Each error message should tell the user:

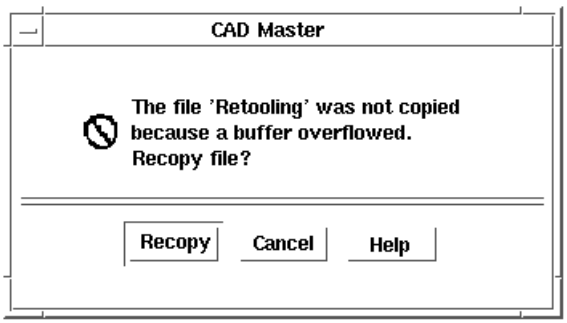
- What happened
- Why it happened
- What should be done to correct the problem

Error Dialog Box is an example of an error dialog box.

It is appropriate to assume that the user knows basic desktop terms, such as *files* or *programs*. However, avoid terminology that is typically understood only by an expert or frequent computer user unless the application is specifically targeted at computer professionals.

In many cases, the only user response to an error dialog box is to click the OK button to dismiss the dialog box. However, you should be able to offer resolutions to the problem. If you have buttons for user actions, be sure to also include a Cancel button.

Figure 1 Error Dialog Box

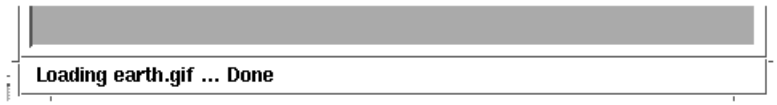


Informational Messages

Use informational messages in the window footer to present progress, status, or helpful information to the user. Do not use informational messages to present crucial information because informational messages are deliberately designed to be nonobtrusive and many users may not notice them.

Motif provides a message area at the bottom of the main window, but this is rather clumsy and ugly. A more elegant approach is to provide a wider margin below the data area of the main window where status information can be unobtrusively displayed, as shown in Informational Message in the Lower Margin of a Window . For other examples of using informational messages, see the status message area in the CDE Mailer.

Figure 2 Informational Message in the Lower Margin of a Window



The text "Loading earth.gif..." is displayed at the start of the load and the text "Done" is added when the load has completed. The entire message is removed 5 seconds later.

Informational messages in the footer area should be left–justified and displayed in a light font in keeping with their unobtrusive nature. Note that the margin where informational messages are displayed should not accept mouse focus. Progress messages in the footer area are normally displayed only while the operation is in progress. Remove notices and other information that is no longer valid within a few seconds to avoid confusion about whether the information is current.

Other Message Dialogs

CDE supports the following dialog boxes:

Information dialog box

> Used to display status, completion of activity, or other informative types of messages to which the user need not necessarily respond other than to acknowledge having read the message.

Question dialog box

> Used to ask questions of the user. The question should be clearly worded to indicate what a yes or no response means. The buttons displayed are Yes, No, and Help. Help provides additional information as to what the application will do in response to a Yes or No choice. When possible, you should replace the label for the Yes and No buttons to make it clear what action will be performed as a result of choosing either operation. For example, when a file is not yet saved but the user chooses to close it, Save, Discard, Cancel, and Help would be more appropriate choices than Yes or No in a question dialog box.

Warning dialog box

> Used to communicate the consequences of an action requested by the user that may result in a loss of data or other undesirable event. The dialog box is presented before the action is performed and offers the user the opportunity to cancel the requested operation. The buttons typically displayed are Yes, No, and Help –– or Continue, Cancel, and Help.

Working dialog box

> Used to display in–progress information to the user when this information is not displayed in the footer of your application’s window. The dialog box contains a Stop button that allows the user to terminate the activity. When pressed, the operation is terminated at the next appropriate breakpoint, and a confirmation might be displayed asking whether the user really wants to stop the activity.

Drag and Drop

This section discusses the drag–and–drop operation and provides CDE guidelines for incorporating drag and drop into your CDE application.

You should be familiar with the description of Motif drag and drop. For more information, see Data Transfer . Where differences occur, this section supersedes Data Transfer when developing applications for CDE.

Mouse–Based Selection

CDE has incorporated two changes to mouse–based selection that is significantly different than in Motif. The first is that the user may elect to have either adjust or transfer capability on the middle mouse button. In addition, CDE integrates drag and select on the first mouse button.

On a 3–button mouse, MB2 is typically used as the TRANSFER (or SELECT) button. However, in CDE the user may change an environment setting to indicate that MB2 should be used as the ADJUST button. The ADJUST button can be used to toggle the selection state of elements under the multiple selection model.

Drag–And–Drop User Model

In CDE the user can select and drag icons in File Manager, mail messages and attachments in Mailer, appointments in Calendar, and text from text editors or text fields. The user can drop items onto any drop zone that accepts them. For example, a document icon from File Manager can be dropped onto a folder icon in File Manager, onto the Print Manager icon in the Front Panel, or onto the attachment list in Mailer.

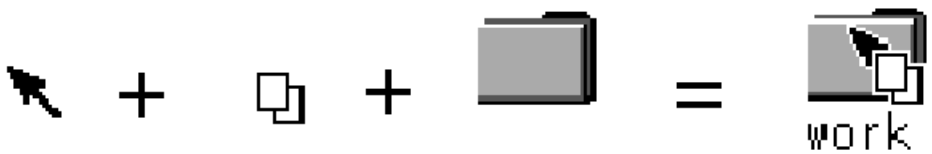
Parts of a Drag Icon

The drag icon consists of three parts:

- State indicator
- Operation indicator
- Source indicator

Parts of a Drag Icon shows these parts.

Figure 1 Parts of a Drag Icon



State Indicator

The *state indicator* is a positioning pointer combined with a valid or invalid drop zone indicator. The valid state indicator should resemble an arrow pointer with a hot spot so the user can position the cursor in a predictable manner. The invalid state indicator, a cannot pointer, appears when the user is over an invalid drop zone. Invalid State Indicator shows an example of an invalid state indicator.

Figure 2 Invalid State Indicator

Operation Indicator

The *operation indicator* gives the user feedback on what operation is occurring during the drag. Operation Indicator shows the copy and link feedback. Because most drags are move operations, an operation indicator is added to the drag icon only for copy or link operations.

Figure 3 Operation Indicator

Source Indicator

The *source indicator* is a representation of the selection (or the item being dragged). It comes in several versions, depending upon whether the selection represents single or multiple items and what kind of item the selection represents. Source Indicator shows examples of source indicator drag icons.

Figure 4 Source Indicator

Valid Move

Invalid Move

Invalid Copy

Invalid Link

The hot spot for text drag icons is located on the top left corner (1,1). The hot spot for single and multiple drag icons is located at the top left pixel of the invalid icon (3,3). CDE has tuned each drag icon to increase user accuracy at targeting and positioning.

Drag Icon Graphic

You are responsible for supplying a graphic to be used in the drag icon in your application. This graphic is usually one of the icons already supplied with the application, such as the 32x32 icons that File Manager uses to represent documents. The graphic used depends on what is being dragged.

In cases where the user does not select an icon to start a drag, it is still appropriate to show a relevant graphic in the drag icon, especially if that graphic will be used by the destination application upon the drop. For example, in Calendar Appointment Editor, the user can select an appointment from the scrolling list, which does not show icons. An appointment icon is used as part of the drag icon. If the appointment were dropped on File Manager, File Manager would display the appointment, using the same graphic.

In CDE three operations are associated with the drag icon. These operations are explained in Parts of a Drag Icon . The drag icon has alternate graphics that indicate to the user when the operation is a copy or link. The default operation, move, requires no alternate icon graphics.

Success or Failure of a Drag–and–Drop Operation

The user needs an indication that the drag–and–drop operation either succeeded or failed. You should use transition effects to indicate the success or failure of the drop. There are two kinds of transition effects: melt and snap back.

Use the *melt* effect when the user drops a drag icon on a valid drop zone. The effect looks like the drag icon melts into the drop zone. The drag icon disappears and is replaced by whatever is appropriate for the destination application. Dropping a drag icon on the Print Manager control in the Front Panel may show nothing other than the melt effect. Dropping a drag icon on the File Manager control would show the melt effect followed by the icon appearing in File Manager.

Use the *snap back* effect when the drop fails. Drops can fail for two reasons: because the drop zone is invalid or because the data transfer fails. If the user drops a drag icon over an invalid drop zone, one that shows the cannot pointer drag icon, then the drag icon snaps back to the source application.

Once a drop occurs, the source and destination applications have to transfer the data. If the data transfer fails, the destination application should do the following:

- Indicate to the API that the drop failed so that the dropped item will get snapped back to the source application
- Display an error notice that clearly indicates why the drop failed and what, if anything, the user can do to correct the situation

Sometimes the transition effect does not take place immediately. The icon appears where it is placed until the transfer is done. During this time, your application should set the cursor to the busy state. The user cannot move or select the icon until the transfer is complete; the busy cursor tells the user the transfer is in process.

Drag–and–Drop Mechanics

You should understand the following areas of the CDE underlying application architecture when designing a drag–and–drop operation:

- What type of object is being dragged
- What action takes place when the object is dropped
- How to match operations between source and destination applications

Types of Objects

There are three types of draggable objects in CDE:

- Files
- Buffers
- Text selections

Each application has its own objects that can be dragged and dropped. For example, Calendar uses appointments, Mailer uses mail messages, and File Manager uses folders and files.

The folder and file icons in File Manager exist as separate entities in the underlying file system and are, therefore, treated as files when dragged and dropped. However, Calendar appointments and Mailer messages do not exist as separate entities in the file system. When a user drags these

objects they are treated as buffers.

Text selections fall into a different category because selecting a piece of text is different than selecting an icon. The user selects a range of text in a document window; the text does not represent the whole document −− only a piece of a document. Rarely does a user see the piece of text as a distinct object and the user does not expect a piece of text to behave like an icon when dropped. For this reason, the drag−and−drop model for text mirrors the cut, copy, and paste operations available from the Edit menu.

Actions That Occur When Objects Are Dropped

There are two actions that can occur when an object is dropped: insert or load.

The insert action inserts the dropped object into the destination, adding it to the current data in the application or document. The object is inserted when a user schedules an appointment, prints a document, attaches a document, pastes text, or appends a mail message. Such an action is a move or copy operation, depending on the destination and the user. The user might decide to copy a piece of selected text as opposed to moving it. The drag icon should indicate whether the operation is a copy or move.

The load action operates the same as if the user had chosen Open from the File menu, selected a file, and clicked the Open button. The dropped object gets loaded into the application. The user can edit it and save changes back to the original file. Load works only with files at this time, not with buffers or text. The user should see the copy drag icon when dragging an object over a drop zone that supports the load action.

Load works with buffers; however, buffers are loaded as read−only.

Matching Drag−and−Drop Operations

When designing drag−and−drop operations for an application, you must understand how Motif decides which operation executes when the source and destination of a drag and drop do not match.

For each drag source, an application advertises which drag−and−drop operations are possible and on what destinations the drag source can be dropped. For each drag destination, the application advertises the possible sources and the types of operations. If a source and its destination have two or more operations in common, Motif follows a specific order to determine which operation to use. That order is move, copy, link. The application cannot change the operation that is accepted based on the type of item being dragged.

For example, application A might allow an element to be moved or copied. Application B might allow the destination to accept a copy or link. The intersection in this example is copy. If the destination in application B accepts move or copy, then the source is moved because the move operation comes first in the operation order.

In this example, the user could override the move operation by holding down a modifier key, for example `Ctrl`, to make the operation a copy. This will work if the copy operation is in the common set of operations. If the copy operation is not in the common set, then the drag becomes an invalid drag.

The only time you may have to consider matching operations is when you have a destination that accepts moves but functions better with copies. In that case, the destination should accept only copies because accepting a copy broadens the scope of acceptable drops. In most cases where a move is accepted, a copy would work just as well. Remember that move is implemented as a copy followed by a delete.

Determining Drag Sources

If you use drag and drop in an application, you must decide what control elements can be dragged and how those elements are to be represented. Typically, the user selects something like text or a file to drag, but the application may also allow a drag−and−drop of other elements, such as mail messages or appointments.

This section provides general guidelines for determining drag sources and then discusses specific elements that can be dragged.

Dragging from a Scrolling List

In CDE, items in a scrolling list are text objects by default. They can be buffer objects, but they cannot be both text and buffer objects. For example, the Calendar Appointment Editor has a scrolling list of appointments that the user can select and drag. When dragging an appointment, the user is manipulating a buffer and the drag icon shows an appointment icon as the source indicator (see Scrolling List with a Draggable Item). A Mailer container window has a list of mail messages in the upper portion of the window. Users can select and drag one or more messages from this list. These messages are actually buffers and the drag icon shows a mail message as the source indicator. If multiple mail messages are dragged, then the drag icon shows the multiple source indicator.

Figure 5 Scrolling List with a Draggable Item

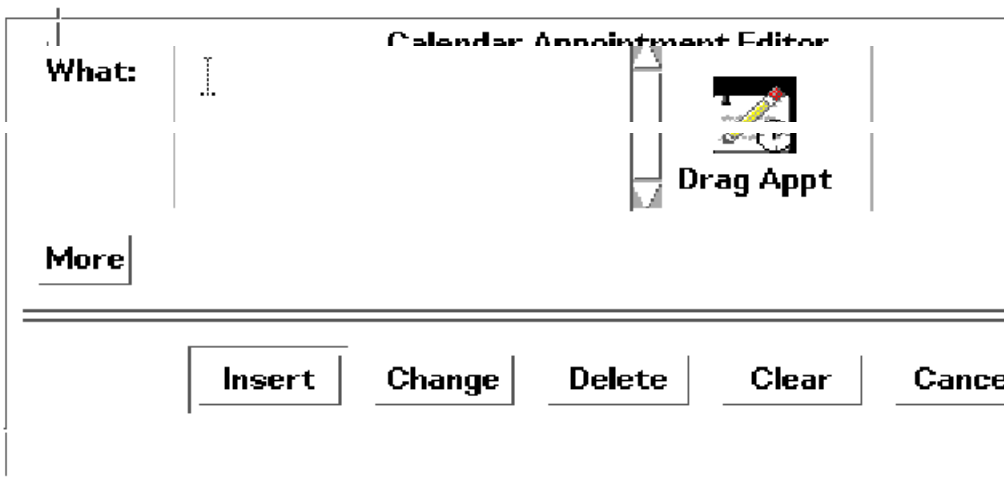
If your application uses a scrolling list to show mail message headers or to list other kinds of objects, then you need to integrate dragging with an extended selection model, as in the Mailer application.

Dragging from a Dialog Box

Sometimes the user needs to be able to drag from inside a dialog box. For example, in the Calendar Appointment Editor there are a series of text fields on the left side where the user enters information about an appointment. Allowing drags from this area lets the user drag text from the appointment description.

The recommended method for indicating that the user can drag something is to include a draggable icon graphic in the dialog box. This icon graphic must be the same icon that represents data in File Manager. In Calendar, the appointment icon is shown just as it would appear in File Manager (see Calendar Appointment Editor Dialog Box), with a label under it. This is the same icon that the drag icon source indicator uses. Place the icon graphic in the dialog box adjacent to the information to be dragged. The upper right corner of the dialog box or window is the default position, but you can change it for your application. In Calendar Appointment Editor, the icon is placed near the main text field to indicate that you can drag the text fields.

Figure 6 Calendar Appointment Editor Dialog Box



Dragging from a Window

In some applications, you might want to allow the user to drag an entire document or file. For example, in Icon Editor, you might want to allow the user to drag the file for an icon currently in the editor. You should incorporate a draggable icon graphic in the window of your application to indicate that the user can drag the document or file. In the case of Icon Editor, you can use one of the icons for displaying the contents of the icon file. Follow the guidelines for icons used in dialog boxes. For example, the icon should:

- Be the same icon used to represent the document in File Manager
- Be 32x32 pixels
- Have a label
- Be placed adjacent to the data being dragged
- Be used as the source indicator in the drag icon

Dragging and Dropping Multiple Selections

When the user selects more than one item to be dragged, change the drag icon to indicate there is more than one item selected.

Some drop zones may be able to accept only a single item. A drop zone cannot differentiate between a single and a multiple set of items being dropped. Since the drop icon does not display the cannot pointer, melt the items in and then have the destination application snap them back.

Follow the snap back with an error notice that tells the user why the drop failed.

Standard Supported Drop Zones

The standard supported drop zones in CDE are Front Panel controls, open windows, and some icons in File Manager, including folder and action icons. CDE does not support dropping on minimized icons and on File Manager icons that do not support drops.

Front Panel Drop Zones

The Front Panel is a collection of controls and other functions that provide the user with easy and fast access to the operating system. As a consequence, its drag−and−drop behaviors are heavily dependent upon the context of the destination. For example, if the destination is a printer, then the system should print it. If the destination is a subpanel, then the system should install it in the subpanel. Most applications will not vary in behavior quite as broadly as the Front Panel.

File Manager Drop Zones

File Manager for CDE allows the user to drop an icon on the desktop, where it becomes a reference. Within File Manager windows, File Manager allows dropping onto icons other than folders and action icons. For example, dropping a mail message icon onto a mail container icon appends the mail message.

When mail messages or calendar appointments, or other buffers, are dragged from the source application and dropped onto File Manager, they must be named. The underlying API supports a name field for the item being dragged. You should use this name as the name of the buffer. Create names that are consistent with the application from which it came. If there is no appropriate name, as in dropping a text selection in File Manager, name the resulting file "unnamed." If there is a name conflict, display a dialog box and ask the user to rename the dropped file.

Drop−Only Targets

CDE does not support the concept of a specific control or graphical target used only for drops. Any control in the human interface that has selectable items can be dropped upon and should provide drop−zone feedback. This includes data panes, scrolling lists, and text−entry fields. The operation that takes place upon the drop should be consistent with the user's expectations for that application type.

Drag−and−Drop Performance Guidelines

There are several points during a drag−and−drop operation when the timing and response to the user is critical. The responsibility for ensuring optimal performance belongs to the source and destination applications, as well as to the Motif Drag−and−Drop API and Drag−and−Drop Convenience API.

The following time line explains the individual user steps and system responses in a drag−and−drop operation. The suggested guideline for interaction timing is noted after the relevant step.

1. The user makes a selection. The pointer is over the selected object. The user presses and holds down the mouse button.
2. The user starts to move the pointer. The user should be able to move the pointer 10 pixels before a drag is initiated. If the user is pressing the TRANSFER button, there is no drag threshold.
   - Change the pointer to a drag icon when the drag is initiated.
   - The latency from hand movement to drag icon display should be less than 50 msec.
3. The user drags the object over a drop zone, crossing the boundary line with the hot spot on the drag icon.
   - Change the drag icon to the cannot pointer if it is not over a valid drop zone. Highlight the drop zone if it is a valid drop zone.
   - The latency from hand movement to drag icon display should be less than 50 msec.
4. The user drops the drag icon on the drop zone.
   If the drop zone is not valid, use the snap−back transition effect to snap the drag icon back to the source.

   If the drop zone is valid, use the melt−in transition effect to melt the drag icon into the destination.
   - The display latency from mouse button release to feedback echo should be less than 50 msec, with a maximum limit of 120 msec.
   - Transitional animations should run from 200 to 350 msec, with a maximum limit of 500 msec. The animation should run at the same speed, regardless of hardware conditions.
5. The destination application starts the data transfer.
   - Display a message to the user indicating that data transfer has started.
   - Indicate progress with further messages.
   - Indicate the completion of the data transfer to the user.

- If the data transfer fails, the destination application should provide the user with appropriate feedback as to why it failed.
- The latency from command invocation (drop occurred) to completion should be in the range of 0.3 to 1 second, with a maximum of 2 seconds.
- When a command might run longer than 2 seconds, display a busy cursor whenever the cursor is over the busy object. When possible, display partial results. The progress indicator or busy cursor should be displayed in less than 0.5 second.
- Display a status message or an in−progress message upon completion of the transitional animation that indicates the data transfer is taking place. For example: `Data transfer is 10% complete.` Update this message every 2 to 3 seconds until the transfer is 100% complete.
- If the data transfer fails, display a message, either in the status area or in an in−progress message, that indicates why the drop failed and what the user can do about it, if anything.

Using Attachments in Your Application

This section discusses the CDE user model and guidelines for attaching (or including) documents. You can see this functionality in the Mailer software application. If you plan to include an attachment list in the interface of your application, read this section.

Note: Note
This set of guidelines is not a description of an embedded document architecture.

In CDE, an attachment and attachment list are defined as follows:

attachment

An attachment is an object that is included with another object in order to complete it.

Suppose you had two documents called A and B. If document A is attached to document B then A continues to exist as a separate document that is "carried" by B. Show document A as an icon within document B. The user can open and view document A independently and can detach A from B at a later time, as if they were never attached at all.

attachment list

An attachment list is the area in which you can display attachments. An attachment list should be scrollable and include room for displaying icon labels.

Note: Note
Containers are an implementation concept that should not appear in the attachments interface. Therefore, do not use the term "container" to describe attachments to the user. (It may be an appropriate term elsewhere.)

Attachment User Model

Show attachments as icons where they are attached. These icons are the same icons as those used in File Manager and other places in CDE. The basic rule is that if the same icon is used in File Manager as in an attachment, then you should make every effort to ensure that the two behave the same in every situation.

There are three levels of functionality for attached documents:
1.Ability to attach and detach documents
2.Ability to open, view, and quit an attached document in a separate window
3.Ability to edit the attachment in a separate window and save changes back to the attached document

The goal is to provide level 3 functionality whenever possible. If you cannot provide this level, then degrade the attachment’s level of functionality in the steps shown. This section assumes you are providing level 3 functionality.

If a document provides significantly different functionality as an attachment from that provided as a File Manager icon, then provide a different icon for the attachment to clearly indicate to the user the difference in functionality.

What Can Be Attached?

You should determine for each application what items it can attach. For example, Mailer can attach documents, scripts, and applications, but not folders.

What is the Method for Attaching?

There are two methods of attachment:

- Through the file selection dialog box that comes up when you choose Add File from the Attachment menu
- Through drag and drop from File Manager or another application

What Happens When an Object Is Attached?

The act of attaching document A to document B copies the bits of document A into document B. There is no further connection with the original file. If the user opens the attached document and makes changes, the changes are saved back to the attached document only, not back to a file in the file system.

Attachments Within Attachments

The user can attach messages or text files that have attachments inside them. This is sometimes referred to as "nesting." For example, a text file might have a mail message icon that the user could open and that could have a text message and more attachments.

Editing and Saving Attachments

The user should be able to open an attachment, edit it, and save the changes back to the attachment. If the attachment cannot do this, then do not provide the Open action in the Actions portion of the menu when that attachment is selected and do not allow double−clicking to open the attachment.

Executing Attachments

When the user tries to open or double−click on an executable attachment, there may be times when you should ask the user to confirm this operation. Both the name of the attachment and the name of the action being taken on the attachment should be variables. An example error message follows:

```
"Invitation" is an executable attachment. Do you want to Run it?
Buttons: Run, Cancel, Help
```

Read−Only Attachments

The user can open read−only attachments for reading only. Indicate this state by deactivating the menus in the attachment application, deactivating the selection cursor, or by some other obvious method. At a minimum, you should dim the Save menu item in the attachment application.

Drag Load and Attachments

The user can accomplish a drag load in two ways. In applications that directly support drag load, the user can drag an icon from File Manager over the open window for that application and drop it. This should load the file represented by that icon. The same result can be accomplished by dropping an icon onto an action icon. The action starts an editor, which then loads the file represented by the icon. When an icon from File Manager is drag loaded, it is equivalent to choosing Open from the File menu. The user can then edit and save the open file.

The user can drag and drop an attachment onto editors or actions that support drag load but any edits made are not saved back to the attachment. Attachments, which are implemented as buffers, are loaded as read−only data.

When the user tries to save changes to a loaded attachment, the editor displays a file selection dialog box and asks the user to confirm the name and to choose a place in the file system to save the file. The name used in the file selection dialog box is the same as the attachment name. If the editor (command line application) cannot bring up a file selection dialog box, then you should clearly and visibly indicate to the user that the loaded file is read−only.

If the user wants to edit the attachment directly, the user must select the attachment in the attachment list, choose Open from the Attachment menu, or double−click on the attachment. This opens the attachment, allowing the user to edit and save changes.

Another option is to drag load an attachment, edit it, save it to a new file name, and replace the old attachment with the new one manually.

Widgets and Components Cross Reference

The following tables show how Motif widgets correspond to the components described in this guide. Gadgets, which are essentially performance –oriented versions of widgets, are not listed in this appendix but correspond closely to their widget counterparts.

Table 1 Basic Controls

Control	Motif Widget
Cascade button	XmCascadeButton
Check button	XmToggleButton with XmNindicatorType set to XmN_OF_MANY
Drop–down combination box	XmComboBox
Container	XmContainer
Label	XmLabel
Notebook	XmNotebook
Option menu button	An XmCascadeButton within an XmRowColumn with XmNrowColumnType set to XmMENU_OPTION
Push button	XmPushButton, XmArrowButton, XmDrawnButton
Radio button	XmToggleButton with XmNindicatorType set to XmONE_OF_MANY
Separator	XmSeparator
Spin box	XmSpinBox
Toggle button	XmToggleButton (it can also be simulated by XmPushButton and XmDrawnButton)

Table 2 Field Controls

Control	Motif Widget
Canvas	XmDrawingArea (without children), XmDrawnButton
List	XmList
Sash	Private element of XmPanedWindow
Scale	XmScale
Scroll bar	XmScrollBar
Text	XmText and XmTextField

Table 3 Basic Groups

Control	Motif Widget
Menu	XmRowColumn with XmNrowColumnType set to XmMENU_PULLDOWN or XmMENU_POPUP
Menu bar	XmRowColumn with XmNrowColumnType set to XmMENU_BAR
Panel	XmRowColumn with XmNrowColumnType set to XmWORK_AREA and composed of basic controls

Table 4 Layout Groups

Control	Motif Widget
Composite	XmBulletinBoard, XmDrawingArea (with children), XmForm, XmRowColumn when XmNrowColumnType is set to XmWORK_AREA and it is not a basic group
Split window (paned window)	XmPanedWindow

Table 5 Framing Groups

Control	Motif Widget
Group box	XmFrame
Main window	XmMainWindow
Scrolled window	XmScrolledWindow

Table 6 Dialog Boxes

Control	Motif Widget
Command box	XmCommand
Error message	XmMessageBox
File selection box	XmFileSelectionBox
Information message	XmMessageBox
Message	XmMessageBox
Prompt dialog	XmSelectionBox
Question dialog	XmMessageBox
Selection box	XmSelectionBox

Warning message	XmMessageBox
In−progress dialog	XmMessageBox

Keyboard Model and Key Bindings

This appendix contains tables that list the Motif model keyboard bindings.

Model Keyboard

Since not all keyboards are the same, it is difficult to give style guidelines that are correct for every manufacturer's keyboard. To solve this problem, this guide describes keys that use a "Motif model keyboard" mechanism. Wherever keyboard input is specified, the keys are indicated by the engraving they have on the Motif model keyboard. The model keyboard does not correspond directly to any existing keyboard; rather, it assumes a keyboard with an ideal set of keys.

In addition to the standard letter, number, and character keys, the Motif model keyboard is composed of the following special keys:

- The special printing characters `/`, `\`, and `!`
- The standard modifier keys `Ctrl`, `Alt`, and `Shift`
- Ten function keys `F1` through `F10`
- The arrow keys `[darr ]`, `[larr ]`, `[rarr ]`, and `[uarr ]`
- `Backspace`
- `Cancel`
- `Delete`
- `End`
- `Escape`
- `Help`
- `Home`, `Begin`, or both
- `Insert`
- `Menu`
- `PageDown`
- `PageUp`
- `Return`
- `Space`
- `Tab`

The Motif model keyboard also contains the following optional keys, which, although useful, are either not necessary or may be created by combinations of other keys:

- `CapsLock`
- `Copy`
- `Cut`
- `Enter`
- `ModeSwitch`
- `NumLock`
- `PageLeft`
- `PageRight`
- `Paste`
- `ScrollLock`
- `Select`
- `Undo`

Throughout this guide, behavior is described in terms of model keyboard keys. When a behavior takes advantage of an optional key from the model keyboard, it is also described in terms of the required special keys. Each of the keys described on the Motif model keyboard must be available either as specified or by using other keys or key combinations if the specified key is unavailable. The following are a few of the more important alternative key bindings:

- If `Cancel` does not exist, use `Escape`.
- If `Help` does not exist, use `F1`.
- If `Menu` does not exist, use `Shift F10`.
- If `F10` does not exist, use `Shift Menu`.
- If `Home` or `Begin` does not exist, use `Alt [larr ]`.
- If `End` does not exist, use `Alt [rarr ]`.

- Wherever you can use `Select` and `Space` for a selection action, you can also use `Ctrl Space`.
- Wherever you can use `Enter` and `Return` for activation, you can also use `Ctrl Return`.

Keyboard Function

Shortcut Keys for Menu Choices shows the key bindings for a unique choice on a menu.

Table 1 Shortcut Keys for Menu Choices

Menu Choice	Key Bindings	Menu
Bold	Ctrl B	Application specific
Close	Alt F4	Window
Copy	Copy, Ctrl C, Ctrl Insert	Edit
Cut	Cut, Ctrl X, Shift Delete	Edit
Delete	Backspace (in text), Delete	Edit or Selected (to Trash)
Deselect All	Ctrl \	Edit
Italics	Ctrl I	Application specific
New	Ctrl N	File or Selected
On Item [Help]	Shift Help, Shift F1	Help
Open	Ctrl O	File or Selected
Paste	Ctrl V, Shift Insert	Edit
Print	Ctrl P	File
Properties	Ctrl I	Selected
Redo	Shift Undo, Ctrl Y, Alt Shift Backspace	Edit
Repeat	Shift Undo, Ctrl Y, Alt Shift Backspace	Edit
Save	Ctrl S	File
Select All	Ctrl A, Ctrl /	Edit
Underline	Ctrl U	Application specific
Undo	Undo, Ctrl Z, Alt Backspace	Edit

Navigation and Activation Key Bindings shows navigation and activation key bindings and their function.

Table 2 Navigation and Activation Key Bindings

Operation	Key Bindings	Function
Activate Choice	Select, Ctrl Space	Activates an action or dialog choice.
Adjust Begin Data	Ctrl Shift Home	Adjusts a selection to the beginning of the data.
Adjust Begin Line	Shift Home	Adjusts a selection to the beginning of a line.
Adjust Down	Shift [darr]	Adjusts a selection down a line.
Adjust End Data	Ctrl Shift End	Adjusts a selection to the end of the data.
Adjust End Line	Shift End	Adjusts a selection to the end of a line.
Adjust Left	Shift [larr]	Adjusts a selection to the left.
Adjust Page Down	Shift PageDown	Adjusts a selection down a page.
Adjust Page Left	Shift PageLeft, Ctrl Shift PageUp	Adjusts a selection a page to the left.
Adjust Page Right	Shift PageRight, Ctrl Shift PageDown	Adjusts a selection a page to the right.
Adjust Page Up	Shift PageUp	Adjusts a selection up a page.
Adjust Paragraph Down	Ctrl Shift [darr]	Adjusts a selection down a paragraph.
Adjust Paragraph Up	Ctrl Shift [uarr]	Adjusts a selection up a paragraph.
Adjust Right	Shift [rarr]	Adjusts a selection to the right.
Adjust Selection	Shift Select, Shift Space (not in text), Ctrl Shift Space	Adjusts a selection to the cursor position.
Adjust Up	Shift [uarr]	Adjusts a selection up a line.
Adjust Word Left	Ctrl Shift [larr]	Adjusts a selection by a word to the left.
Adjust Word Right	Ctrl Shift [rarr]	Adjusts a selection by a word to the right.
Begin Data	Ctrl Home	Navigates to the beginning of the data.
Begin Line	Home	Navigates to the beginning of a line.
Cancel	Cancel, Escape	Removes a menu or dialog or cancels a direct manipulation operation.
Default Activate	KeypadEnter, Enter (not in text), Ctrl Enter	Performs a default action; activates a menu control.
Delete End Line	Ctrl Delete	Deletes to the end of the line (optional).
Delete Next Character	Delete	Deletes the next character.
Delete Previous Character	Backspace	Deletes the previous character.
Drop Down	Alt [darr]	Drops down a drop–down list or combination box.
End Data	Ctrl End	Navigates to the end of the data.
End Line	End	Navigates to the end of a line.

Help	Help, F1	Obtains help on the cursored item.
Navigate Down	[darr], Ctrl [darr]	Navigates down between elements.
Navigate Left	[larr], Ctrl [larr]	Navigates left between elements.
Navigate Menu Bar	Shift Menu, F10	Navigates to the menu bar.
Navigate Right	[rarr], Ctrl [rarr]	Navigates right between elements.
Navigate Up	[uarr], Ctrl [uarr]	Navigates up between elements.
Next Family Window	Alt F6	Navigates to the next window in the window family.
Next Tab Group	Tab (where possible), Ctrl Tab	Navigates to the next tab group.
Next Window	Alt Tab, Alt Escape	Navigates to the next window family or root icon.
New Line	Enter	Inserts a new line in text.
Page Down	PageDown	Navigates downwards by a page.
Page Left	PageLeft, Ctrl PageUp	Navigates a page to the left.
Page Right	PageRight, Ctrl PageDown	Navigates a page to the right.
Page Up	PageUp	Navigates up by a page.
Paragraph Down	Ctrl [darr]	Navigates down a paragraph.
Paragraph Up	Ctrl [uarr]	Navigates up a paragraph.
Pop–Up Menu	Menu, Shift F10	Invokes a pop–up menu.
Pop–Up Workspace Menu	Alt Menu, Alt F10	Invokes a pop–up menu for the workspace.
Previous Family Window	Alt Shift F6	Navigates to the previous window in a window family.
Previous Tab Group	Shift Tab (where possible), Ctrl Shift Tab	Navigates to the previous tab group.
Previous Window	Alt Shift Tab, Alt Shift Escape	Navigates to the previous window family or root icon.
Primary Move	Alt Cut, Alt Ctrl X, Alt Shift Delete	Moves a primary selection to the cursor position.
Primary Copy	Alt Copy, Alt Ctrl C, Alt Ctrl Insert	Copies a primary selection to the cursor position.
Select	Select, Space (not in text), Ctrl Space	Makes a selection (selects or toggles, depending on mode).
Show Tasks	Ctrl Esc	Raises and normalizes the icon window.
Space	Space, Shift Space	Inserts a space in text.
Stop	Cancel, Escape	Ends a task and removes an in–progress message window.
Tab	Tab	Inserts a tab or navigates to the next tab stop.
Toggle Choice	Select, Space, Ctrl Space	Toggles a value choice.
Toggle Add Mode	Shift F8	In editing controls, switches between normal mode and add mode.
Toggle Insert Mode	Insert	In text, toggles between replace mode and insert mode (optional).
Window Menu	Alt Select, Alt Space, Shift Escape	Pulls down the window menu.
Word Left	Ctrl [larr]	Navigates a word to the left.
Word Right	Ctrl [rarr]	Navigates a word to the right.

Nonprinting Key Key Bindings shows the key bindings for all nonprinting keys. These include directional keys, editing keys, and function keys.

Table 3 Nonprinting Key Key Bindings

Key	Key Only	Shift	Ctrl	Ctrl Shift	Alt	Alt Shift	Alt Ctrl
[rarr]	Navigate Right	Adjust Right	Navigate Right, Word Right	Adjust Word Right			
[larr]	Navigate Left	Adjust Left	Navigate Left, Word Left	Adjust Word Left			
[darr]	Navigate Down	Adjust Down	Navigate Down, Paragrah Down	Adjust Paragraph Down	Drop Down		
[uarr]	Navigate Up	Adjust Up	Navigate Up, Paragraph Up	Adjust Paragraph Up			
Backspace	Delete Previous Character (in text)				Undo	Redo	Repeat
Cancel	Cancel or Stop						
Copy	Clipboard Copy				Primary Copy		
Cut	Clipboard Cut				Primary Move		
Delete	Delete Next Character (in text)	Clipboard Cut	Delete End Line			Primary Move	
End	End Line	Adjust End Line	End Data	Adjust End Data			
Enter	Default Activate or New Line		Default Activate				
Escape	Cancel or Stop	Invoke Window Menu	Show Tasks		Next Window	Previous Window	
Help	Help	On Item [Help]					
Home	Begin Line	Adjust Begin Line	Begin Data	Adjust Begin Data			

Insert	Toggle Insert Mode	Clipboard Paste	Clipboard Copy				Primary Copy
Menu	Invoke a Pop–Up Menu	Navigate Menu Bar			Pop Up Workspace Menu		
PageDown	Page Down	Adjust Page Down	Page Right	Adjust Page Right			
PageLeft	Page Left	Adjust Page Left					
PageRight	Page Right	Adjust Page Right					
PageUp	Page Up	Adjust Page Up	Page Left	Adjust Page Left			
Paste	Clipboard Paste						
Select	Activate Choice, Select, or Toggle Choice	Adjust Selection			Invoke Window Menu		
Space	Activate Choice, Select, Space, or Toggle Choice	Adjust Selection, Space	Select	Adjust Selection	Invoke Window Menu		
Tab	Next Tab Group or Tab (in text)	Previous Tab Group	Next Tab Group	Previous Tab Group	Next Window	Previous Window	
Undo	Undo	Redo, Repeat					
F1	Help	On Item [Help]					
F4				Close			
F6					Next Family Window	Previous Family Window	
F8		Toggle Add Mode					
F10	Navigate Menu Bar	Invoke a Pop–Up Menu			Invoke a Workspace Pop –Up Menu		

Mouse Techniques

This appendix contains tables that list mouse operations and their button bindings.

Mouse Model and Buttons

On a 1–, 2–, or 3–button mouse, the mouse buttons are assigned to various functions, which are defined in tables later in this appendix. Often a 2–button mouse can use chording as a way to simulate a 3–button mouse, where MB1+MB2 (chorded) would be equal to MB3. Motif supports two different mouse models:

- Separate selection and transfer operations, where:
  - MB1 is used only for selection and activation
  - MB2 is used only for data transfer and direct manipulation
- Integrated select and transfer operations, where MB1 is used both for selection and activation and for data transfer and direct manipulation

The actual way that the mouse buttons are assigned depends on the number of mouse buttons available, as well as whether or not selection and transfer operations are integrated or separate. Regardless of the way that the buttons are actually assigned, a number of *''virtual''* mouse buttons are defined as follows:

SELECT

A virtual mouse button used for selection and activation. SELECT is always MB1 (the leftmost mouse button for a right–handed person).

ADJUST

A virtual mouse button used for adjusting a selection. ADJUST is always `Shift` MB1. In addition, on a 3–button mouse, with integrated selection and transfer, ADJUST may optionally be assigned to MB2.

TRANSFER

A virtual mouse button that may be used for data transfer and manipulation operations. With separate selection and transfer, TRANSFER is always assigned to MB2. With integrated selection and transfer, TRANSFER is MB1 (integrated with SELECT), and on a 3–button mouse may also optionally be assigned to MB2.

MENU

A virtual mouse button used to select pop–up menus:

On a 3–button mouse, MENU is always assigned to MB3.
On a 2–button mouse with integrated selection and transfer, MENU is assigned to MB2.
If neither of the previous applies, MENU is assigned to `Alt` MB1.

Assignments for 2– or 3–Button Mouse with Separated Selection and Transfer lists the virtual mouse buttons assigned to a 2– or 3–button mouse with separated selection and transfer.

Table 1 Assignments for 2– or 3–Button Mouse with Separated Selection and Transfer

Operation Name	Binding
SELECT	MB1
ADJUST	`Shift` MB1
TRANSFER	MB2
MENU	MB3 on a 3–button mouse, or `Alt` MB1 on a 2–button mouse.

Assignments for 1–, 2–, or 3–Button Mouse with Integrated Selection and Transfer lists the virtual mouse buttons assigned to a 1–, 2–, or 3–button mouse with integrated selection and transfer.

Table 2 Assignments for 1–, 2–, or 3–Button Mouse with Integrated Selection and Transfer

Operation Name	Binding
SELECT	MB1 (integrated with TRANSFER)
ADJUST	`Shift` MB1. This may also optionally be MB2 on a 3–button mouse.
TRANSFER	MB1 (integrated with SELECT). This may also optionally be MB2 on a 3–button mouse.
MENU	This may be one of the following: • MB3 on a 3–button mouse • MB2 on a 2–button mouse • `Alt` MB1 on a 1–button mouse

Note: Note
On a 3–button mouse with integrated selection and transfer, if neither ADJUST nor TRANSFER are assigned to MB2, MB2 may be used for application–defined purposes.

Mouse Operations and Functions lists the mouse operations and the functions they actually perform.

Table 3 Mouse Operations and Functions

Operation Name	Function
Activate	Activates a control that does not have selections.
Default Activate	Selects and performs a default action on an item.
Open	Opens a view that corresponds to an icon.
Manipulate	Manipulates nonselectable aspects of the interface (such as scroll).
Focus–Only Navigation	Moves the cursor to a component or element.
Point Select	Selects an item if the pointer is on one, deselecting other items.
Browse Select	Shows which items can be selected, selecting the one on which the pointer is released.
Group Click Select	Selects a group of elements.
Group Swipe Select	Selects a group of elements.
Point Toggle	Toggles the selection state of an item.
Group Click Toggle	Toggles the selection state of a group of elements.
Group Swipe Toggle	Toggles the selection state of a group of elements.
Adjust Click	Adjusts the current selection region.
Adjust Swipe	Adjusts the current selection region.
Select Word	Selects a word in text.
Range Click Select Word	Selects a range of words.
Range Swipe Select Word	Selects a range of words.
Toggle Word	Toggles the selection of a word.
Range Click Toggle Word	Toggles a range of words.
Range Swipe Toggle Word	Toggles a range of words.
Adjust Click Word	Adjusts the selection to a word boundary.
Adjust Swipe Word	Adjusts the selection in word increments.
Primary Copy	Copies a primary selection to the pointer location.
Primary Move	Moves a primary selection to the pointer location.
Primary Link	Links a primary selection to the pointer lacation.
Quick Copy	Makes and copies a secondary selection to a destination.
Quick Move	Makes and moves a secondary selection to a destination.
Quick Link	Makes and links a secondary selection to a destination.
Drag Transfer	Transfers dragged item(s) to the pointer location (usually a move).
Drag Copy	Copies dragged item(s) to the pointer location.
Drag Move	Moves dragged item(s) to the pointer location.
Drag Link	Links dragged item(s) to the pointer location.
Click Drag Transfer	Transfers dragged item(s) to the pointer location.
Click Drag Copy	Copies dragged item(s) to the pointer location.
Click Drag Move	Moves dragged item(s) to the pointer location.
Click Drag Link	Links dragged item(s) to the pointer location.
Pop–Up Menu Display	Displays a pop–up menu.
Pop–Up Menu Choose Swipe	Displays and makes choices in a pop–up menu.
Pop–Up Menu Choose Click	Displays and then chooses from a pop–up menu.
Cascaded Menu Display	Displays a cascaded menu.
Cascaded Menu Choose Swipe	Displays and makes a choice in a cascaded menu.
Cascaded Menu Choose Click	Displays and then chooses from a cascaded menu.
Cascaded List Display	Displays a cascaded list.
Cascaded List Choose Swipe	Displays and makes a choice in a cascaded list.
Cascaded List Choose Click	Displays and then chooses from a cascaded list.

SELECT and ADJUST Bindings lists the mouse operations for the SELECT and ADJUST button bindings.

Table 4 SELECT and ADJUST Bindings

Operation Name	Key Bindings
Activate	Click SELECT
Default Activate	Double–click SELECT
Open	Double–click SELECT
Manipulate	Press SELECT, move mouse, release SELECT
Focus–Only Navigation	Click `Ctrl` SELECT
Point Select	Click SELECT
Browse Select	Press SELECT, move mouse, release SELECT

Group Click Select	Click SELECT, move mouse, click ADJUST
Group Swipe Select	Press SELECT, move mouse, release SELECT
Point Toggle	[In select mode] Click `Ctrl` SELECT
Group Click Toggle	[In select mode] Click `Ctrl` SELECT, move mouse, click ADJUST
Group Swipe Toggle	[In select mode] Press `Ctrl` SELECT, move mouse, release SELECT
Point Toggle	[In toggle mode] Click SELECT
Group Click Toggle	[In toggle mode] Click SELECT, move mouse, click ADJUST
Group Swipe Toggle	[In toggle mode] Press SELECT, move mouse, release SELECT
Adjust Click	Click ADJUST (or `Ctrl` ADJUST)
Adjust Swipe	Press ADJUST (or `Ctrl` ADJUST), move mouse, release ADJUST
Select Word	Double−click SELECT
Range Click Select Word	Double click SELECT, move mouse, click ADJUST
Range Swipe Select Word	Double−press SELECT, move mouse, release SELECT
Toggle Word	Double−click `Ctrl` SELECT
Range Click Toggle Word	Double−click `Ctrl` SELECT, move mouse, click ADJUST
Range Swipe Toggle Word	Double−press `Ctrl` SELECT, move mouse, release SELECT
Adjust Click Word	Double−click ADJUST (or `Ctrl` ADJUST)
Adjust Swipe Word	Double−press ADJUST (or `Ctrl` ADJUST), move mouse, release ADJUST
Cascaded Menu Display	Click SELECT
Cascaded Menu Choose Swipe	Press SELECT, move mouse, release SELECT
Cascaded Menu Choose Click	Click SELECT, move mouse, click SELECT
Cascaded List Display	Click SELECT
Cascaded List Choose Swipe	Press SELECT, move mouse, release SELECT
Cascaded List Choose Click	Click SELECT, move mouse, click SELECT

TRANSFER Bindings lists the TRANSFER button bindings. The TRANSFER button is the virtual mouse button that may be used for data transfer and manipulation operations.

Table 5 TRANSFER Bindings

Operation Name	**Key Bindings**
The following set of bindings are defined only when TRANSFER is assigned to MB2:	
Manipulate	Press TRANSFER, move mouse, release TRANSFER
Primary Copy	Click TRANSFER (or `Ctrl` TRANSFER)
Primary Move	Click `Shift` TRANSFER
Primary Link	Click `Ctrl Shift` TRANSFER
Quick Copy	Press `Alt` TRANSFER, move mouse, release TRANSFER
Quick Copy	Press `Alt` TRANSFER, move mouse, press `Ctrl` and release TRANSFER
Quick Move	Press `Alt` TRANSFER, move mouse, press `Shift` and release TRANSFER
Quick Link	Press `Alt` TRANSFER, move mouse, press `Ctrl Shift` and release TRANSFER
The following set of bindings are always defined. When selection and transfer are integrated, the bindings are used for transfer versus selection:	
Drag Transfer	Press TRANSFER, move mouse, release TRANSFER
Drag Copy	Press TRANSFER, move mouse, press `Ctrl` and release TRANSFER
Drag Move	Press TRANSFER, move mouse, press `Shift` and release TRANSFER
Drag Link	Press TRANSFER, move mouse, press `Ctrl Shift` and release TRANSFER
The following set of bindings are not available on a 1−button mouse:	
Click Drag Transfer	Click `Alt` TRANSFER, move mouse, click TRANSFER
Click Drag Copy	Click `Alt` TRANSFER, move mouse, click `Ctrl` TRANSFER
Click Drag Move	Click `Alt` TRANSFER, move mouse, click `Shift` TRANSFER
Click Drag Link	Click `Alt` TRANSFER, move mouse, click `Shift Ctrl` TRANSFER

MENU Bindings lists the mouse operations and the MENU button bindings. The MENU button is the virtual mouse button used to display pop−up menus.

Table 6 MENU Bindings

Operation Name	**Key Bindings**
Pop−Up Menu Display	Click MENU
Pop−Up Menu Choose Swipe	Press MENU, move mouse, release MENU
Pop−Up Menu Choose Click	Click MENU, move mouse, click MENU (or SELECT)
Cascaded Menu Display	Click MENU

Cascaded Menu Choose Swipe	Press MENU, move mouse, release menu
Cascaded Menu Choose Click	Click MENU, move mouse, click MENU (or SELECT)

Glossary

About <appname>

An action choice that displays a secondary window that contains information about an application, such as its copyright notice, logo, and version number.

acceleration

The ratio of the rate of movement of the pointer to the rate of mouse movement when in motion.

accelerator

A key or sequence of keys (typically a modifier key and some other key) that provides a shortcut, immediately accessing a program function. *See also* shortcut key.

action

1. In awk, lex, and yacc, a C language program fragment that defines what the program does when it recognizes input.
2. A procedure associated with a widget and invoked by the Xt event dispatcher when the widget receives an event of a given type. The widget's translation table associates event descriptions with actions.
3. A desktop construct that provides a method for running applications, executing commands, and performing other activities such as printing, removing files, and changing directories.

action choice

A type of choice used to invoke an action where further specification of parameters for invocation of the action are not required. There are two types of action choices: command choices and dialog choices.

action message

A message that indicates that a condition requiring a response from the user has occurred.

activatable element

An element that invokes or initiates an action when the SELECT button is used. Activatable elements represent action choices, dialog choices, and cascading choices.

activate

To initiate the activity associated with a choice.

activation

1. Invocation of a component's primary action. For example, the user activates a push button by using the SELECT button while positioned on the activatable push button.
2. The process of initiating the activity associated with a choice.

activation preview

A special user assist action that applications can use. In an application that supports activation preview, when a user presses and holds the SELECT button over a push button or toggle button, information is presented that describes the effect of activating the control. Also referred to as *previewing*.

active cursor

The cursor displayed in or on the control that has focus, sometimes referred to as a *highlight cursor* or *selection cursor*.

active text region

A region of text that acts like an activatable element.

active window

1. The window that is currently selected to receive the input focus. Only one window can be active at a time.

2.A window that receives keyboard input.
3.A window that is differentiated from other windows in the workspace by a distinctive title−bar color or shade.

See also focus.

add mode

A keyboard−based selection mode in which navigation in a scope of selection does not affect the current selection and in which selection techniques toggle the selection state of identified elements.

ADJUST button

The assigned button on a pointing device that the user presses to adjust the current selection region in a selection scope. It is always bound to MB1 augmented by the `Shift` modifier, but may also be bound to MB2.

adjust click technique

An adjustment technique that allows the user to enlarge or reduce a selected range or area, depending on the location of the point relative to the selection.

adjust swipe technique

An adjustment technique that allows the user to enlarge or reduce a selected range or area by moving the mouse or cursor.

adjusted toggling

Use of an adjustment technique following a selection technique that toggles the selection state of the identified elements.

adjustment policy

A selection policy that determines how the current selection region is to be adjusted by an adjustment technique. The three possibilities are a reselect policy, an enlarge−only policy, or a balance−beam policy.

adjustment technique

A selection technique used to adjust the current selection region.

anchor

A position in a collection of selectable objects that marks one end point of an extended selection range.

anchor element

An element identified by a selection technique for later use with an adjustment technique.

anchor inclusion policy

A selection policy that determines whether to enlarge an adjusted selection region to include an anchor element or region.

anchor point

A point identified by a selection technique for later use with an adjustment technique.

anchor toggle

A discontiguous, mouse−based selection toggling technique in which the selection state of the elements in a range is toggled to the inverse of the initial state of the anchor element. *See also* full toggle.

anomaly

A kind of contrast: the irregular introduced into a regular pattern. A deviation from a normal expected task.

anti−alias

A technique employed to smooth the stair−stepped appearance of pixel graphics that include curved or diagonal lines.

application

1.The use to which program code is put, such as payroll or inventory.
2.A computer program that provides tools to do work.

The terms *application* and *product* are sometimes used interchangeably. *See also* program.

application developer

A person who develops code for a set of programs that are collectively used as an application.

application group

A container in the Application Manager where related accessible applications in the desktop are placed.

Application Manager

The software application that manages the tools and other software applications available to the user.

application modal

A state of a window that limits user interactions within windows owned by the same application. *See also* system modal.

applications menu

A list of options within an application.

Apply

An action choice that appears in a window and makes the changes indicated in the window without closing it. This choice often appears on a push button.

area click technique

An area technique in which two separate mouse or keyboard operations are used to indicate the corners of the area.

area inclusion policy

A selection policy that determines whether or not elements only partially within an area are included in the current selection region.

area swipe technique

An area technique in which the corners of the rectangular area are indicated by moving the mouse or cursor from one corner of the rectangle to the other.

area technique

A group selection technique in which the user selects elements within a rectangular area by indicating the opposite corners of the rectangle.

armed emphasis

See ready emphasis.

arrow button

1.An activatable element that contains an arrow graphic.
2.An element of a scroll bar used to scroll a window by small increments.
3.A control on the CDE Front Panel that is used to slide up or down a subpanel.

arrow keys

The four directional keys on a keyboard. *See also* navigation keys.

attachment

1.An encapsulated data object inside a document.

2.Mailer: A data object within an electronic mail message that is displayed as an icon in the Attachments list. An attachment can be an image, data, or executable file. Multiple messages can be added (attached) to a single email message. An attached message is displayed or activated by selecting it.

attachment list

The area where you can display attachments. An attachment list should be scrollable and include room for displaying icon labels.

audible cue

An audible warning signal. *See also* audible signal, cue.

audible signal

A sound generated by the operating environment for use as an audible warning signal. An audible signal is generally used in a quiet environment where it can be effective. *See also* audible cue.

augmented toggling

In select mode, the use of augmenting mouse−based selection techniques with the `Ctrl` or `Alt` modifier to toggle the identified elements.

automatic stacking order

A model in which a window or element is raised to the top of the stacking order of windows when it gains focus. *See also* manual stacking order.

autoscroll

Scrolling that automatically occurs during a related user interaction, such as selection or drag−and−drop transfer.

available

1.Used to describe a choice that can be activated or toggled by a user.
2.Used to describe a control that can be manipulated by the user.

backdrop

The pattern that covers the workspace background.

background

A location within a selection scope not covered by any element that can be selected.

balance−beam policy

An adjustment policy in which the current selection region is adjusted relative to the endpoint furthest from the point at which an adjustment is initiated.

bidirectional language support

Supporting languages that read both left to right (such as English) and right to left (such as Arabic, Hebrew, or Urdu).

binding

A mapping of functions to variables, buttons, or controls. On a 3−button mouse, the virtual buttons SELECT, TRANSFER, and MENU are considered to be bound to MB1, MB2, and MB3, respectively.

browse selection model

A selection model in which only a single element can be selected, using the browse technique.

browse technique

An individual selection technique in which moving the mouse or cursor to an element selects it.

button

A generic term for a window control. *See also* push button.

Cancel

An action choice that removes a secondary window without applying any changes made in that window.

cancel action

An action that terminates the current task or user interaction or exits from a special mode and restores the application state, if possible, to that preceding the start of the task, user interaction, or interface mode.

cannot pointer

A predefined pointer. It indicates that the operation the user is trying is not currently possible. For example, a cannot pointer is used when the user tries to drag an object that cannot be deleted to a trash can.

cascaded control

A control that is temporarily displayed via a cascading choice.

cascaded list

A list that is a spring–loaded, cascaded control.

cascaded menu

A menu that is a spring–loaded, cascaded control.

cascading button

A control that causes a menu to drop down from a menu choice.

cascading choice

A type of choice that, when activated, displays a cascaded control.

Change View

A cascading choice that displays a menu from which the user can choose the view to be presented within the window.

check box

A control used to set values that are not mutually exclusive, such as on, off, or indeterminate.

check mark

A graphic that indicates that a value choice, such as a check box, is set.

choice

1. An option in a pop–up menu or a menu used to influence the operation of the system.
2. An alternative displayed as a label (text or graphics) on the screen that a user can choose. Choices are available via controls through which a user chooses values or invokes actions, represented on push buttons, in menu items, with check boxes, and so on. There are four types of choices:
   - Action choices
   - Dialog choices
   - Cascading choices
   - Value choices

chord

1. To press more than one button on an input device while the motion of the pointer is within the limits specified by the operating environment.
2. In graphics, a short line segment whose end points lie on a circle. Chords are a means for producing a circular image from straight lines. The higher the number of chords per circle, the smoother the circular image.

Clear

An action choice that removes selected elements without compressing the visible space they occupied.

Clear to Trash

An action choice that removes selected elements (generally objects) without compressing the visible space they occupied and puts them into the trash.

click

1. A mouse technique that involves pressing and releasing a mouse button without moving the pointer. (Pressing and releasing most mouse buttons makes a clicking sound.)
2. To press and release a button on a pointing device without significantly moving the pointer and within a time specified by the operating environment.

client area

1. The area within the borders of a primary window's frame that an application controls.
2. The area of a window inside the window frame. The client area can contain one or more viewing areas, a menu bar, one or more palettes, a command area, a status area, and an information area.

clipboard

1. Any device used to store text or graphics during cut–and–paste operations.
2. An area of storage provided by the operating system to hold data temporarily.

clipboard transfer

A transfer technique in which the user is able to transfer data to and from an intermediate storage area called a *clipboard*.

Close

An action choice that removes a window and all of the windows associated with it from a screen.

collection

A group of objects (for example, a list), the contents of a File Manager view area, and so on.

column heading

1. Text appearing near the top of a column of data for the purpose of identifying or titling the data in the column.
2. A label placed above a table column that identifies the contents of the column.

combination box

A combination text–list control in which both the text field and the list box are visible at all times.

combination text–list control

A control that combines the functions of a text–entry field and a list box.

command area

An area in a window that provides a place for the user to enter commands.

command box

A combination text–list control that allows the user to review previous commands in the list box and to reissue a previous command or to issue a new command entered into the text–entry field.

command choice

An action choice used to immediately invoke an action.

command dialog

A dialog that contains a command box, possibly with other controls added to it.

container

A control whose specific purpose is to display objects as icons and to allow them to be selected and operated upon.

context–sensitive help

Help information about the specific choice or object that the cursor or pointer is on. The help is context sensitive because it provides information about the element in its current context.

contiguity policy

A selection policy (used primarily in text) that specifies whether a selection scope allows discontiguous selections.

Continue

An action choice that resumes a task that has been interrupted by the operating environment when the user can proceed as originally requested.

control

A visually recognizable element or group of elements that the user interacts with in a well–defined way. Avoid using this term in documentation; instead, describe the specific type of control.

control bar

A name for the palette at the top of the window and below the menu bar.

control navigation

Navigation among controls within a tab group.

Copy

1.An action choice that copies selected elements to the clipboard.
2.An action choice used in the context of a specific transfer mechanism to indicate that the transfer should result in a copy.

Copy Link

An action choice that places a link to selected elements on the clipboard.

Copy Special

A dialog choice that displays a dialog through which a user can choose the formats in which to copy selected elements to the clipboard.

Copy To

A dialog choice that displays a file selection dialog in which the user can specify where copies of files or objects should be placed.

count policy

A selection policy that specifies whether more than one element at a time can be selected.

cue

Information provided to inform users and orient them as they interact with the interface. A cue can be transient (in which case it is termed a warning signal) or persistent. Persistent cues can be used to direct the user's attention to a part of the screen or user interface, to indicate a particular state of an object, or to alert the user about potentially serious situations.
See also audible cue, graphical cue, persistent cue, visual cue.

current selection

The elements selected within a specified selection scope.

current selection region

The elements in a selection scope identified by the most recently used selection technique in that scope.

cursor

1. A movable symbol (such as an underline) on a display that indicates to the user where the next typed character will be placed or where the next action will be directed.
2. A marker that indicates the current data access location within a file.
3. A graphical image, usually a pipe (|) or block, that shows the location where text will appear on the screen when keyboard keys are pressed or where a selection can be made.
4. A visual cue that indicates where the user's interaction with the keyboard will be performed.

cursored element

The element that the cursor is on. *See also* element cursor.

Cut

An action choice that removes selected elements and places them onto the clipboard. The space they occupied is usually filled by the remaining elements in the scope of selection.

data element

An element whose appearance represents its contents, for example, a character or a number.

data transfer

The interaction of the user with elements, such as objects, data, and menu items, to perform tasks.

deactivate

Applied to spring–loaded controls, removes all spring–loaded controls in the system.

default action

The action associated with a window that the user would most likely want to invoke in a given situation when focus is in that window. The default action may change as the focus and state of the window changes. The default action is generally activated when a user double–clicks the SELECT button, presses `Enter` (except when either operation is used for other purposes), or presses `Ctrl Enter` or `keypadEnter`.

default choice

When navigating to a menu, this is the choice on that menu that gets focus.

default emphasis

The emphasis on a choice used to indicate that it would be activated if the user requested the default action.

Delete

An action choice that removes selected elements. The space they occupied is usually filled by the remaining elements in the scope of selection.

Delete to Trash

An action choice that removes selected elements (generally objects) and puts them into the trash. The space they occupied is usually filled by the remaining elements in the scope of selection.

Deselect All

An action choice that removes selected elements (generally objects) and puts them into the trash. The space they occupied is usually filled by the remaining elements in the scope of selection.

deselection

The process of removing selection emphasis from a previously selected element.

deselection policy

A selection policy that determines whether the user is allowed to deselect all elements or not.

desktop

See workspace.

desktop environment

1. A set of conditions and tools that a user can depend on when interacting with a system.
2. A graphical user interface for all flavors of UNIX. The Common Desktop Environment (CDE) is being adopted as a standard operating environment by many companies in the UNIX workstation market.

details view

A kind of view of a container in which details associated with each object displayed are presented in columns.

device

1. A mechanical, electrical, or electronic machine that is designed for a specific purpose and that attaches to your computer, such as a printer, plotter, or disk drive.
2. A physical or logical device, such as a printer, scanner, mouse, or joystick, that is accessible from the operating environment.

dialog

1. A widget that provides a means of communicating between the user and the application. A dialog is a pop−up control that usually asks a question or presents some information to the user. A dialog can be modal (suspending the application until the user provides a response) or modeless (allowing the user to interact with the application during the dialog).
2. In an interactive system, a series of related inquiries and responses similar to a conversation between two people.
3. A specialized interaction with the user that occurs in a secondary window.

dialog box

A window that an application displays and that requires user input. *See also* dialog window, secondary window.

dialog choice

A type of choice used to display a dialog to gather parameters for invocation of an action.

dialog window

A secondary window used for specialized interaction with the user.

dimmed

Reduced contrast to show unavailable emphasis.

dimmed emphasis

See unavailable emphasis.

direct editing

A mode of interaction in which the contents of a normally noneditable textual label is made available for editing.

direct manipulation

A transfer technique that allows the user to perform actions on elements by interacting directly with the elements. An example is dragging an element with a mouse and dropping it onto another element. *See also* indirect manipulation.

directional keys

See arrow keys and navigation keys.

directory

In a graphical desktop environment, a directory may also be referred to as a *folder*.

1. A logical unit for storing entries under one name (the directory name) in a given namespace. In addition to object entries, a directory can contain soft links and child pointers. You can copy, delete, and control access to a directory. Each

physical instance of a directory is called a replica.
2. A collection of open systems that cooperate to hold a logical database of information about a set of objects in the real world.
3. A type of file that contains the names and controlling information for other files or other directories.
4. A table of identifiers and references to the corresponding items of data.
5. An index that a control program uses to locate blocks of data that are stored in separate areas of a data set in direct access storage.
6. Information containers, like files. However, instead of text or other data, directories contain files and other directories. In addition, directories are hierarchically organized; that is, a directory may have a parent directory *''above''* it and may also have subdirectories *''below''* it. Similarly, each subdirectory can contain other files and also can have more subdirectories. Because they are hierarchically organized, directories provide a logical way to organize files.

directory mask

A screening of directory names by using matching strings or wildcards.

discontiguous selection

A selection technique that enables the user to select multiple elements that are not necessarily adjacent in a collection.

display control button

A button, which cannot take focus, that indicates whether the object's subobjects are displayed or not.

double−click

1. A mouse technique that involves pressing and releasing a mouse button twice in rapid succession.
2. To press and release a button on a pointing device twice without significantly moving the pointer and within a time specified by the operating environment.
3. An alternative method for selecting a menu item. Use of the SELECT button is assumed.

double−press

See double−click.

Drag

An action choice that initiates drag and drop of the file or object being viewed or of specified elements within the window.

drag

1. To press and hold down a mouse button while moving the mouse and thus the pointer on the screen. Dragging is typically used when selecting menus, moving and resizing windows, and transferring data.
2. A user interaction in which elements or their representations change their position or appearance in conjunction with movement of the pointer.

drag and drop

1. A transfer mechanism whereby data is dragged from a source to a drop site by using mouse motion.
2. A user interaction in which a user drags source elements to a target element on which they are dropped.

drag auto scroll

Scrolling that occurs during a drag.

drag icon

An icon selected to start a drag. A drag icon is composed of three parts: the state indicator, the operation indicator, and the source indicator. *See also* drag pointer.

drag pointer

A pointer displayed during dragging. It is made up of a source indicator, a state indicator, and an operation indicator.

drop

A user action that terminates a drag, identifying the destination of the drag−and−drop interaction as the element under the pointer.

drop site

See drop zone.

drop target

A rectangular graphic that represents the drop zone in a given application.

drop zone

An area of the workspace (including the Trash Can, Printer, and Mailer controls) that accepts a dropped icon. Icons can be dropped on the workspace for quick access.

drop−down combination box

A combination text−list control in which the text−entry field is always visible, but the list box is hidden until the user performs an action to display it.

drop−down list

A combination text−list control that contains a text−display field that is always visible and in which the list box is hidden until the user performs an action to display it.

drop−down list box

A variation of a list box. Only one item in the list is displayed until the user takes an action to display the rest of the list.

Duplicate

An action choice that duplicates selected objects and displays the duplicates in the same selection scope.

Edit

A cascading choice that appears as a menu−bar item. It provides access to other menu items that allow a user to modify the contents of a selection scope.

element

1. A distinguishable part of a user interface. For example, an element can be an object, choice, control, or part of a control.
2. The smallest unit of data in a table or array.
3. In a set, an object, entity, or concept having the properties that define a set. Synonymous with *member*.
4. The component of an array, subrange, enumeration, or set.
5. Any of the bits of a bit string, the octets of an octet string, or the octets by means of which the characters of a character string are represented.

element cursor

A visual cue that highlights an entire element (or its border) to indicate that the user's keyboard operations will interact with that element.

emphasis

Highlighting, color change, or other visual indication of the condition of an element or choice, and the effect of that condition on the user's ability to interact with it. Emphasis can also give the user additional information about the state of an element or choice. In documentation, refer to the specific type of emphasis, such as *''selected emphasis.''*

end−point inclusion policy

A selection policy that specifies whether an area will be enlarged to include the elements at the corners of an identified area.

enlarge−only policy

An adjustment policy in which the current selection region is enlarged, but never shrunk, to include the final point identified by an adjustment technique.

error message

1.An indication that an error has been detected.
2.A type of action message that requires the user's immediate attention. Error messages are used to convey a message about a user error.

etch

A CDE icon design style that uses shadow effects to create the illusion that the icon is embossed into its background. It is typically used on the Front Panel.

exception

1.An event or situation that prevents, or could prevent, an action requested by a user from being completed in a manner that the user would expect. Exceptions frequently occur when an application is unable to interpret a user's input.
2.In programming languages, an abnormal situation that may arise during execution, perhaps causing a deviation from the normal execution sequence, and for which handling facilities exist.
3.An abnormal condition, such as an I/O error, encountered in processing a data set or a file.
4.One of five types of errors that can occur during a floating−point exception. The errors are invalid operation, overflow, underflow, division by zero, and inexact results.
5.An event that is unexpectedly caused by a process while an instruction is executing.

Exit

An action choice that ends the current application and all windows associated with it. This action is equivalent to closing all primary windows of the application and ending the application.

expand button

The control in an expandable window that is used to display and remove a secondary pane. The control can have a set of labels that toggle, such as More/Less or Expand/Contract.

expandable window

A window that allows the user to selectively display advanced or application−specific functionality in a separate portion of the window that is normally not visible when the window is initially displayed.

expert action

An action invoked by double−clicking a tool to perform a function that is a special case of the actions that might be performed with the tool. For example, if the user double−clicks on an eraser tool, the whole viewing area may be erased. Expert actions provide shortcuts for functions available by other methods.

explicit focus

1.A keyboard focus model that sends keyboard events to the window or component that was specified explicitly with a mouse button press or a keyboard event.
2.A focus policy in which the user must explicitly indicate which window or control receives input focus. Also referred to as *''click−to−type''* focus. *See also* implicit focus.

extended selection model

A selection model in which any number of elements can be selected, and in which select mode and normal mode are the default modes.

extension model

A Motif 1.2 term for a method for extending the scope of a multiple selection. *See also* adjustment technique.

feedback

A visible or audible indication that a user action has been accepted by the computer.

File

A cascading choice that appears as a menu−bar item. It provides access to other menu items that allow a user to invoke actions that affect storage, data transfer, and display of the object, file, or data as a whole being viewed in the window.

File Manager

The software application used to manage the files and folders on a system.

file selection dialog

A dialog that allows the user to specify a file name.

Find

1.A dialog choice that initiates a search for objects or data associated with the contents of the current window.
2.A choice within the Find dialog.

find dialog

A dialog that allows the user to specify the criteria to be used for a search.

first−letter cursor navigation

An internal navigation technique in which typing a character navigates and selects the next element in a control whose textual label begins with that character.

focus

1.A state of the system that indicates which component receives keyboard events. A component is said to have the focus if keyboard events are sent to that component.
2.The place to which keyboard input is directed.

See also keyboard focus.

focus emphasis

A type of emphasis that indicates the current location for keyboard input.

focus policy

1.The model by which keyboard focus is moved among components.
2.A means of determining which element or window receives input focus.

See also explicit focus, implicit focus.

focus−only navigation

Navigation in which the user can move focus to a control without interacting with that control, for example, without activating the push button.

folder

An icon used in a graphical desktop environment to represent a directory.

framing rectangle

A highlighted rectangle used to indicate the current scope of a range selection. (The Motif 1.2 term is *marquee*.)

Front Panel

A centrally located window that contains controls for accessing applications and utilities, including the workspace switch. The Front Panel is present on all workspaces.

full toggle

A discontiguous, mouse−based selection toggling technique in which the selection state of each element in the range is toggled. *See also* anchor toggle.

gain

The ratio of distance the pointer moves to the distance the mouse moves.

gauge

A control that displays a value that the user generally cannot change directly. For example, a gauge can be used to display the percent complete in an in−progress message.

grab handle

A small square displayed at each of the corners and midpoints of a selected graphic element.

graphic

A pictorial presentation or image.

graphic element

An element that is displayed as an image. Graphic elements contain a unique format (such as GIF or JPEG) that is recognizable by an application that can display or edit it.

graphical cue

A persistent cue that consists of an augmentation of the graphical image of an existing element or elements of the interface. There are three kinds of graphical cues; emphasis cues, cursors, and pointers. *See also* cue.

graphics cursor

A cursor that identifies an x,y location within a selection scope.

graphics normal mode

The normal mode used in conjunction with selection scopes that use a graphics cursor and in which navigation does not affect the current selection.

group box

A rectangular box drawn around a group of controls to indicate that the controls are related and to provide a heading for the group.

group heading

A textual label that identifies a group of related fields.

group selection technique

A selection technique that identifies a group of elements in a selection scope whose selection state is to be affected.

handle

1.Provided when size, position, or shape can be manipulated. *See also* grab handle.
2.An opaque reference to information, such as a pointer to a data block.

Help

1.An action choice used on push buttons in secondary windows to provide help specific to that window.
2.A cascading choice that appears as a menu−bar item. It provides access to other menu items that contain information related to the use of the application.
3.In a graphical desktop environment, usually a button or menu item that is available to the user to access help information.
4.On a keyboard, the F1 key, which is a common access method to help information.

help

Information provided to the user about the application and how to recover from exceptions.

help window

A secondary window that displays help information.

hidden

A state in which a control is not displayed on the screen until explicitly requested by the user.

hierarchical linear view

A type of linear view of a container in which the contents are displayed in a column, with hierarchical relations between elements identified by indenting.

highlight

To make a graphical element stand out by selecting or choosing it. This action relies on some visual indicator to distinguish that something is selected. *See also* emphasis.

highlight cursor

See active cursor.

horizontal navigation

Keyboard navigation that moves the cursor in a horizontal direction.

hot spot

1. The actual position on the pointer where input device actions occur.
2. In Motif, the actual position on the pointer that identifies the element to which input associated with the pointer is directed.

icon

1. A picture or graphical representation of an object on a display screen to which a user can point with a device (such as a mouse) to select a particular operation or to perform a certain action.
2. A pictorial representation of an object or a selection choice. Icons can represent objects that the user wants to work on or actions that the user wants to perform. A unique icon also represents the application when it is minimized.
3. A small graphical image used to represent a window. Windows can be turned into icons or minimized to save room or to neaten the workspace.
4. An element that represents an object or a window as a graphic, often with an associated text label.

icon color palette

A collection of colors available for use in designing icons. The CDE Icon Editor, for example, has a palette of 22 default colors.

implicit focus

1. A keyboard focus model that sends keyboard events to the window or component that the mouse pointer is on.
2. A focus policy based on pointer movement in which keyboard events are automatically sent to the window and control where the pointer is located. Also referred to as *pointer−driven focus*.

See also explicit focus, pointer focus.

inactive window

A window that is not receiving keyboard input.

Include

A dialog choice used to determine which elements are displayed within a window.

include dialog

A pop−up control that displays or identifies elements or subsets of elements to be displayed, added, or removed from a view.

Index

An action choice that presents an alphabetic listing of help topics for an application.

indirect manipulation

A transfer technique that allows the user to interact with an element through controls and menus.

individual selection technique

A selection technique that identifies an individual element whose selection state is to be affected.

information area

A specific part of a window in which information about the current application task context is displayed. The information area can also contain other task–related messages.

information message

A message that indicates to a user that a condition or an event has occurred, such as *Document has been deleted.*

information snippet

A short informational message, of two lines or less, displayed in the information area of a window. The message describes some immediate aspect of the current interaction, is displayed briefly, and is usually overwritten as the interaction continues.

in–progress message

A message that informs the user about the status of a task, such as copying a set of files.

input device

A mechanism that allows the user to interact with the operating system. Some examples of input devices include the keyboard, mouse, track ball, and joystick.

input focus

See focus, keyboard focus.

insensitive

An unavailable control or choice that cannot receive input or be navigated to.

Insert

1. A dialog choice that leads to a file selection dialog in which the user can select a file or object to be inserted into a specified location or into an object or objects displayed within the window.
2. A cascading choice that may appear as a menu–bar item. It provides access to other choices that enable a user to insert elements into a specified location or into an object or objects displayed within the window.

insertion cursor

See cursor.

insertion point

The position within an editable selection scope at which inserted or pasted data is placed.

instant help

Information displayed for the element under the pointer.

interacted emphasis

Emphasis used to identify the last control that contains an editable selection scope on the user's display with which the user interacted. In explicit mode, it is the last such control that had focus; in implicit mode, it is the last such control to which a key or mouse–button press or release was directed.

interface designer

A person who designs the interface of an application.

interface image

The visual composition of all the elements on the screen.

internal navigation

1.Moving the keyboard focus within a single control.
2.Use of the keyboard to move the active cursor from one element or point to another within a control.

in−use emphasis

A visual cue that indicates that an object is in use, for example, if a view of the object is being displayed in a window.

Keyboard

An action choice that presents information about the application's use of keys, including function keys, shortcut keys, and mnemonics.

keyboard

A device that consists of systematically arranged keys that allow the user to type information, move the cursor, or activate functions assigned to the keys.

keyboard focus

A state in which a window or an element within a window receives keyboard input. *See also* cursor and insertion point.

keyboard navigation

Use of the keyboard to move the active cursor.

keyboard selection mode

A model or technique for navigation and selection that are employed when the keyboard is the input device. The two keyboard selection modes are add mode and normal mode. Each of the various selection techniques may support only one or both of these modes.

keyboard−based selection techniques

Individual and group selection techniques for keyboard users.

label

1.A noneditable text string that is used to describe or name a control. It displays only text information and can appear next to an icon to identify the name of the action or application that the icon represents.
2.A descriptive piece of text used before a text or data entry field to indicate the type of information needed by that field. Some examples are: *static text field, read−only message*, and *read−only text field.*

lazy drag

A transfer operation that allows the user to transfer data from a source to a destination without dragging it and without using a storage mechanism (such as a clipboard).

linear view

A view in which the icons are laid linearly from top to bottom, one row per icon.

Link

An action choice used in the context of a specific transfer mechanism to indicate that the transfer should result in a link.

link

1.In the file system, a connection between an inode and one or more file names associated with it.
2.In data communications, a transmission medium and data link control component that together transmit data between adjacent nodes.
3.In programming, the part of a program that passes control and parameters between separate portions of the computer program.

4.To interconnect items of data or portions of one or more computer programs, such as using a linkage editor to link object programs or using pointers to link data items.
5.A representation of a relation between elements or groups of elements.

Link To

A dialog choice that leads to a window in which the user can specify where links to files or objects should be placed.

list box

1.A component that provides the user with a scrollable list of options from which to choose.
2.A control that contains a list of items that a user can select.

list cascade button

A button that represents a cascading choice used to display a cascaded list.

list item

An element in a list that can be selected.

locale

A language−specific or country−specific environment.

Lower

An action choice that lowers a window in the stacking order.

main control

The single control, if any, in a window in which the bulk of the interaction with the user occurs.

manual stacking order

A model in which windows or elements do not change their stacking order simply as a result of gaining focus. *See also* automatic stacking order.

margin selection technique

A group selection technique in which a user interaction in the margin of a selection region selects a group of related elements in the selection scope.

marquee

See framing rectangle.

Maximize

An action choice that enlarges a window to its largest possible size.

maximize button

1.A control button placed on a window manager window frame and used to initiate the maximize function.
2.A button on the title bar that represents the Maximize choice. The user activates this button to enlarge the window to its largest size.

MBCS

See multibyte character set.

medium scrolling increment

The scrolling increment that is larger than the unit scrolling increment and smaller than a paging increment. *See also* page scrolling increment, unit scrolling increment.

melt effect

A kind of transition effect that occurs when the user drops a drag icon on a valid drop zone. The effect looks like the drag icon melts into the drop zone.

menu

1.A displayed list of items from which a user can make a selection.
2.A pop−up widget that usually allows the user to make a single selection from a constrained set of choices. A menu is usually modal, suspending the application until the user makes a selection or dismisses the menu. When torn off, a menu becomes modeless, allowing the user to interact with the application while the menu remains visible.
3.A control that generally contains a list of choices of any type.

menu bar

1.A menu displayed below the title bar that contains only cascading choices.
2.The part of an application window between the title bar and the work area where menu names are listed.

MENU button

The button on a pointing device that the user presses to view a pop−up menu. For example, mouse button 3 is the default MENU button on a 3−button mouse.

menu cascade button

A button that represents a cascading choice used to display a pull−down menu.

menu item

An element in a menu that represents a choice.

menu pane

A rectangular box that contains menu choices.

menu system

A spring−loaded system whose cascaded controls are all menus.

menu topic

A menu−bar choice.

menu−bar item

A cascading choice that appears on a menu bar. Menu−bar items provide access to menus, which contain additional choices.

menu−bar system

A spring−loaded system that consists of a menu bar and any menus cascaded from it.

message

1.Information from the system that informs the user of a condition that may affect further processing of a current program.
2.Information not requested by the user but displayed in a secondary window by an application in response to an event or exception. There are three types of messages: information, in−progress, and action messages.
3.An error indication or any brief information that a program writes to a standard error or queue.
4.Information sent from one user in a multiuser operating system to another.
5.A general method of communication between two processes.
6.A group of characters and control bit sequences transferred as an entity.

message area

A specific part of a window in which the user can get detailed information or directions about performing a task in the current context.

message window

A message displayed in a secondary window.

metaphor

A user interface representation of an object that suggests a likeness or an analogy with some other object or idea. User interface metaphors may be words, phrases, environments, graphics, or icons that suggest some real–world object or domain.

midpoint technique

See balance–beam policy.

Minimize

An action choice that removes a window and all of the secondary windows associated with that window from the workspace and displays the window icon that represents that window.

minimize button

1. A control button placed on a window manager window frame and used to initiate the minimize function.
2. A button on the title bar that represents the Minimize choice. The user activates this button to remove the window and all secondary windows dependent on the window being minimized and to display the corresponding window icon.

minimized window box

See window icon box.

mnemonic

1. A symbol chosen to help the user remember the significance of the symbol.
2. A single character (frequently the initial character) of a menu selection. When the user presses a character that can be associated with a menu selection, that selection is chosen.
3. A character that the user can type (possibly augmented with `Ctrl` or `Alt`) to move the focus elsewhere in a window or menu and/or to activate or toggle a choice whose label contains and emphasizes that character.
4. The field of an assembler instruction that contains the acronym or abbreviation for a machine instruction. Using mnemonics frees the programmer from having to remember the machine's numeric operator codes.

modal

1. A dialog box that requires a response before the user can interact with other components in an application.
2. A state of a dialog that requires the user to interact with the dialog before interacting with other parts of the application or with other applications. Three modal styles exist: primary application modal, full application modal, and system modal.
3. A state in which the user must complete the operation of the mode before continuing.

See also modeless.

modal secondary window

A window that prevents the user from continuing to work in the existing window from which the secondary window was opened.

mode

1. A method of operation in which the actions that are available to a user are determined by the state of the system.
2. A method of operation (frequently used in software systems based on UNIX) to refer to read, write, execute, or search permissions of a file or directory.

model keyboard

An idealized keyboard that contains the keys and key labels described in this style guide. A model keyboard is used in descriptions because not all keyboards contain the same keys.

modeless

1. A dialog box that does not limit the user's interaction with the rest of an application.
2. A state of a dialog that does not require the user to interact with the dialog before interacting with other parts of the application or with other applications.
3. A state that does not interfere with the user performing any other action.

See also modal.

modeless secondary window

A window that lets the user continue to work in the existing window from which the secondary window was opened.

modifier key

A key used with other keys or with mouse buttons (or other buttons on input devices) to modify the behavior associated with that key or button. The standard modifier keys are `Shift`, `Ctrl`, and `Alt`.

More

1. An action choice that displays additional controls in a separate window.
2. An action choice that, when chosen, causes the associated window to expand and to present the user with more information. If the user has already expanded the window, pressing the More choice again will shrink the window. The More choice, in this case, is labeled <> when the user can expand the window; it is labeled >< when the user can shrink the window.

Mouse

An action choice that presents information about the application's use of the mouse.

mouse

1. A hand–held locator that a user operates by moving it on a flat surface. The user presses mouse buttons to select objects and scroll the display screen.
2. A pointing device commonly used in conjunction with a keyboard in point–and–click, object–oriented user interfaces.
3. A pointing device that has one or more buttons that a user presses to interact with the operating environment.

mouse button

1. A button on a mouse–pointing device. Mouse buttons can be pressed, released, moved, clicked, and double–clicked. On a 3–button mouse, certain default actions are associated with each button, as follows:
   - MB1 – selection
   - MB2 – transfer and adjust
   - MB3 – pop–up activation
2. A mechanism on a mouse–pointing device that the user presses to make choices or directly manipulate elements.

mouse–based selection technique

Individual and group selection techniques for mouse users.

Move

1. An action choice that allows the user to move a window.
2. An action choice that initiates drag and drop, allowing a user to move specified elements within the window.
3. An action choice used in the context of a specific transfer mechanism to indicate that the transfer should result in a move.

move

A mouse technique that involves moving the mouse, which moves the pointer in the interface.

Move To

A dialog choice that leads to a file selection dialog in which the user can specify where files or objects should be moved.

multibyte character set (MBCS)

A set of unique characters that constitute more than what a single byte can represent. Examples of MBCSs are the Chinese and Japanese languages.

multi–click

A mouse technique that involves pressing and releasing a button on a pointing device two or more times without significantly moving the pointer and within a time specified by the operating environment.

multilevel selection technique

A group selection technique used within selection scopes whose elements can be organized hierarchically.

multipage control

A type of control used to show one or more pages at a time from among a larger set of pages, each page containing one or more controls.

multiple selection technique

A selection model in which any number of elements can be selected at a time and in which all selection techniques toggle the identified elements.

multi−press

See multi−click.

navigation (traversal)

1. An action that causes the focus to move to another component.
2. The act of moving the active cursor in response to input.

navigation keys

The keys whose use is specialized for navigation, for example a directional key, paging key, or a `Home` or `End` key possibly augmented with the `Ctrl` or `Shift` key.

New

An action or dialog choice that creates a new file or object of the type used by an application. When chosen from the File menu, the newly created file or object is displayed as well.

New Window

An action or cascading choice that allows a user to display a view of the currently viewed data in a new window.

No

An action choice that indicates a negative response to a question presented in a message.

normal mode

A keyboard−based selection mode in which selection techniques select the element or elements identified by the technique and deselect all other elements in the selection scope.

notebook

A multipage control that shows one page at a time and that supports the use of notebook tabs to switch pages.

notebook tab

A control that switches the page displayed in a notebook.

object

1. A data structure that implements some feature and has an associated set of operations.
2. Any of the complex information constructs created, examined, modified, or destroyed by means of the interface.
3. For threads, an object refers to either a thread, a mutex, or a condition variable.
4. In object−oriented programming, an instantiation of a class. It contains the name, structure, and components defined for the class and contains data.
5. An element that visually represents something that has behavior and contents not revealed solely by the visual representation. Objects are often represented as icons, but could be represented in other ways as well, for example, as list items.

object type

A desktop mechanism used to associate particular data files with the appropriate applications and actions. Object typing defines the criteria for typing the file (such as name or contents), the appearance (for example, an icon used to represent the object), and the behavior of the object (what happens when you double−click on it or drag and drop an item onto it).

OK

An action choice that accepts the information in a window and closes it. If the window contains changed information, those changes are applied before the window is closed.

on (a selectable element)

Describes whether a cursor or pointer identifies a selectable element.

on (a selection)

Describes whether a cursor or pointer identifies a selection.

On Item

An action choice that allows the user to indicate the element for which help is desired.

Open

On the File menu, a dialog choice that leads to a file selection dialog in which a user can select the file or object to open. On the Selected menu, an action choice that opens the selected objects.

operation indicator

The part of the drag pointer displayed during a drag−and−drop operation that indicates whether a drop will result in a move, copy, or link.

option menu

A menu that contains only value and cascading choices and that is displayed from an option menu button or from a cascading choice within an option menu.

option menu button

See option menu cascade button.

option menu cascade button

A button used to display an option menu.

Options

A cascading choice that appears as a menu−bar item. It provides access to other menu items that enable a user to customize an application.

outline view

A view of a container in which the contents are displayed in a column, with hierarchical relations between elements identified by indenting.

Overview

An action choice that provides a brief overview of each action and task that a user can perform within a window.

Pack Icons

An action choice that packs the window icons as close as possible to each other within the visible area of a window icon box.

page scrolling increment

The amount by which an area is scrolled (using a paging operation) equal to the width or height of the scrolled area minus the unit scrolling increment.

page scrolling unit

See page scrolling increment.

paging key

Keys on a keyboard used for paging, for example, page up, page down, page left, page right, and possibly augmented with the `Ctrl` key.

palette

A range of graphically displayed choices, such as colors or a collection of tools, that a user can select from the application's palette area.

palette area

An area within a window that provides a place to store commonly used groups of controls or tools.

pane

1.On a display screen, the inner portion of a window used to present information to the user. A window can consist of one or more panes.
2.A widget that is a child of a paned window. The user adjusts the size of a pane by means of a sash.
3.One of the separate areas in a split window or a paned box.

paned box

A control that can be divided into panes by using split bars.

Paste

An action choice that places the contents of the clipboard into a specified location or into an object or objects displayed within the window.

Paste Link

An action choice that pastes a link to elements previously placed on the clipboard into a specified location or into an object or objects displayed within the window.

Paste Special

A dialog choice that displays a dialog through which a user can choose the format in which elements previously placed on the clipboard are pasted into a specified location or into an object or objects displayed within the window.

Pause

An action choice that temporarily suspends a task without ending it.

pending delete

1.A state of a text component in which some user actions cause the current selection to be deleted.
2.A mode of a selection scope in which an insertion within a selected region replaces the selected item.

persistent

A mode of a spring–loaded control in which the control is displayed and in which the keyboard can be used to navigate through it.

persistent cue

A cue that directs the user's attention to a part of the screen or user interface, indicates a particular state of an object, or alerts the user about a potentially serious situation.

persistent selection

A selection whose state is unaffected by making a selection in another selection scope.

point technique

An individual selection technique in which a single element or point is identified.

pointer

1. The graphical image that appears on the workspace and represents the current location of a mouse or other pointing device.
2. A visual cue, usually in the shape of an arrow, that a user can move with a pointing device. Users place the pointer over elements they want to work with.
3. In computer graphics, a manually operated functional unit used to specify an addressable point. A pointer may be used to conduct interactive graphic operations (such as selection of one member of a predetermined set of display elements) or to indicate a position on a display space while generating coordinate data.
4. The device attached to the cursor and tracked on the screen.
5. A physical or symbolic identifier of a unique target.
6. A variable that holds the address of a data object.

pointing device

A device such as a mouse, trackball, or graphics tablet that allows the user to move a pointer about on the workspace and to point to graphical objects on the screen.

pop−up control

A control temporarily displayed as a result of a context−sensitive user interaction (other than from a cascading choice) that appears at or near the location where it was requested.

pop−up menu

A menu that is a spring−loaded, pop−up control and that contains context−sensitive choices.

posted

The state of a menu, list box, or combination box where it remains visible when a mouse button is not being held down. *See also* spring loaded.

preedit area

An area for entering alphabetic text that is translated to language−specific characters.

prefix completion

An internal navigation technique used in a combination text−list control in which entering a printing character immediately moves the cursor in a text−display field to the next item whose textual label begins with the prefix.

prefix navigation

An internal navigation technique used in a combination text−list control in which entering a printing character immediately moves the cursor in the associated list to the next item whose textual label begins with a prefix.

press

A mouse technique that involves pressing and holding down a key or mouse button or other button on an input device.

press and move

A mouse technique that involves pressing the mouse button without releasing it and then moving the position of the pointer, such as moving an element (like an icon) across the screen.

Primary Copy

An action choice that copies the contents of the primary selection to a specified location or to an object or objects displayed within the window.

primary key mapping

The standard assignment of keys and key sequences to functions in Motif and CDE.

Primary Link

An action choice that places a link to the elements contained in the primary selection in a specified location or in an object or objects displayed within the window.

Primary Move

An action choice that moves the contents of the primary selection to a specified location or to an object or objects displayed within the window.

primary pane

The part of an expandable window that reveals the window's core functions and is displayed by default. *See also* expandable window, secondary pane.

primary selection

1.The principal selection used to transfer data from one client to another or to another window of the same client.
2.A selection that is automatically deselected when a new primary selection is made elsewhere.

primary transfer

A transfer mechanism where the primary selection is transferred to the destination target.

primary window

1.The window in which the main interaction between the user and an object or application takes place.
2.A top–level window of an application. Primary windows can be minimized.

Print

A dialog choice that prepares and arranges data to be printed on a designated printer.

Print Setup

An action choice that opens a window in which the user can set the parameters for printing.

program

1.A sequence of instructions that a computer can interpret and execute.
2.A sequence of instructions suitable for processing by a computer. Processing can include the use of an assembler, compiler, interpreter, or translator to prepare the program for execution and to execute it.
3.A file that contains a set of instructions that conform to a particular programming language syntax.
4.In programming languages, a logical assembly of one or more interrelated modules.
5.To design, write, and test computer programs.

See also application.

Properties

An action choice that displays a window in which the user can display and set properties or characteristics of an object.

property dialog

A dialog in which the user can display and set properties or characteristics of a file object or other element.

pull–down menu

A cascaded menu that is displayed from a menu cascade button or a menu bar.

push button

A control that resembles a button and that represents an action or dialog choice.

question message

A type of action message that does not require the user's immediate attention. Question messages are used to get a response to a question.

quick transfer

1. A transfer mechanism where selected data is immediately transferred to the destination.
2. A data transfer technique used to copy, move, or link data to the insertion point of the control that has interacted emphasis.

radio box

A group of mutually exclusive radio buttons that are grouped together.

radio button

A control used to set values that are mutually exclusive.

range click technique

A range technique in which two separate mouse or keyboard operations are used to indicate the endpoints of the range.

range selection model

A selection model in which any number of contiguous elements in a range can be selected and in which select mode and normal mode are the default modes.

range swipe technique

A range technique in which the endpoints of the range are indicated by moving the mouse or cursor from one endpoint to the other.

range technique

A group selection technique in which the user selects elements within an area by indicating the opposite endpoints of a range (such as in a list or in text).

read−only text

Displayed text that a user can read but not directly select or change, such as the calculation of a total. Read−only text is displayed in a read−only text field. *See also* label.

read−only text field

A control that displays text but does not allow the user to change or select it directly. *See also* text−display field.

ready emphasis

When the SELECT button is pressed, a visual cue for an element used to indicate that the element will be activated or that its value or selection state will be changed when the SELECT button is released.

Redo

An action choice that reverses the effect of an undo operation, returning the application state to what it was before the undo was performed.

Reference

An action choice that provides access to reference pages for the application.

Refresh

An action choice that updates the window or screen to reflect the current state of the underlying data.

release

A mouse technique that involves releasing a mouse button after pressing it. Releasing the mouse button performs an action initiated by pressing it, such as activating a push button.

Remote Host

An action choice that displays a message that contains information about the host on which the application is running.

Repeat

An action choice that repeats the last action invoked by the user.

Reselect

An action choice that reselects the previously selected elements within a selection scope.

reselect policy

A selection policy that adjusts the previous selection relative to its anchor.

Reset

An action choice that resets the values displayed in a dialog or property window to the values they had when the window was displayed or when the values were last saved as defaults, whichever is most recent.

Reset to Defaults

An action choice that resets the values displayed in a dialog or property window to the values they had when default values were last saved.

Reset to Factory

An action choice that resets the values displayed in a dialog or property window to the values they had at the time the application was installed or delivered.

resize corner

A Motif 1.2 term for a window border control that enables the user to alter the size of windows by dragging the border corner graphic toward or away from the center of the window. *See also* size border.

resize handle

See resize corner.

Restore

An action choice that returns a window to the size it was and the position it was in before the user minimized or maximized the window.

restore button

A button on the title bar that represents the Restore choice. The user activates this button to restore the window to its normal size.

Resume

An action choice that resumes a task that the user paused.

Retry

An action choice that attempts to complete an interrupted task.

Revert

An action choice that resets a view of an object or file to the state it was in when the object or file was last opened or last changed, whichever is more recent.

sash

1.A control with which the user changes the sizes of the panes in a paned window.
2.A box on a split bar through which the user can directly manipulate the split bar to change the sizes of associated panes.

Save

An action choice that stores the application's data into a previously determined file or object.

Save As

A dialog choice that leads to a file selection dialog in which the user can specify the file or object in which application data is to be stored.

Save as Defaults

An action choice that saves the values displayed in a dialog or property window as defaults to be used when the application subsequently displays the same (or similar) window to the same user.

SBCS

See single byte character set.

scale

See slider.

scope of selection

An area that contains related elements that the user can select.

screen

1.The physical surface of a workstation display upon which device information is shown to the user.
2.An abstraction that represents a single bitmapped output device on a display.

scroll

1.To move text vertically or horizontally in order to view information that is outside the display or pane boundaries.
2.To incrementally shift the view of the elements being displayed through the control or window.

scroll bar

1.A graphical device used to change a user's viewpoint of a list or data file. A scroll bar consists of a slider, scroll area, and scroll arrows. A user changes the view by sliding the slider up or down in the scroll area or by pressing one of the scroll arrows. This causes the view to scroll up or down in the window adjacent to the scroll bar.
2.A user–interface element associated with an area that can be scrolled. The scroll bar indicates to a user that more information is available or that it can be added in a horizontal or vertical direction and scrolled into view.

scroll box

The part of a scroll bar that indicates the position of the visible information relative to the total amount of information available in a window. A user manipulates the location of a scroll box with a pointing device to see information that is not currently visible.

scroll track

Within a scroll bar, the rectangular region that contains the scroll box. The user moves the scroll box within the scroll track.

secondary key mapping

A key assignment to a function made in addition to the standard Motif/CDE functional key assignment. For example, `Ctrl Z` is mapped to the Undo function in Motif/CDE. An optional secondary key mapping of `Alt Backspace` may be assigned to the Undo function to assist users of other systems.

secondary pane

The part of an expandable window that reveals additional function. *See also* expandable window, primary pane.

secondary window

1.A child window of a primary window.

2.A window dependent on another window, either primary or secondary, that is used to supplement the interaction in that window.

select

1.To choose an object to be acted upon or an action to be performed.
2.To explicitly identify one or more elements with which to interact.

Select All

An action choice that causes all of the elements in a scope of selection to be selected.

SELECT button

The button on a pointing device that the user presses to make a selection. On a mouse, it is always bound to logical mouse button 1 (MB1).

select mode

A selection mode in which the identified elements are selected.

Select Pasted

An action choice that selects the element last transferred into a selection scope.

Selected

A cascading choice that appears as a menu−bar item. It provides access to choices that apply to the selected objects in the current view.

selected emphasis

A visible cue that indicates that an element is selected.

selection

The process of selecting an element or group of elements.

selection auto scroll

Scrolling that occurs during selection.

selection box

A combination text−list control in which both the text−entry field and the list box are visible at all times.

selection cursor

See active cursor.

selection dialog

A dialog that contains a selection box, possibly with other controls added to it.

selection mode

A mode that determines whether selection techniques select or toggle the selection state of identified elements.

selection model

A description of how selection works in a selection scope, including the selection techniques available in the scope, the available and default selection modes, and the selection policies used in the scope.

selection policy

The characterization or constraint on selections in a scope, for example, whether or not more than one element at a time can be selected.

selection scope

The limit on how a user can select elements. A selection model determines how selection works in a selection scope.

selection technique

The method by which the user identifies elements to be selected or deselected.

separator

1.A punctuation character that separates parts of a command or file or that delimits character strings.
2.A boundary, such as blank space, a line, or color change, that provides a visual distinction between two adjacent areas.

Session Manager

A software application that controls saving sessions, restoring sessions, screen locking and unlocking, and the use of screen savers. When a session is saved, the state of the desktop environment (location of icons, placement of open windows, state of applications, current color palette, and so on) is preserved so that it can be restarted at the next login.

shortcut key

A key or combination of keys assigned to a menu item that activates that menu item, even if the associated menu is not currently displayed, such as `Alt F4` for Close.

shortcut key

A key or combination of keys assigned to a menu item that activates that menu item, even if the associated menu is not currently displayed, such as `Alt F4` for Close.

single byte character set

A set of no more than 256 unique characters. The English alphabet is contained in a single byte character set. ASCII is an example.

Size

An action choice that allows a user to change the size of a window or of specified elements.

size border

A window border whose corners and edges can be used to size the window. *See also* window border.

slider

1.One of the graphical components of a scroll bar or scale. The slider is the object that is dragged along the scroll area to cause a change.
2.A control that represents a value. When a slider is used to display a particular value amid a range of possible values, it typically shows a scale marked in equal units.

slider arm

The part of a slider that shows the current value of the slider and allows the user to change the value.

slider track

The part of the slider in which the slider arm sits.

snap−back effect

An effect that occurs when the user drops a drag icon on an area that is not a valid drop zone or because the data transfer failed. When this happens, the drag icon snaps back to the source application.

Sort

A cascading or dialog choice used to arrange the elements presented in a view in a specified order.

sort dialog

A dialog that allows the user to specify the criteria used for sorting.

source

1.The object that is selected, dragged, and dropped in a drag−and−drop operation.
2.To initiate a drag−and−drop operation with a particular object or location. Do not use as a verb. Use *initiate* or *start*.
3.The object that is selected, dragged, and dropped in a drag−and−drop operation; or the object that is selected and moved, copied, cut, pasted, or linked in a data transfer or file−manipulation operation.

source element

An element that is the source of a data transfer operation.

source emphasis

A visual cue that indicates the element from which a user made a request or initiated a transfer operation.

source indicator

The part of the drag pointer displayed during a drag−and−drop operation that describes the source.

spatial view

A view of a container in which the user can place an object at an x,y location subject to layout constraints.

spin box

A control used to display a sequenced ring of related but mutually exclusive choices, such as days of the week. The accepted value is displayed in a text element; optionally this is an editable text−entry field into which the user can type a valid choice. A spin box may have multiple text elements. If so, the element of the field affected by the user's action in the spin box is the element or field that displays the active cursor.

spin button

A spin box that consists of a single field.

split bar

An element that separates panes in a window or a paned box. It may provide a sash that allows the user to change the size of the panes. *See also* separator.

split window

A window that is split into multiple panes by one or more split bars and that allows an application to display views in each pane.

spring−loaded choice

A modal choice used to display a spring−loaded control. Currently there are two types of spring−loading choices: menuing choices and listing choices.

spring−loaded control

A kind of cascaded or pop−up control that is removed when the user makes a choice within the control.

spring−loaded menu

A menu that is a spring−loaded control. It is either a cascaded menu or a pop−up menu.

spring−loaded system

A system that consists of a base control and its spring−loaded descendants.

spring−sensitive menu

A menu that appears as the result of pressing an appropriate mouse button or when an appropriate mouse button is pressed

in a menu that is already displayed. In a spring−sensitive menu, an active cursor follows the pointer as the user moves the pointer through the menu.

spring−sensitive mode

A mode of a displayed control in which moving the pointer within the control over a cascading choice displays its associated cascaded control.

stacking order

The order in which windows of an interface or elements within a window or control are stacked one on top of the other. *See also* automatic stacking order, manual stacking order.

standard normal mode

The mode used in conjunction with selection scopes that use an element cursor in which navigation deselects all elements except the element (if any) to which the cursor moves.

state indicator

1. The part of the drag pointer displayed during a drag−and−drop operation that indicates whether or not the pointer is at a place where a drop is likely to result in a successful operation.
2. The change of a cursor to an I−beam to indicate an available data/text−entry region.

static text

A control that displays text but does not allow the user to change or select it directly. *See also* text−display field.

status area

1. The part of the window that identifies the input style (phonetic, numeric, stroke and radial, and so on) and the current status of an input method interaction.
2. A specific part of a window used to display information about the state of the current application task. *See also* information area and message area.

Stop

An action choice that ends a task and removes the message window that refers to it.

Style Manager

The software application used to customize some of the visual elements and system device behaviors of the workspace environment, including colors and fonts, as well as keyboard, mouse, window, and session startup behaviors.

submenu

1. A cascading menu.
2. A menu displayed from a cascading choice in a cascaded menu, a tear−off menu, or a pop−up menu.

subpanel

An extension of the Front Panel that slides up to provide access to additional elements. Subpanels usually contain groups of related elements.

swipe

The process of selecting an element or elements by moving the mouse over the elements while holding the SELECT button down.

system modal

A state of a window that prevents user interaction with any other control outside of that window. *See also* application modal.

tab group

1. A control or group of controls that the user can navigate to. There are two categories: tab groups that contain a group of other controls (for example, a radio box that contains a group of radio buttons) and tab groups that consist of a single control (for example, a text field).

2.A control or group of controls to which the user can navigate by pressing the Tab key.

tab group navigation

Navigation among tab groups within a window.

Table of Contents

An action choice that displays a table of contents of the help information available.

target element

The element that is the target of a data transfer operation.

target emphasis

A visual cue that indicates the element that will receive the results of a transfer operation.

target well

A control used as the destination of a transfer operation in order to transfer elements into an associated selection scope.

Tasks

An action choice that enables a user to determine how to perform specific tasks.

tear−off choice

A kind of action choice used to tear off a control.

tear−off menu

A copy of a pull−down, pop−up, or cascaded menu that the user has "torn off" and placed in a separate window.

technique initiation policy

In a selection scope that supports both touch and area techniques, specifies which technique is initiated in which circumstances.

text cursor

1.A cursor that indicates where to type a character. The text cursor is controlled by the keyboard.
2.A cursor that identifies a point between adjacent characters in text within a selection scope.

text field

A control in which characters can be displayed and selected. All text fields are either text−display fields or text−entry fields. Text fields with keyboard focus have a blinking text insertion cursor

text normal mode

A mode that can be used in conjunction with selection scopes that use text cursors and in which navigation deselects all elements but that also supports the range click technique.

text−display field

A control in which noneditable alphanumeric text can be displayed and selected.

text−entry field

A control into which a user can type or place alphanumeric text.

textual cue

A cue that consists of a word or words that describe the current situation.

title bar

The area across the top of a window that consists of the window menu button, the title area, and the window's control buttons. *See also* window title.

toggle

1.A switching device such as a toggle key on a keyboard.
2.Pertaining to any device having two stable states.
3.To switch between two modes on a computer or network.
4.To interact with the representation of a choice in order to set or unset it.
5.To switch the selection state of an element or group of elements.

toggle choice

A choice that is selected by pressing the SELECT button (or keyboard) to cycle among a limited set of values. Toggle choices include radio buttons and check boxes.

toggle mode

A selection mode in which the identified elements have their selection state toggled.

toggle removal policy

A selection policy that determines the selection state of elements removed from a selection region that was adjusted using toggling.

toggling policy

A selection policy that determines exactly how the selection states of elements identified by a selection technique are toggled when they are not all either selected or all deselected.

tool

A kind of value choice, usually labeled by a graphic, that establishes a mode when set.

tool bar

See palette area.

tool choice

A choice in a value set that, when set, places the user in a mode that limits or specializes the user's interaction with the application.

tool palette

A palette area that contains a value set made up of tools.

touch swipe technique

A touch technique in which an element is added to the current selection by moving the pointer over it.

touch technique

A selection technique in which *''touched''* elements are added one at a time to the current selection.

TRANSFER button

The assigned button on a pointing device that is used for data transfer operations. On a 3–button mouse it is bound by default to MB2, but it may also be integrated with selection and bound to MB1.

transfer icon

A pointer icon that is used to represent the object or data transferred in a move, copy, or link operation.

transfer operation

A move, copy, or link operation.

transfer technique

A technique for performing data transfer operations such as moving, copying, or linking. *See also* clipboard transfer, drag and drop, primary transfer, quick transfer.

transfer well

A control used as the target of a transfer operation to transfer elements into an associated selection scope.

trash can

An object to which the user drags and drops another object to be deleted. The object to be deleted stays in the trash can until the user deletes the object or until the operating system automatically removes it.

Tutorial

An action choice that gives a user access to online educational information.

unavailable choice

A choice that is not available.

unavailable emphasis

A visual cue that indicates that a choice or control is not available. *See also* insensitive.

Undo

An action choice that reverses the effect of the last action invoked by the user, returning the application state to what it was before the action was performed.

unit scrolling increment

The smallest amount a scrolled area can be scrolled when using an associated scroll bar.

urgent message

A message that is essential for the user to see immediately.

user interface

1.The area where the user and an application come together to interact.
2.The ensemble of hardware and software that lets a user and a computer communicate.

user model

A user interface technique.

Using Help

An action choice that gives a user information about how the help system works.

value choice

A type of choice that allows the user to indicate whether a value is set, unset, or indeterminate. Value choices can be presented, for example, in list boxes, radio buttons, check boxes, value sets, and menus.

value set

A control that contains a mutually exclusive set of choices, each of which is usually labeled graphically.

vertical navigation

Keyboard navigation that moves the cursor in a vertical direction.

vertical writing support

Support for languages that use ideographic scripts, such as Japanese or Chinese. These scripts are written vertically, from right to left.

View

A cascading choice that appears as a menu−bar item. The View choice provides access to menu items that allow a user to choose how data is presented, how much information is presented, in what order it is presented, and other choices related to the presentation within a view.

view

A presentation of information by an application in the viewing area of a window; or, the representation of an object within a window.

viewing area

One or more controls that together present a coherent view of data or information in the window.

virtual button

A model used by CDE and Motif that defines mouse−button functions independent of the actual number of buttons on the mouse. The virtual buttons are SELECT, TRANSFER, ADJUST, and MENU.

visible signal

A transient but prominent change in the visual appearance of an element of the interface, usually generated by the operating environment, for use as a visible warning signal.

visual cue

A change in the appearance of an application's elements. There are two types of visual cues: graphical cues and textual cues.

visual priority

The priority of elements, or what elements on the screen appear most important to the user.

warning message

A kind of action message that indicates that although an undesirable condition might occur, the user can allow the process to continue.

warning signal

A transient cue that alerts the user about a minor, nonfatal error or problem with some interaction. For instance, it could draw the user's attention to, or produce feedback about, an event or the state of the user's task or of the environment. A warning signal may be presented using an audible signal (such as a beep), a visible signal (such as a flashing of the screen), or both.

warp the pointer

An action taken by an application in which the pointer is moved independently of pointing−device movements the user makes.

window

1. A division of a screen in which one of several programs executing concurrently can display information.
2. A rectangular area on the display screen where application programs typically have one main window available to initiate interaction. A user can open secondary windows and dialog boxes from this main window.
3. A data structure that represents all or part of the display screen. Visually, a window is represented as a subarea of the display screen.
4. An area with visible boundaries that can be defined so that the user can view interactions with an application.
5. In curses, the internal representation of what a portion of the display may look like at some point in time. Windows can be any size, from the entire display screen to a single character.

6.In data communications, the number of data packets that can be sent across a logical channel before waiting for authorization to send another data packet. The window is the main mechanism of pacing, or flow control, of packets. In X.25 communications, the number of packets that can be outstanding without acknowledgment.

window border

The outer area of the window frame. It is called a size border if it supports sizing the window.

window cluster

See window family.

window family

A group of windows that consists of a primary window and all secondary windows that are directly or indirectly dependent on the primary window.

window frame

1.The visible part of a window that surrounds a software application.
2.The outer area of a window that supports window−management functions. The window frame includes the title bar, minimize button, maximize button, window menu button, and sizing borders.

window icon

A graphical element that represents a minimized window.

window icon box

A window that contains window icons.

Window Manager

1.The program that controls the size, placement, and operation of windows in the workspace. The Window Manager includes the functional window frames that surround each window object and may include a separate menu for the workspace.
2.The program that provides the user with the capability to manipulate windows in the workspace, for example, opening, resizing, moving, and closing windows.

window menu

A menu displayed from a window menu button that consists of choices that affect the window.

window menu button

A menu cascade button on the title bar from which the window menu is displayed.

window navigation

1.Moving the keyboard focus among windows.
2.Navigation among windows and workspace icons.

window title

The area on a title bar that contains a short description of the contents of the window or the name of the object being viewed in the window.

workspace

1.The area on which the windows of a user's environment appear. The workspace is sometimes called the desk, desktop, or root window.
2.The current screen display, the icons and windows it contains, and the unoccupied screen area where the user can place icons.
3.An area that holds elements that make up the user interface.

workspace background

The portion of the display not covered by windows or icons.

Workspace Manager

The software application that controls the size, placement, and operation of windows within multiple workspaces.

workspace menu

A pop–up menu available in the background of the workspace.

workspace object

An object that resides in a workspace rather than inside a window. Workspace objects include windows, icons, and objects that have been dragged in from the File Manager or Application Manager and dropped on the workspace background.

workspace switch

A control that enables the user to select a given workspace from a palette of available workspaces.

workspace switch area

The rectangular area on the center of the Front Panel that contains the palette of available workspace switches, the lock control, the Exit button, and the busy light.

Yes

An action choice that indicates a positive response to a question presented in a message.