Data General

Customer Documentation

OSF/Motif™ Style Guide

AViiON™
PRODUCT LINE

OSF/Motif™ Style Guide

069-100323-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 069-100323

Printed in the United States of America
Revision 00, November 1989

NOTICE

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, MA 01580

OSF/Motif Style Guide

069-100323-00
069-100318-00 (Japan only)

Effective with OSF/Motif Revision 1.0

Contents

Preface

Audience ..... Preface-ix
Contents ..... Preface-ix
Typographical Conventions ..... Preface-ix

Chapter 1 Introduction

What is Motif? 1-1
Empowering the User 1-2
- Consistency 1-2
- Direct Manipulation 1-3
- Flexibility 1-3
- Explicit Destruction 1-3

The Elements of the OSF/Motif Graphical Environment 1-4
- The Input Selection Model 1-4
- The Window Manager 1-4
- Application Programs 1-4

Chapter 2 Motif Fundamentals

The Input Focus Model 2-1
- Explicit Focus 2-1
- Implicit Focus 2-2

The Input Device Model 2-2
- Dual Access via Mouse and Keyboard 2-2
- Keyboard Conventions 2-2
  - Cursor Shapes 2-3
- Pointing Devices 2-4
  - Mouse Buttons 2-5
  - The Pointer 2-6
  - Pointer Shapes 2-6

The Navigation Model 2-7
The Object-Action Selection Model 2-7
- Single Selection Selects One Object Only 2-8
  - Single Selection with a Mouse 2-9
  - Single Selection with a Keyboard 2-9
- Range Selection Selects Contiguous Objects 2-9
  - Range Selection Using a Mouse 2-10
  - Range Selection Using a Keyboard 2-11
- Selecting Additional Non-contiguous Objects 2-12
  - Making Additional Non-contiguous Selection Using the Mouse 2-12
  - Making an Additional Non-contiguous Selection Using the Keyboard 2-13
- Deselecting an Object 2-13
- Selecting the Default Action 2-13
- Auto-selection 2-14

Chapter 3 How the Window Manager Works

Types of Windows 3-1
Primary Windows 3-2
Secondary Windows 3-3
Window Anatomy 3-3
Window Menu and Window Menu Button 3-4
Window Control Buttons 3-5
Minimize Button 3-6
Maximize Button 3-6
Title Area 3-6
Resize Border 3-7
Client Area 3-7
The Icon Box 3-8

Chapter 4 Designing OSF/Motif Clients

Client Areas 4-1
Client Sub-areas 4-1
Menu bar 4-1
Control panels 4-2
Other Client Sub-areas 4-2
Message Area 4-3
Command Line 4-3
Client Controls 4-3
Window panes 4-3
Scroll bars 4-3
Other Controls 4-4
Other Client Areas 4-4
Menus 4-5
Dialog Boxes 4-5
Grouping Similar Controls 4-5
Designing Grouped Controls with Push Buttons 4-6
Combining Controls 4-6
Presenting Multiple Controls 4-7
Pull-Down Menus 4-7
Pop-Up Menus 4-7
Dialog Boxes 4-7
Control Panels 4-8
Beginning a Client Area 4-8
Arrange Controls in Natural Scanning Order 4-8
Arrange Controls in the Sequence People Use Them 4-8
Adjusting the Client Area 4-8
Choosing the Appropriate Control 4-8
Deciding Between a Pop-Up Menu and Push Buttons 4-9
Deciding Between Dialog Boxes and Menus 4-9
Aligning Columns of Controls 4-9
Using Defaults 4-9

Chapter 5 Providing Controls: Buttons, Boxes, and Valuators

Types of Buttons 5-1
Push Buttons 5-1
Radio Buttons 5-1
Check Buttons 5-2
Types of Boxes 5-3
List Boxes 5-3
Entry Boxes 5-4
Pending Delete 5-4
Text Cursor Shapes 5-5
Pre-formatting Entry Areas 5-5
Types of Valuators 5-5
Using a Scale 5-6
Using a Scroll Bar 5-6
Scroll Bar Components 5-6
Operating Scroll Bars 5-7
Automatic Scrolling 5-8
Slider Size 5-8
Application Extras 5-8
Combining Controls 5-9

Chapter 6 Designing Menus

What Types of Menus There Are 6-1
Pull-down Menus Are Always Available 6-1
Option Menus 6-2
Pop-Up Menus Save Space 6-3
Cascading Menus Provide Further Selection Detail 6-4
What Components Make Up Menus 6-4
Menus Have Titles 6-5
Menus Have Selections 6-5
Menus Have Mnemonics 6-6
Menus Have Keyboard Accelerators 6-7
How Users Operate Menus 6-7
Displaying Menus 6-7
Browsing the Menu Bar 6-7
Choosing Menu Selections 6-8
Avoiding Menu Selection 6-8
The Standard OSF/Motif Menus 6-8
A Look at the File Menu 6-9
A Look at the Edit Menu 6-11
A Look at a View Menu 6-12
A Look at an Options Menu 6-13
A Look at the Help Menu 6-14
How to Design Motif Menus 6-15
Group Like Menu Selections Together 6-16
List Selections in Order of Frequency 6-16
Keep Menu Structures Simple 6-16
Provide Accelerators and Mnemonics 6-16
Control Availability of Menu Selections 6-17

Consider Use of Graphic Images 6-17
Keep Menu Selections Stable 6-17
Allow Users to Customize Menus 6-18

Chapter 7 Designing Dialog Boxes

The Characteristics of Dialog Boxes 7-1
The Purpose of Dialog Boxes 7-1
Ending a Dialog 7-1
Making Controls Unavailable 7-2
Dialog Box Actions 7-2
The Anatomy of a Dialog Box 7-3
Arranging Push Buttons in Dialog Boxes 7-4
Default Push Buttons 7-4
Message Dialog Box Types 7-5
Information 7-5
Progress 7-5
Question 7-6
Warning 7-6
Action 7-7
Common Dialog Boxes 7-7
File Selection 7-8
Command 7-8
Selection 7-9
Starting a Dialog 7-10
Navigating Through a Dialog Box 7-10
Determining Dialog Box Location and Size 7-11
Location 7-11
Size 7-11

Chapter 8 Providing Online Help

Types of Help 8-1
Help On Context 8-1
Help On Window 8-1
Help On Keys 8-2
Help Index 8-2
Help on Help 8-2
Help Tutorial 8-2
Help On Version 8-2
Help Windows 8-2

Chapter 9 Designing International Software

Collating Sequences 9-1
Country-Specific Data Formats 9-1
Thousands Separators 9-2
Decimal Separators 9-2
Positive and Negative Values 9-2
Currency 9-2
Dates 9-2

Time Formats 9-3
Time Zones 9-3
Telephone Numbers 9-3
Icons and Pointer Shapes 9-3
Scanning Direction 9-3
Translating Screen Text 9-3
Messages 9-4

Appendix A Default Keyboard and Mouse Bindings

Glossary

Preface

The *OSF/Motif Style Guide* provides a framework of behavior specifications to guide application developers, widget developers, and window manager developers in the design of new products consistent with Presentation Manager and the OSF/Motif user interface.

The *Style Guide* establishes a consistent behavior among new products by drawing out the common elements from a variety of current behavioral models. The *Style Guide* anticipates the evolution of graphical user interfaces as new technology becomes available and as the use of the Motif user interface spreads. Research and the passage of time will enable additional behaviorial elements to be added to the style guide.

The details of coding an application program, widget, or window manager are explained in the *OSF/Motif Programmer's Reference* and the *OSF/Motif Programmer's Guide.*

Clients using the X Window System must follow the guidelines set forth in the *Inter-Client Communication Conventions Manual* (ICCCM).

Audience

This document is written for:

- Application designers
- Widget designers
- Window manager designers

Contents

This document is organized into nine chapters and one appendix.

- *Chapter 1* discusses OSF/Motif design and its principles.
- *Chapter 2* discusses the four basic models of OSF/Motif behavior: input focus, input devices, navigation, and object-action selection.
- *Chapter 3* discusses the functional elements of an OSF/Motif window manager.
- *Chapter 4* discuses the client area and its organization.
- *Chapter 5* discusses control types and how to use them.
- *Chapter 6* discusses different types of menus, their components, and how to use them.
- *Chapter 7* discusses the characteristics of dialog boxes, standard dialog box actions, the graphical control elements of dialog boxes, and how to design OSF/Motif dialog boxes.
- *Chapter 8* discusses issues relating to user help.
- *Chapter 9* discusses the design of applications designed for international markets.
- *Appendix A* provides default keyboard and mouse bindings.

Typographical Conventions

This volume uses the following typographical conventions:

- **Boldfaced** strings represent literals; type them exactly as they appear.
- *Italicized* strings represent variables (for example, function or macro arguments).
- Ellipses (...) indicate that additional arguments are optional.
- <*Key*> represents a key on the keyboard.

Additionally, throughout the style guide certain words are used in a particular sense when referring to what constitutes consistent behavior. For example, "should," "may," and "can" denote a recommendation; "must" denotes a prescription.

Another convention used throughout the style guide regards pointing devices. For the sake of simplicity, the style guide uses "mouse" to refer to any and all pointing devices including but not limited to track balls, graphics tablets, joy sticks, and special sets of graphics navigation keys.

Chapter 1
Introduction

This style guide was written for three audiences:

Application developers	Those who write application programs that use an interface consistent with the OSF/Motif user interface.
Widget developers	Those who create new widgets or modify existing widgets to add substantial new functionality to a widget set consistent with the OSF/Motif user interface.
Window manager developers	Those who create new window managers or modify the functionality of existing window managers to be consistent with the OSF/Motif user interface.

Application developers, widget programmers, and window manager developers who design and implement code according to this *Style Guide* ensure that their new software product's behavior is consistent with the Presentation Manager and OSF/Motif user interface. Adherence to the principles in this *Style Guide* adds value to your new product, and eliminates the need for you to create multiple machine-dependent versions.

What is Motif?

OSF/Motif, OSF's first offering, is a graphical user interface combining a toolkit, presentation description language, window manager, and style guide.

Toolkit	The OSF/Motif toolkit is a rich and varied collection of widgets and gadgets for building OSF/Motif applications. The toolkit provides a standard graphical interface upon which the window manager is based. The behavior of the toolkit conforms to Microsoft's Presentation Manager (PM), ensuring an easy transition between PC and workstation environments. Toolkit widgets provide a 3-D reference appearance that gives users real-world, visual cues to the effects of their actions.
Presentation	User Interface Language (UIL), the OSF/Motif presentation description language allows application developers and interface designers to create simple text files which describe the visual properties and initial states of interface components. Changes to components are made in the text file, eliminating the need to change application code when tuning an interface.
Window	The window manager works with the toolkit to manage the operation of windows on the screen. The window manager provides functions for moving and resizing windows, reducing windows to icons, restoring windows from icons, and arranging windows on the workspace. The OSF/Motif window manager provides compatibility with PM behavior. An additional OSF/Motif window manager feature is the icon box. The icon box contains icons for all windows operating under the window manager.
Style	This style guide describes the standard for window manager and toolkit behavior. It is a guide to usage, providing application writers with guidelines for using toolkit widgets, widget writers with guidelines for designing new widgets, and window manager writers with guidelines for designing new or customized window managers.

Together, these four elements provide a standard of user interface behavior for applications.

Empowering the User

A major software design goal should be to *empower the users who use your software*. To do this, give users both the tools to get the job done and an easily achieved control over those tools.

Users are in control of your product when you design it with these principles in mind:

- Consistency.
- Direct manipulation.
- Flexibility.
- Explicit destruction.

Because of the particular nature of your client application, widget, window manager, or in responce to customers' needs, you may not be able to apply all of these principles all of the time.

Consistency

Above all else, an application must be consistent. It must be consistent within itself; but to be truly successful in the marketplace, it must be consistent with other applications that share the same environment.

Consistency means the following:

- Similar controls operate similarly and have similar uses. For example, since pull-down, pop-up, and cascading menus are similar controls, their operation and use should be similar.
- The same action should always have the same result. For example, click the Select mouse button on the Window Menu button of the window frame to display the window menu. Double-click to perform the default action.
- The location of the mouse pointer should be determined by direct manipulation and not positioned arbitrarily by the application.

Additionally, an application should present its capabilities in an orderly manner. Necessary and commonly used functions are presented first and in a logical order. For example, essential functions could be included in a menu bar at the top of the client area where they are always visible and ready for selection.

More sophisticated or less frequently used functions can be hidden from immediate view. For example, dialog boxes provide a mechanism for hiding settings and functions that are infrequently used.

Decisions about the placement of functions are not easy to make. From the implementation standpoint, all functions are important. Often, however, a relatively small number of functions account for the majority of usage. These functions need to be prominently featured, but they can be prominent only if other functions are hidden.

Consistency among applications increases users' sense of mastery. Experience with one application can be readily applied to another application, creating a positive transfer of knowledge. The focus of a computer session becomes the task at hand, not "learning a new application." When applications work in a manner that is consistent with other applications, users enjoy a feeling of immediate confidence in their ability to master the new program. Also, they are pleasantly surprised when trying new functions because, although new, the functions seem familiar.

Direct Manipulation

Direct manipulation describes the interaction between a user and an object. Direct manipulation connects an action to an observable response from an object. In direct manipulation interfaces, users experience the immediate visible result of an action.

The immediacy of the visual response is *crucial* to the experience of direct manipulation. Performance problems caused by inefficient program design and implementation make it difficult for users to concentrate on the task at hand and can render an otherwise well-designed application unusable.

Direct manipulation simulates the "real" world where users employ tools to perform tasks on physical objects. Users control their OSF/Motif environments by directly manipulating graphical controls similar to controls they have encountered in real life; for example, buttons "push" to start an action and the slider on a scale actually slides to select a setting.

Another feature of direct manipulation is that the output of the application is also available as input. For example, a list of files is not only the result of a command, but also a collection of screen objects that a user can act upon.

Direct Manipulation empowers users by enabling them to manipulate objects by "grabbing" them (or "pointing" and "selecting" them) rather than by typing a command on a command line. Empowering users through direct manipulation means reducing wherever possible the amount of information they must memorize.

Flexibility

Flexibility should be apparent in both operability and configurability.

Providing multiple ways for users to access application functions and accomplish their tasks increases their sense of control. For example, a function could be accessed through a pull-down menu, a mnemonic key press, or a keyboard accelerator. Empowering users through flexibility enables them to select the best method for accessing a function based on criteria *they* choose: experience level, personal preference, unique situation, or simply habit.

Allowing users to configure settings and select personal preferences enhances their sense of control and encourages them to take an active role in understanding your product and how it works. To be effective, the configurability of your application must be easily accessible.

Explicit Destruction

Explicit destruction means that, if an act has irreversible negative consequences, it should require users to make an explicit action to perform it.

For example, while a worksheet could be saved by clicking the Select mouse button or typing the <*Select*> key on a Save push button, to delete the worksheet should require clicking the Select mouse button or typing <*Select*> on an Erase push button *and* answering some type of an "Are you sure you want to erase this worksheet?" question with another selection action.

Warnings protect users from inadvertent destructive operations yet allow them to remain in control of the application. Operations that may cause a serious or unrecoverable loss of work should warn users of the consequences and request explicit confirmation.

By anticipating errors you enable the support of recovery attempts, and can provide messages informing users of the proper corrective action. Part of this support could be context-sensitive help and provisions for undoing an action. To avoid excessive errors, you can temporarily disable controls when it would be inappropriate to use them. Disabled controls should provide a visual cue that they are not currently

operable.

Context-sensitive help aids understanding, reduces errors, and eases recovery efforts. Help information text should be clear, concise, and written in everyday language. Help information should be readily accessible and just as readily removable.

Many users are most comfortable with learning how to use software applications when they use a natural, trial-and-error method. An "undo" function supports learning by trial-and-error by minimizing the cost of errors. The undo function allows users to retract previous actions, and fosters a spirit of exploration and experimentation that is essential.

The Elements of the OSF/Motif Graphical Environment

At the highest level, the OSF/Motif environment is composed of the following three elements:

- An input selection model.
- A window manager.
- Application programs.

The Input Selection Model

The OSF/Motif environment is based on an object-action input selection model. The selection model defines the actions that users must perform to control the window manager and applications in the OSF/Motif environment.

The selection model follows a point-and-click paradigm. Users first point at and select an object with which to work, and then point at and select an action to perform on the selected object. The OSF/Motif selection model is discussed in Chapter 2.

The Window Manager

The window manager provides users with a way to manipulate the windows displayed in their OSF/Motif environment. Typical manipulations include moving, resizing, minimizing, and maximizing windows, arranging them as required on the workspace.

The OSF/Motif window manager (MWM) frames application windows with an eight-segment border that can be stretched to resize the window. A title area supplied by the window manager displays a title for the window and can be used to move it. Graphical buttons embedded in the window manager frame provide a window management menu and other window controls. Additionally, the OSF/Motif window manager has a three-dimensional appearance so that the control buttons, when "pressed" by the mouse pointer, actually look like they have been pressed. The window manager helps provide for consistent behavior from one application to the next and is discussed in Chapter 3.

Application Programs

Application programs fill the space inside the window frame. By following the selection model and the guidelines in the rest of this **Style Guide**, your application's behavior will be consistent with the behavior of all other applications in the OSF/Motif environment. Designing applications with consistent behavior and the proper use of controls are discussed in Chapters 4 through 7.

Chapter 2
Motif Fundamentals

The OSF/Motif environment provides users with a behaviorally consistent graphical user interface based on a number of discrete but related operational models. Using these models, much of the interaction with an application program can be reduced to the following scenario: Users select an object using the input device with which they feel most comfortable; they then select an action to perform on the selected object.

Consistent behavior augments users' ability to perform a task by enabling them to focus on the task itself rather than on the tools or the methodology they use to perform the task. Just as an automobile is a tool for transportation that doesn't require the average person to be a mechanic, a user-oriented software application provides users with a productivity-enhancing tool without requiring them to become engineers.

In part, OSF's goal of consistent behavior can be achieved by your full understanding and consistent use of the following four operational models discussed in this chapter:

- The input focus model.
- The input device model.
- The navigation model.
- The object-action selection model.

The Input Focus Model

A typical OSF/Motif workspace can contain many windows. Like sheets of paper stacked on a desk, windows can be stacked on the workspace. Some windows may be partially or completely obscured by other windows. At any given time, however, only *one* window has the **input focus**. The window with the input focus is known as the **active window** and is the window where keyboard input will appear. The active window is also the only window that has the location cursor. Input focus can be moved from window to window.

When a window receives the input focus, the default behavior moves the window automatically to the front of the workspace so that no part of the window is obscured. This behavior can be modified to suit the needs or preferences of the users of your application. Additionally, the window with the input focus has a highlighted frame, supplied by the Motif Window Manager (mwm). This provides a visual cue that the window is active and further distinguishes the active window from inactive windows in the workspace.

The input focus model can use either an explicit or an implicit policy for moving the input focus from one window to another. Explicit focus is the default.

Explicit Focus

In **explicit focus** (the default), users explicitly select which window receives the input focus. With a mouse, users move the mouse pointer into a window *and* press the Select mouse button to move the input focus to that window. With a keyboard, users press *<Alt>+<Esc>* to move the input focus sequentially through the windows stacked in the workspace. Explicit focus is sometimes known as "click-to-type."

Implicit Focus

In **implicit focus**, the input focus moves to the window into which users move the mouse pointer. No explicit selection action is performed. The focus policy that a user chooses to implement is a matter of personal preference and need. Some mouse users find implicit focus more convenient. When using only keyboards, there is no distinction between explicit and implicit focus. Implicit focus is sometimes referred to as "pointer", "track pointer," "track listener," or as being "real estate driven."

The Input Device Model

The OSF/Motif user interface enables users to communicate with your application using a keyboard and, optionally, a mouse.

Dual Access via Mouse and Keyboard

Some users prefer controlling software programs by pointing and clicking a mouse. Other users dislike removing one hand from the keyboard to "catch" a mouse and prefer instead to control programs solely from the keyboard. Other users lack the room for a pointing device on their desk. Still others prefer to mix mouse usage with keyboard usage: for frequently performed functions, they use keyboard accelerators; for other functions, they point and click with the mouse.

In OSF/Motif applications, the keyboard is virtually interchangeable with the mouse. This enables users to pick the appropriate tool for the job or to choose the tool with which they feel the most comfortable.

Design your application so that users can control it using a pointing device, the keyboard, or both. Although you may decide to make the pointing device the primary means of control, you should not constrain users from using the keyboard for application control.

Designing your application for dual accessibility empowers users by enabling them to choose the input device they find best, given their particular work situation and personal preferences.

Keyboard Conventions

The keyboard conventions outlined in this section work with ANSI keyboards. Keyboards, however, can differ greatly among manufacturers, and even among models made by the same manufacturer. Therefore, exact key labels as described here may differ from those that you are used to seeing. For example, <*Alt*> is sometimes labeled <*Meta*>, <*Extend*> or <*Extend char*>.

In general, all keyboards have the following types of keys:

Alphabetic keys	Keys representing the letters of the alphabet, the marks of punctuation, and text-formatting functions such as <*Tab*>, <*Return*>, and <*Spacebar*>.
Numeric keys	Keys that represent the numbers from 0 to 9. These are included near the top of the keyboard or, in a numeric keypad on one side.
Navigation keys	Keys that are pressed to move the insertion cursor. These keys are commonly called the "arrow" keys and have labels like <→>, <←>, <↑>, and <↓>. On some keyboards they are separate keys, on others they are included as part of the numeric keypad. Also included in this category are keys like <*Home*>, <*End*>, <*PgUp*>, and <*PgDn*>.
Modifier keys	Keys that are used in conjunction with other keys to "modify" the meaning or effect of those other keys. The modifier keys include <*Ctrl*>, <*Shift*>,

and *<Alt>*. If your application designates a particular key as a modifier key, that key should not have any other function associated with it.

Special-purpose keys — Keys that have particular functions and frequently have labels stating their purpose. Among these keys are *<Help>*, *<Menu>*, *<Esc>*, *<Select>*, *<Enter>*, *<Delete>*, *<Backspace>*, and *<Insert>*.

Function keys — Keys that most keyboards provide for "extra" or general functions. Usually these keys are labeled *<f1>*, *<f2>*, and the like. Some keyboards have ten function keys, others twelve or more. The OSF/Motif environment assumes a keyboard with at least ten function keys. Function keys are usually placed either across the top of the keyboard or on one side, often the left.

As noted, keyboards differ greatly and some may not have all of the keys just mentioned. Table 2.1 lists some common substitutions.

TABLE 1. Common Keyboard Substitutions.

Key	Description	Substitution
<Select>	Makes a selection from the keyboard	*<Spacebar>* or *<Enter>*
<Menu>	Invokes a pop up menu	*<F4>*
<Help>	Invokes the help function	*<F1>*
<Alt>	Modifies the meaning of another key	*<Extend>*
<Esc>	Cancels current action	*<F12>*
<Enter>	Invokes the default action	*<Return>*
<Next Field>	Moves location cursor to next field.	*<Tab>* or *<Ctrl>+<Tab>*
<Previous Field>	Moves location cursor to previous field.	*<Shift><Tab>* or *<Ctrl>+<Shift>+<Tab>*

Also, some keyboards may have duplicate sets of keys. Numeric keys, for example, are often found both in the top row of alphabetic keys and in a numeric keypad; function keys are sometimes found both at the top of the keyboard and as a separate keypad on one side. These duplicate keys may have different keycodes associated with them. In such a case, you can design your application to provide some special functionality on the extra keys. For example, you might retain the top row of alphabetic keys as numeric keys and use the numeric keypad for special functions.

Cursor Shapes

While the traditional use of a keyboard has been for text entry, the keyboard has also been adapted for use as a pointing device. Because of this, the keyboard has two cursors, one for location, the other for text insertion.

The **location cursor** reflects movement on the screen using the keyboard's cursor navigation keys and indicates the current location of the keyboard input focus. The position of the location cursor gives users a visual cue to which object they are about to select. The location cursor is displayed whether or not a mouse is attached and is in addition to the mouse pointer. For example, the location cursor can indicate which item in a list box a user wants to select. The location cursor is often shown as a rectangle around a control as shown in Figure 1 below. The location cursor is sometimes called the "selection cursor."

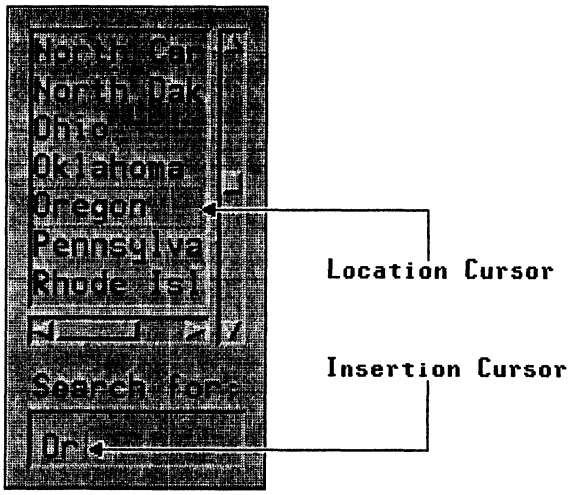


Figure 1. A Typical Location Cursor.

The **insertion cursor** is used in edit controls such a text entry box. It indicates the point in text or graphics where new characters will be inserted. In text entry, the insertion cursor's default shape is a pipe (|) or bar for inserting, and a block character cell or underscore (_) for overstriking. In graphics entry, the insertion cursor's shape might be a pencil, paintbrush, or some other graphically descriptive image.

Figure 2 shows typical insertion cursor shapes.

	Insertion	Overstrike
Active	|	▮ or _
Inactive	¦ or ^	▮ or _

Figure 2. Typical Text Cursor Shapes.

The insertion cursor is *only* displayed where graphics or text entry is allowed. The insertion cursor should change size to match the size of the current font.

Pointing Devices

Direct manipulation allows users to control an application by choosing selections from a menu and by setting controls following a point-and-click method. In the point-and-click method, users move the mouse until the mouse pointer is over (points to) the desired object. They then click the Select mouse button.

Direct manipulation usually requires a pointing device: a mouse, graphics tablet, track ball, joystick, or some other such device. Typically, keyboards with two sets of arrow keys use one set for mouse pointer navigation and the other set for location cursor navigation. Using the pointing device, users can navigate rapidly around the screen and can point at and choose objects to work on and actions to perform.

Throughout this guide we use the term mouse to refer to all pointing devices. You can use any pointing device in place of a mouse.

Mouse Buttons

You can use mouse buttons in combination with the mouse pointer to make selections, move the input focus, or position the insertion cursor.

This guide assumes a three-button mouse and gives the following names and functions to mouse buttons.

TABLE 2. Mouse Button Functionality

Button	Name	Description
1	Select	Used for selection.
2	Menu	Used for direct manipulation of objects.
3	Custom	Used to display pop up menus and otherwise as needed for application-specific functionality.

If your application requires five mouse buttons, you can emulate buttons four and five using a three-button mouse. Pressing buttons 1 and 2 together activate button 4. Pressing buttons 2 and 3 together activate button 5.

Note that the position of mouse buttons can vary depending on whether the mouse is configured for left- or right-handed operation. These button positions are illustrated in Figure 3.

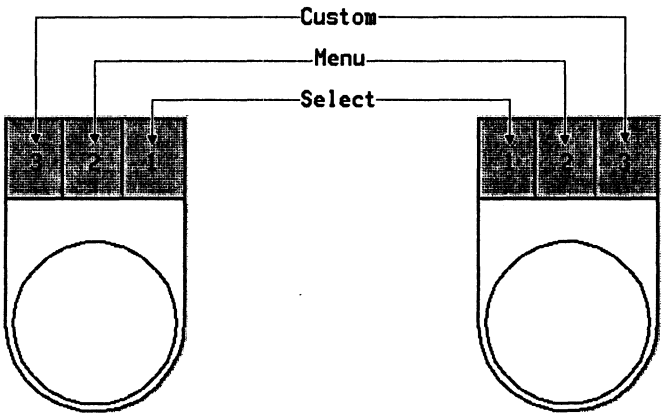


Figure 3. Button Positions for Left- and Right-Handed Mice.

You can perform the following mouse button operations:

Pressing Holding down a mouse button.

Releasing Releasing a mouse button after it has been pressed.

Clicking Quickly pressing and releasing a mouse button without moving the mouse.

Dragging Moving the mouse while a mouse button is pressed.

Double-Clicking Clicking a mouse button twice in rapid succession without moving the mouse pointer.

The Pointer

The mouse is associated with one and only one **mouse pointer**. The mouse pointer appears on the workspace and represents the location of the mouse. Movements of the mouse pointer correspond to movements of the mouse. The mouse pointer is not confined to any specific application. Users can move it anywhere on the workspace. The mouse pointer is sometimes known simply as the pointer.

Your application should only interpret the mouse pointer position; it should *not* attempt to change it. To do so would violate users' trust in the consistency of your program and their sense of control. Also, changing the mouse pointer location may create problems in applications that use absolute location devices (like graphics tablets).

Pointer Shapes

The shape of the mouse pointer provides users with an important visual cue indicating the functionality of the area in which the mouse pointer is currently located. Figure 4 illustrates the standard mouse pointer shapes. While you shouldn't create new mouse pointer shapes for functions that already have mouse pointer shapes associated with them, you can create new mouse pointer shapes for functions not already associated with a pointer shape. Also, do not use a predefined shape to symbolize a function it was not designed to represent.

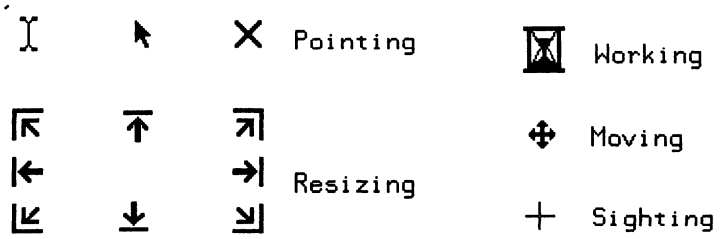


Figure 4. Mouse Pointer Shapes Provide Visual Cues to Activity.

If you decide to use other mouse pointer shapes in your applications, avoid shapes that are hard to see, hard to comprehend, create visual clutter, or flicker excessively when changing shape repeatedly.

Ensure that any mouse pointer you create has an obvious **hot spot** (active point). This is the area of the pointer image that marks the location of the mouse pointer. This is particularly critical for mouse pointer shapes that point to objects or specify positions. Users should be able to intuitively locate the hot spot (for example, the tip of an arrow or the center of a crosshair).

Design your application so that users can set the ratio of mouse pointer speed to mouse speed. This ratio is called the **gain**. Mouse pointer speed can be constant or accelerating. The gain is typically set globally in the OSF/Motif environment. Therefore, if your application needs to adjust the gain, implement a zoom feature rather than change the gain. A zoom feature (similar to the zoom lens of a camera) adjusts the gain by varying the magnification of the application.

The Navigation Model

Regardless of whether they use a mouse, a keyboard, or both, users will need to move the mouse pointer and the location cursor to new positions. That is, they will need to navigate around the workspace.

The mouse pointer is a graphical representation of the current location of the mouse. The only way to move the mouse pointer is to physically move the mouse.

The location cursor shows the current location of the keyboard focus. A user can control the location cursor with either the mouse or the keyboard.

The Object-Action Selection Model

In **object-action selection**, users first select an object, and then select an action to perform on that object. Object-action is patterned after real life and provides users with a readily comprehensible operational model for OSF/Motif applications.

It is helpful to note that the term **object** includes not only recognizable objects like windows and push buttons, but also objects such as the individual letters of a text file, that are perhaps less often thought of as discrete entities.

The OSF/Motif selection model employs the following kinds of selection:

- The selection of a single object.
- The selection of a range of objects.
- The selection of additional (non-contiguous) objects, including multiple ranges.

To make selections in OSF/Motif applications, users always use the same basic steps. First, they place the pointer (mouse) or location cursor (keyboard) on the object they wish to select. Second, they perform a specific selection action. Thus the kinds of selection listed here and explained below are not separate types of selection. Rather, they are variations of the one selection model theme. Which variation they use depends on whether they wish to select a single object, a range of objects, or several additional (non-contiguous) objects.

Some controls, as selection objects, make specific assumptions about the selection model. For example, a set of radio buttons assumes that each button's selection is mutually exclusive. Thus, while radio buttons follow the selection model for single objects, they do not allow any other type of selection. Check buttons, on the other hand, assume that each button's selection is *not* mutually exclusive. While they also follow the selection model for single objects, they allow the selection of multiple buttons without deselecting the prior selection (as is the case in strict single selection). Check buttons are an example of what is called multiple selection.

Table 3 lists the mouse button and keyboard operations of the OSF/Motif selection model.

TABLE 3. The OSF/Motif Object-Action Selection Model.

Selection Task	Mouse Button Operation	Keyboard Selection Operation
Select a single object (set the anchor point) and deselect all other objects.	Click Select	Press <*Select*>
Select a range of objects (set the anchor point at range beginning) and deselect all other objects.	Drag Select	Press <*Shift*>+navigation keys.
Toggle the selection of all objects between current location and the anchor point.	<*Shift*>+click Select	Press <*Shift*>+<*Select*>
Toggle the selection of an additional object.	<*Ctrl*>+click selection operation	Press <*Ctrl*>+selection operation.

Single Selection Selects One Object Only

Single selection is probably the most common type of selection. In single selection, users select a single object upon which to perform an action.

In single selection, users move the mouse pointer or location cursor to a selectable object and then make their selection. Selecting an object changes the object's appearance. This provides users with the necessary visual cue to reinforce their sense of control over the selection process. For example, the insertion cursor in a text entry box is emphasized when the box is selected and deemphasized when the box is not selected. In single selection, when one object is selected, other objects previously selected are deselected.

You can use single selection for such actions as selecting the active window (when the input focus policy is explicit selection) or selecting a push button or other type of control. In a text entry area, they use single selection to position the insertion cursor.

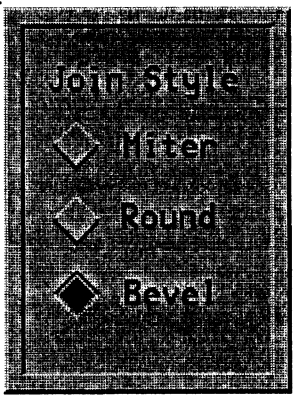


Figure 5. A Typical Single Selection Operation.

Single Selection with a Mouse

Users working with a mouse perform the following steps to select a single object:

1. Move the mouse pointer until it lies over the object they wish to select.
2. Click the Select mouse button to select the object.

Pressing the Select mouse button changes the object's appearance providing the visual cue to which object is about to be selected. Releasing the button selects the object and completes the single selection process.

Single Selection with a Keyboard

Keyboard users perform the following steps to select a single object:

1. Move the location cursor using the cursor navigation keys until it lies over the object they wish to select.
2. Press the <*Select*> key.

Pressing <*Select*> has the same effect as a mouse user's pressing the Select mouse button; it changes the object's appearance and provides the visual cue that the object is about to be selected. Releasing the <*Select*> key, selects the object and completes the single selection process.

Range Selection Selects Contiguous Objects

In **range selection**, users select a range of contiguous objects upon which to perform some action. The range is based on, but not limited to, a rectangular area. To be included in the range, objects must be *completely included* in the selection action.

In range selection, users move the mouse pointer or location cursor to the selectable graphics object and then make their selection. The object's appearance changes to provide a visual cue, just as it does in single selection.

In range selection, as the next object in the range is selected, the previous object in the range *remains* selected. Both objects become part of the range being selected. Any number of objects can be selected in range selection as long as they form a contiguous group.

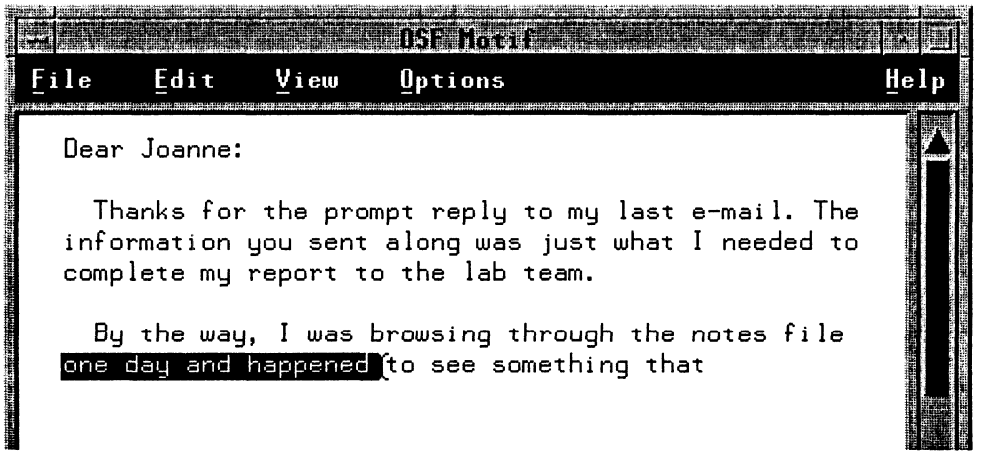


Figure 6. Cutting Text Is a Typical Range Selection Operation.

You can use range selection during cut-and-paste operations on text. Words are selected as a range of ASCII characters and then acted upon (cut or pasted).

Range Selection Using a Mouse

Mouse users for range selection perform the following steps:

1. Position the mouse pointer over the object that starts the range.
2. Press the Select mouse button.
3. Drag the mouse pointer to the object that ends the range.
4. Release the Select mouse button to complete the range selection.

For example, to select five contiguous items from a list box, users position the mouse pointer on the first item, press the Select mouse button, drag the mouse pointer to the fifth item, then release the Select button. The appearance of each selected item changes as it is selected, providing a visual cue. Releasing the Select mouse button completes the range selection.

Figure 7. A Range (Contiguous) Selection Operation.

An alternative to dragging the mouse pointer from the start to the end of the range is to click the Select mouse button on the start, move the mouse pointer to the end of the range, and press <*Shift*>+Select to complete the range selection.

Range Selection Using a Keyboard

Keyboard users for range selection perform the following steps:

1. Position the location cursor (insertion cursor for editable areas) using the navigation keys so that it is over the object that starts the range.
2. Hold down the <*Shift*> key and press the navigation keys to drag the cursor to the object that ends the range.
3. Release the <*Shift*> key to complete the range selection.

As with range selection using the mouse, each object's appearance changes providing the visual cue of selection. When the <*Shift*> key is released, the selection is complete.

For example, to select five contiguous items from a list box, users position the location cursor on the first item, press the <*Shift*> key, press the navigation keys to drag the location cursor to the fifth line, then release the <*Shift*>.

Alternatively, press the <*Select*> key to start the selection, press the navigation keys to move the location cursor to the end of the range, and then press <*Shift*>+<*Select*> to complete the range selection.

Selecting Additional Non-contiguous Objects

Users can make one or more additional selections, forming a non-contiguous group of objects.

To begin an additional selection, users select a first object using single selection or a first range of objects using range selection. Other objects can be added to the selection group by repositioning the mouse or location cursor, and selecting the objects.

Like single and range selection, objects or ranges that are part of a non-contiguous selection provide a visual cue that they are part of the selection.

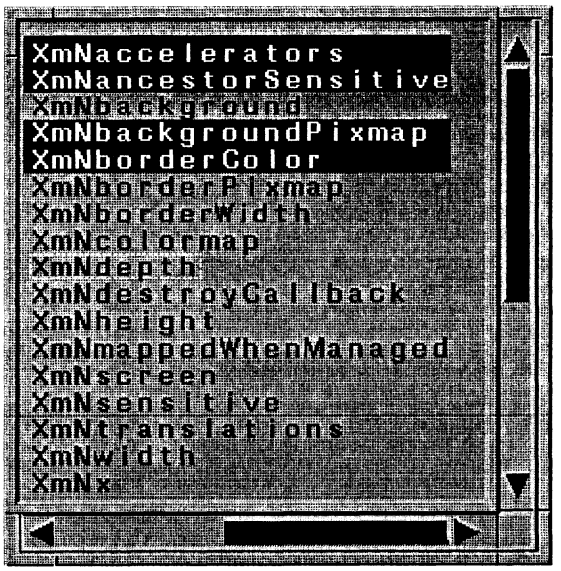


Figure 8. An Additional (Non-contiguous) Selection Operation.

Users find the selection of additional non-contiguous objects, especially the selection of non-contiguous ranges, useful in text processing. Note also that the operation that performs a selection of additional non-contiguous objects works as a "toggle." That is, if the object is not selected, the operation selects it; if the object is selected, the operation deselects it.

Making Additional Non-contiguous Selection Using the Mouse

Mouse users make an additional selection perform the following steps:

1. Select the first single object or range using the methods described in the preceding sections.
2. Position the mouse pointer on the next object they wish to select.
3. Press <*Ctrl*>+Select to mark the next object or next range in the non-contiguous selection.
4. (For range selection only) Drag the mouse pointer to the end of the range, then release <*Ctrl*>+Select.
5. Repeat steps 2-4 for each additional object or range in the non-contiguous selection.

For example, to select several non-contiguous items from a list box, users position the mouse pointer on the first item, click the Select mouse button to select it, move the mouse pointer to the next item for selection,

press the <*Ctrl*> key and click the Select mouse button to select that item. The appearance of the selected list items changes as the item is selected, providing a visual cue to the selection.

An alternative to the press-drag-release operation is to click <*Ctrl*>+Select to begin the selection, reposition the mouse pointer at the end off the selection, then click <*Shift*>+Select. This alternate method for selecting multiple ranges is similar to the alternative method for selecting a single range described above.

Making an Additional Non-contiguous Selection Using the Keyboard

Keyboard users make additional (non-contiguous) selections perform the following steps:

1. Select the first single object or range using the methods described in the preceding sections.
2. Position the location cursor on the next object they wish to select.
3. Press <*Ctrl*>+<*Select*> to mark the next object or start of the next range in the non-contiguous selection.
4. (For range selection only) Press <*Shift*> and press the navigation keys to drag the location cursor to the end of the range. Release the <*Shift*> key to mark the end of the range.
5. Repeat steps 2-4 for each additional object or range in the non-contiguous selection.

For the same example as above, to select several non-contiguous items from a list box, users position the location cursor on the first item, press the <*Select*> key to select it, move the location cursor to the next item for selection, press the <*Ctrl*> key and press the <*Select*> key to select that item. As with mouse selection, the appearance of the selected list items changes as the item is selected, providing a visual cue to the selection.

Alternatively, press <*Ctrl*>+<*Select*> to start the range, move the location cursor using the navigation keys to the end of the range, and then press <*Shift*>+<*Select*> to complete the range selection.

Deselecting an Object

A selection operation can be undone (canceled) by typing the <*Esc*> key before the operation is completed. Once completed, an entire previous selection can be deselected by making a single-object selection.

A previously selected object can also be deselected by "toggling" its selection state. Usually this requires positioning the mouse pointer or location cursor on the object and then pressing <*Ctrl*> and the Select mouse button (for mouse users) or by typing <*Ctrl*> and the <*Select*> key (for keyboard users). This deselection method is commonly used to deselect one object in a range of selected objects without deselecting the rest of the range.

A previously selected range of objects can be deselected by pressing <*Ctrl*> and using a drag operation to toggle their selection state.

Selecting the Default Action

Some objects have default actions associated with them. For example, An icons' default action is resoring itself to a normal window.

For mouse users, double-clicking the mouse with the mouse pointer over an object selects the default operation for that object. For example, double-clicking the Select mouse button with the mouse pointer over an icon restores the icon. Double-clicking the Select mouse button with the mouse pointer over the window menu button closes the window. Double-clicking the Select mouse button with the mouse pointer

in a dialog box performs the default action associated with that dialog box.

For keyboard users, typing the <*Enter*> key with the location cursor over an object selects the default operation for that object. For example, typing <*Enter*> with the location cursor over an icon, normalizes the icon. Typing <*Enter*> with the location cursor in a dialog box performs the default action associated with that dialog box.

Auto-selection

Auto-selection combines the act of moving the location cursor with the act of selecting the object. In the selection model presented in this chapter, users move the location cursor to an object and then explicitly select the object by pressing the Select mouse button or the <*Select*> key. When auto-selection is employed, users simply move the location cursor to an object and the object is selected; no explicit selection action is required.

Chapter 3
How the Window Manager Works

The OSF/Motif user interface provides a rich environment, designed to facilitate communications between users and your application. This environment is composed of discrete graphical elements.

The graphical elements of the OSF/Motif user interface facilitate communication by providing users with a metaphor, a figurative concept suggestive of real world objects. Through this metaphor, interaction with your application is more familiar (thus more intuitive) and less technical than the traditional user interface provided by the command-line prompt. This metaphor is refered to as a "desk," "desktop," "workspace," or "workbench." While desktop is perhaps the more widely known term, this *Style Guide* uses the term "workspace" to emphasize that applications in the OSF/Motif environment need not be office oriented and that the functionality and graphical elements of the user interface are *tools* that empower users to accomplish tasks with their computers.

The elements of the user interface, the objects that users see (for example, windows, icons, menus, and dialog boxes) appear on the workspace and can be stacked one on top of one another like papers on a desk or tools on a workbench.

This chapter discusses the following major points about the graphical elements of the OSF Motif workspace:

- Types of windows.
- Window anatomy.
- The icon box.

Types of Windows

In the OSF/Motif environment, users communicate with your application using windows. A **window** is an area of the screen (usually rectangular) that provides users with the functional means to communicate with your application and through which your application can communicate with them.

A typical OSF/Motif environment may have several applications in operation simultaneously. Each application typically has a main or primary window that displays data and in which users carry on their primary interaction with the application. Additionally, applications usually have one or more secondary windows (dialog boxes) that carry on context-specific dialogues with the people using the application. Figure 9 illustrates a typical OSF/Motif environment.

Figure 9. A Typical OSF/Motif User Environment.

While your application can be made up of many windows, each window will be one of only two basic types:

- A primary window.
- A secondary window.

Primary Windows

A primary window is the window by which all the other windows used by your application are generated. A primary window may be obscured by overlapping windows. Your application can have one *or more* primary windows.

A primary window is the only window by which an application can be closed. That is, when users close the last primary window of an interactive application, the application session should end. Closing a primary window causes all secondary windows associated with that window to go away.

When users invoke your application, the application's first task is usually to display a primary window. Because of this, users often think of the primary window as the main window. This provides users with a valuable point of reference: they feel in control knowing there is a main window from which all else follows and to which they can return.

You should design your application to encourage this sense of control, and so that, as users open and close windows in their dialog with your application, they can always return to a primary window. A primary window should remain consistent in appearance and behavior with the last time they were there.

Secondary Windows

Context-specific dialogs usually occur inside secondary windows called dialog boxes. When the dialog is completed, the secondary window usually disappears.

Secondary windows are always related to a parent window. Sometimes the parent is a primary window, sometimes another secondary window. A primary window can have any number of secondary windows as its children.

Secondary windows are not constrained to be inside the primary window, but they will always appear on top of that parent window in the window hierarchy. Think about how your application will distinguish between primary and secondary windows. One method is to include identifying information in the title area. For application-oriented programs, the application name is followed by the file name in the title bar; for object-oriented programs, the object name is followed by the function. Secondary windows then have title areas that include the application or object name and the implied action.

When a primary window is minimized, its secondary windows are temporarily removed from the display.

Window Anatomy

The OSF Motif Window Manager (MWM) provides windows with a window frame that contains the components shown in Figure 10. These components are functional, providing a convenient way for mouse users to invoke window management functions. Keyboard-only users use the window menu provided by MWM to invoke window management functions. Window frame components are sometimes called decorations.

Along with window frame components, the OSF Motif window layout includes a **client area**, the area inside the window frame, for your application to use.

In general, a window consists of the following components:

- Window menu button.
- Window control buttons.
- title area.
- Resize border.
- Client area.

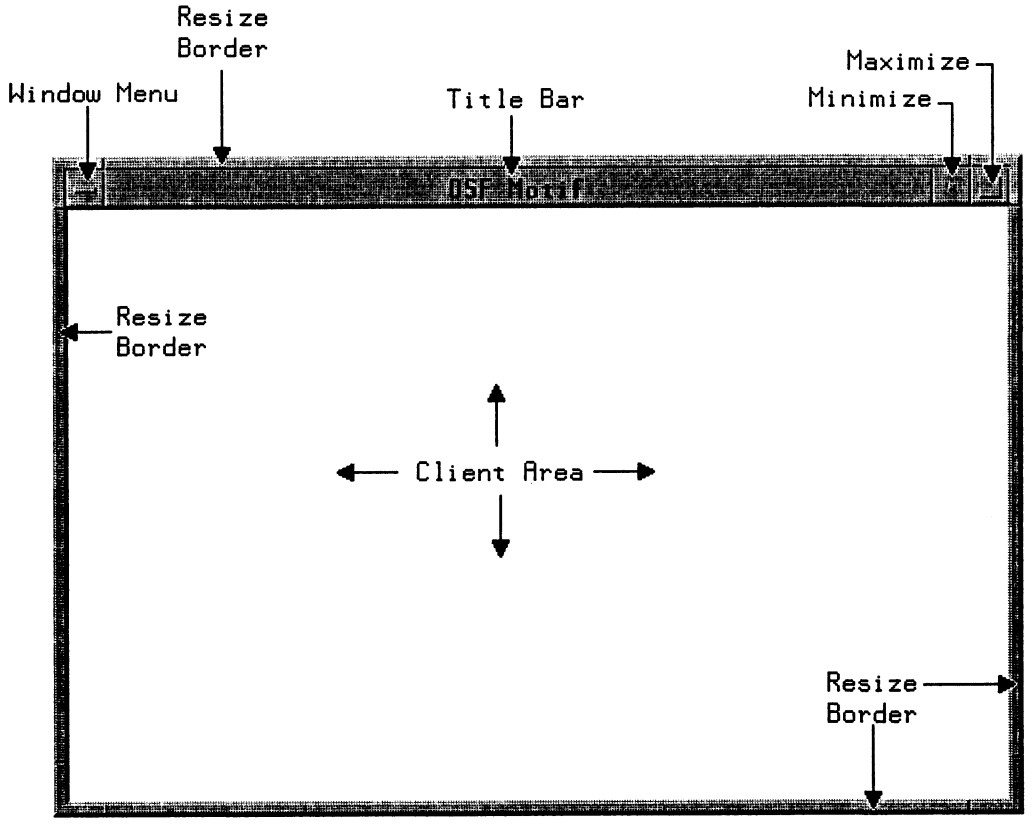


Figure 10. A Standard OSF/Motif Window Layout.

While Figure 9 illustrates the components of the standard window layout, specific implementations may require some amount of modification. For example, it is inappropriate to resize some windows. These should have their resize borders removed.

Window Menu and Window Menu Button

The window menu button is located in the title bar, on the left side of the title area, and is used to display the window menu. Double-clicking the Select mouse button with the mouse pointer over the window menu button closes the window. The window menu provides a standard location for important window management functions. Users can browse the menu to see what actions are available. The window menu pulls down from the upper left corner of the window frame. Mouse users display the window menu by pressing the Select mouse button with the mouse pointer on the window menu button. Keyboard users display the window menu by typing *<Shift>+<Esc>* when the input focus is in the window.

The window menu and window menu button are sometimes called the "system menu" and "system menu button" respectively.

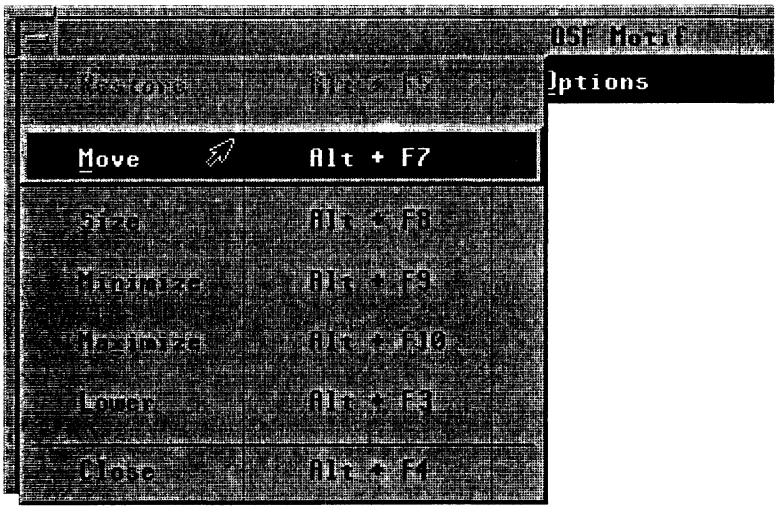


Figure 11. The Window Menu Button with Menu Pulled Down.

Figure 11 illustrates the standard selections of the window menu. In the illustration, the Move selection is being chosen. Also, the Restore function is de-emphasized to provide a visual cue that, in the present context, the function is unavailable. The selections of the window menu have the following functions and accelerators:

Function	Accelerator	Action
Restore	Alt+F5	Restores a minimized or maximized window to its normal size. This selection is de-emphasized (grayed out) when the window is in its normal state.
Move	Alt+F7	Moves a window around on the workspace.
Size	Alt+F8	Stretches or shrinks a window in the direction indicated by the mouse pointer.
Minimize	Alt+F9	Changes a window into an icon.
Maximize	Alt+F10	Enlarges a window to its maximum specified size.
Lower	Alt+F3	(Optional) Moves a window to the back" of the workspace (the bottom of the window stack).
Close	Alt+F4	Closes a window and removes it from the workspace.

Keyboard accelerators for the window menu are optional. If you decide to use accelerators, use the accelerators suggested above.

Window Control Buttons

Window control buttons are push buttons located in the upper right corner of the OSF/Motif window frame. They provide a short-cut for invoking window management functions without pulling down the window menu. Users invoke window management functions by clicking on the appropriate window control button.

The functions chosen for window control buttons are implementation-dependent. Figure 12 illustrates two such functions, Maximize and Minimize.

Minimize Button

The minimize button is located to the immediate right of the title area. It provides the same function as the Minimize selection in the window menu. Users click the Select mouse button with the mouse pointer on the minimize button to shrink a window to an icon.

Maximize Button

The maximize button is located between the minimize button and the resize border. It provides the same function as the Maximize selection in the window menu. Users click the Select mouse button on the maximize button to enlarge a window to its maximum size. The maximize size of a window is established by the application. Clicking the Select mouse button with the mouse pointer on the maximize button of a maximized window restores the window to its original size, the same function as the Restore selection of the window menu.

Figure 12. Window Control Buttons on the Top Right of the Window Frame.

Title Area

The **title area** is the horizontal bar that lies between the window menu button and the window control buttons, and, as part of the window frame, highlights when the window has the input focus. The title in the title area identifies the window.

Pressing the Select mouse button with the mouse pointer on the title area and dragging the mouse pointer on the screen will move the window to a new location. Clicking the Select mouse button with the mouse pointer on a title area (or frame) raises that window to the top of the window stack.

The window manager displays an **application title** in the title area. The application title is supplied by the application and clearly identifies the window and its role within your application.

You provide the window manager with a title. In object-oriented environments, this title should be the name of the object followed by the application name; in file-based environments, the title could be the name of the application followed by the file name. Multiple windows of the same application should have titles that identify them with the application in some way, but can otherwise be distinct from one another.

The title area should not display the version number of your application. Nor should you use it to display system messages. Use the title area for information that stays relatively constant throughout the work

session of your application.

Resize Border

Your application suggests the initial size of its windows to the window manager. Window sizes vary according to the work users perform in them. At any time, users should be able to alter the size of a primary window.

The window manager provides a functional window frame which surrounds the client area. The window frame highlights when input focus is passed to the window. Resizable windows have a wide frame border that users can drag when they want to change the window's size.

Client Area

The **client area** is the portion of the window in which users perform most application-level tasks. For example, if users are working with a graphics editor or a text editor, the client area contains the figure or document being edited. The client area is inside the window frame and can be composed of multiple work areas. Figure 13 shows a window with a hypothetical client area.

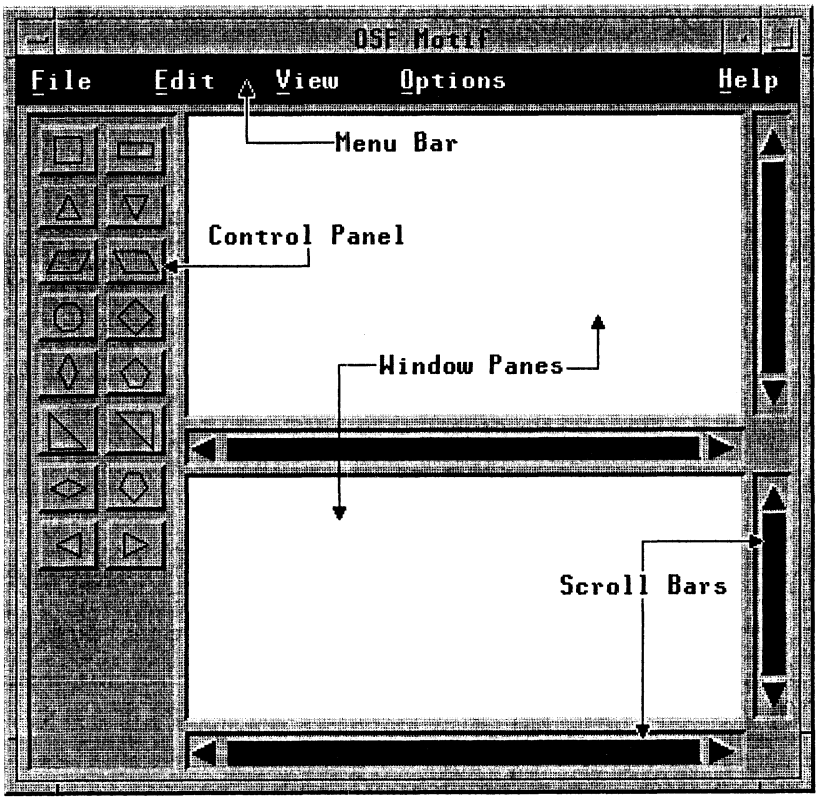


Figure 13. The Anatomy of A Typical OSF/Motif Application.

The Icon Box

By default, minimized windows (icons) are placed on the workspace in a row beginning at the lower left corner of the screen. However, under the management of the OSF/Motif Window Manager, users can choose to group minimized windows in an icon box. They can also choose to have minimized windows placed at the location the normal window occupied.

An icon box acts like a typical window in the sense that it has a window frame and frame controls. Like other windows it can be sized, moved, minimized, maximized, restored, and lowered. However, an icon box cannot be closed. The following figure shows a typical icon box.

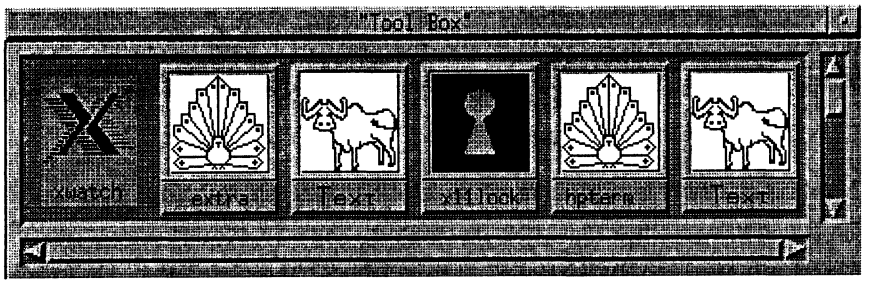


Figure 14. A Typical OSF/Motif Icon Box.

Chapter 4
Designing OSF/Motif Clients

The OSF/Motif Window Manager is responsible for providing window management services for the windows of all applications in the OSF/Motif environment. Your application is responsible for organizing the client area of the main window, any sub-areas, and any secondary windows.

This chapter discusses the following client area design topics:

- Client areas.
- Grouping similar controls.
- Presenting multiple controls.
- Laying out an application's areas.

Successive chapters discuss the use of controls, menus, dialog boxes, and help in more detail.

Client Areas

Organizing client areas is an important part of your design process. Depending on the nature of your application, you may choose to divide the client area into one or more sub-areas. Additionally, you may choose to design your application with secondary windows (dialog boxes). Sub-areas and secondary windows visually reinforce the organization of your application and increase users' sense of control over its operation.

Client Sub-areas

As mentioned, you can divide the client area of your application window into one or more sub-areas. Client sub-areas are very application specific, with the possible exception of the menu bar.

Figure 15 illustrates a hypothetical OSF/Motif main application window using some standard sub-areas.

Menu bar

The **menu bar** is the horizontal bar that appears just below the title area. It contains a list of menu topics from which users can select. A single letter **mnemonic** for each menu topic is underlined.

Keyboard users select a topic by typing *<F10>* to move the location cursor to the menu bar and then typing the mnemonic letter of the topic. Mouse users select a topic by positioning the mouse pointer over that topic and selecting it with the Select mouse button. Selecting a topic causes a pull-down menu to display selectable items related to that topic.

Note that commands are not included as topics in the menu bar because they would prohibit users from browsing the menu topics.

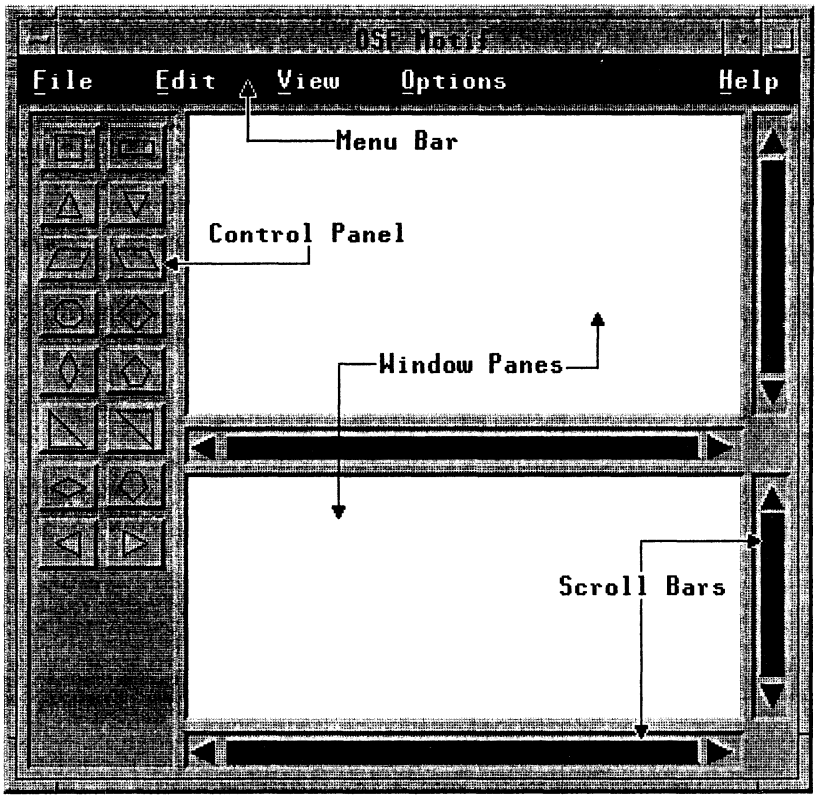


Figure 15. A Hypothetical Window with Sub-areas.

Because menus are a principal method of interaction between users and OSF/Motif applications, most applications require a menu bar. Refer to Chapter 6 for more detailed information about menus.

Control panels

Some applications benefit by organizing part of the client area into a sub-area called a **control panel.** A control panel is a group of like controls having similar functions. Control panels can either be part of the client area, if their use is required frequently, or part of a dialog box, if their use is required occasionally. Control panels are made up of the controls discussed in Chapter 5.

Other Client Sub-areas

In addition to a menu bar and control panels, your application design might call for other types of client sub-areas.

Message Area

You may decide that it is more efficient for users to view messages, but not warnings or messages requiring immediate action, within a sub-area of the client area rather than in a separate message box. If so, the messages should appear at the bottom of the client area so that messages aren't obscured by pull-down menus. The message can be either in a line reserved for messages or in a line temporarily used for the message and then returned to its previous use. Warnings and messages that require immediate action are not displayed this way. Rather, these are always displayed in message boxes to give them greater visibility in the workspace.

Command Line

Although the OSF/Motif environment provides a graphical-oriented, object-action selection model, your particular application may permit the use of typed commands to enhance the control users have over your application.

Command lines should generally run from border-to-border across the bottom of the client area, just below the message area, or directly below the menu bar at the top of the client area.

Client Controls

Each sub-area can contain a variety of controls enabling users to manage that sub-area. Like sub-areas, the controls chosen for a particular sub-area, are very application-specific. The controls you choose depend on the needs of the people who will use your application, and your application design.

Window panes

Depending on the needs of your application and the people who use it, you may decide that it is better to divide your client area into **window panes** rather than into fixed-partition sub-areas.

Panes can be either vertical (one on top of the other) or horizontal (side by side). Users can resize panes by dragging the boundary between the panes. Making one pane bigger makes the other pane smaller, while the overall size of the window remains the same.

Scroll bars

People use scroll bars to scan rapidly through the contents of a window. The current location of the scroll bar is shown by the position of the slider in the scroll bar area.

Resizable windows and window panes require scroll bars if the information in them will require more space or become obscured by the border of the sub-area. The following figure shows vertical and horizontal scrollbars.

Figure 16. Vertical and Horizontal Scroll Bars

Scroll bars are located to the right (vertical) or on the bottom (horizontal) of the area to be scrolled. Scroll bars can also be used in dialog boxes or in combination with other controls. Their operation is discussed in Chapter 5.

Other Controls

The OSF/Motif object-action selection model uses a number of other controls. These controls are graphical representations of real life controls and include the following:

- Push buttons.
- Radio buttons.
- Check buttons.
- List boxes.
- Entry boxes.
- Scales.
- Scroll bars.

These controls are described in detail in Chapter 5.

Other Client Areas

In addition to organizing your application's client area into sub-areas, you may choose to organize your application and its operation using some of the other methods provided by the OSF/Motif environment. These include pop-up menus, cascading menus, and dialog boxes (secondary windows). Like sub-areas, these other areas visually reinforce the organization of your application and increase users' sense of control

over its operation.

Menus

OSF/Motif menus work just like real life menus. Menus enable users to choose from a list of possible selections. Besides the pull-down menus on the menu bar, the OSF/Motif environment has the following types of menus:

Pop-ups	Menus that pop up from nowhere rather than being related to a menu bar.
Cascading	Menus that cascade to the right and down from either pull-down or pop-up menus and provide a subsidiary level of selection.
Option	Menus that display from a dialog box. Those usually appear when a push button is pressed.

Menus are described in detail in Chapter 6.

Dialog Boxes

Controls that are not frequently in use during the operation of your application can be included in a dialog box. A dialog box is a separate window from the application's main window, and contains controls that should be readily accessible but that don't need to be displayed permanently in the client area.

Dialog boxes are described in detail in Chapter 7.

Grouping Similar Controls

Some controls perform similar functions or are logically related. For ease of use, as well as for proper visual design, these controls should be grouped into sets. This keeps the user interface of your application organized. Similar or related controls can be placed either in group boxes or in window sub-areas. Typically, group boxes occur in dialog boxes and are often surrounded by a simple frame. Window sub-areas often appear as part of an application's main window, and contain frequently-used controls that must remain readily available.

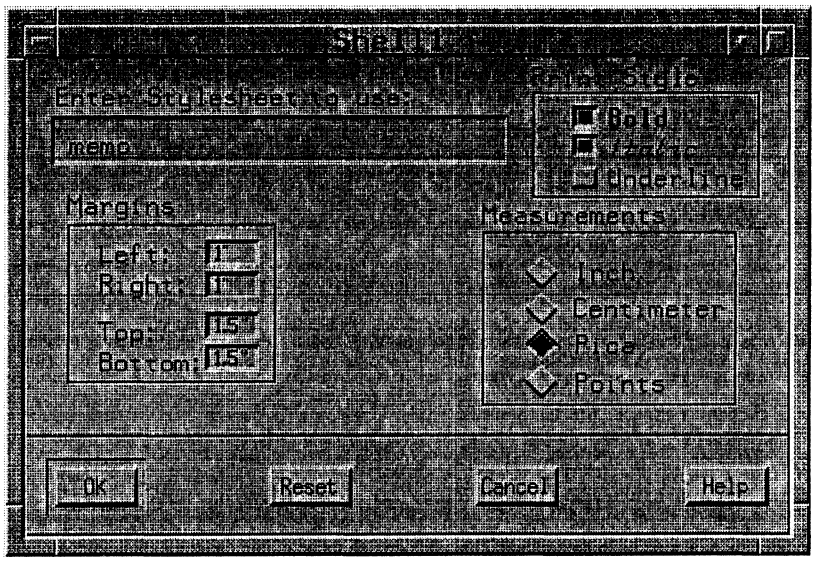


Figure 17. Related Controls Can Be Grouped into Sets.

Grouped controls provide a visual cue that the controls are related to one another by isolating them from other controls. A group box or control panel usually has a title printed near it. Controls can be grouped in one row or column, or in multiple rows or columns.

Designing Grouped Controls with Push Buttons

Grouped controls containing push buttons should adhere to the following guidelines:

- Use push buttons sparingly.
- Use push buttons only for frequently-used commands.

Unless you are trying to duplicate the physical appearance of an existing piece of equipment, place push buttons in dialog boxes in a row along the bottom of the box or in a column along the right side of the box.

Combining Controls

The OSF/Motif environment does not restrict the combination of controls only to those combinations mentioned in this *Style Guide*. The criteria for developing control combinations should always be consistent: will this combination empower the people who use it to work more efficiently and, as a result, become more productive.

Presenting Multiple Controls

The OSF/Motif environment uses four basic ways of displaying multiple controls:

- Pull-down menus.
- Pop-up menus.
- Dialog boxes.
- Window Sub-areas.

Pull-Down Menus

Pull-down menus usually contain push buttons, radio buttons, or check buttons as selection items. Selections can lead to dialog boxes or other controls. Menu items are always presented in a vertical column. To display a pull-down menu requires some degree of "mouse travel" or the use of an *<Alt>+mnemonic* accelerator. While the selections of a pull-down menu do not appear until the menu is selected, the title of a pull-down menu is always displayed in the menu bar. Pull-down menus combine a visual cue of their presence with an efficient use of space.

Pop-Up Menus

Pop-up menus, like pull-down menus, usually contain push buttons, radio buttons, or check buttons, and can have selections that lead to dialog boxes or other controls. Also like pull-down menus, pop-up menus are always presented as a vertical column. Pop-up menus are associated with a particular area of the screen. The advantage of pop-up menus is that they require no mouse travel; they simply pop-up at the current mouse location (provided that location has a menu associated with it). While pop-up menus take up no screen space until they are displayed, they provide no visual cue to their existence.

Dialog Boxes

Dialog boxes can contain all the button, box, and valuator controls. Usually, when the dialog box is displayed, all necessary controls are present. However, your application may require some extra display operations that should be in the dialog box but need not be displayed all the time. If so, you can include an option menu push button in your dialog box.

Dialog boxes allow a lot of flexibility in the arrangement of controls. Controls can be grouped in boxes, organized in rows or columns, and separated by white space for better visibility.

Message boxes are usually displayed by the application without any explicit action. Dialog boxes, on the other hand, are usually displayed as a result of some explicit action. Dialog boxes can be displayed directly using a keyboard accelerator. This saves the time and extra steps required when selecting them from a menu. Like pop-up menus, dialog boxes use space efficiently because they are not visible until displayed, however, this means they don't provide a visual cue to their existence. Dialog boxes are removed from the workspace when their primary window is minimized. Dialog boxes are returned to the workspace when their primary window is restored.

Control Panels

Control panels, like dialog boxes, can contain all the button, box, and valuator controls. They also offer the same flexibility of arrangement. Control panels, since they are permanently displayed, offer the potential of having frequently used controls always available. Being displayed, they offer a strong visual cue to their existence, but they do take up screen space.

Beginning a Client Area

The nature of your application may be such that you will need to design control panels or dialog boxes specific to your situation. When doing so, it is important for the OSF/Motif-conformity of your application to use the following criteria:

Arrange Controls in Natural Scanning Order

Design the layout of your application windows according to the natural scanning order of the people who will be using your application. In most cases, this order will be from left to right and from top to bottom.

Arrange Controls in the Sequence People Use Them

Intimately connected to the use of natural scanning order in control layout is the arrangement of controls in the sequence in which people will use them. The natural scanning order gives the position; the sequence of use gives the priority.

For example, suppose you have a dialog box that has push buttons that accept changes, test them, restore original values, and cancel the dialog box without making changes. Western conventions dictate that the push buttons be displayed from left to right. The sequence of use suggests that the OK button should be on the left (as the most frequently used button), followed by Apply, Reset, and Cancel.

Adjusting the Client Area

To increase the control users have over your application, your application should allow users to adjust the client area to fit their needs.

If your application uses window panes, it should allow users to adjust the size of the panes to suit their needs by repositioning the **sash**, the border separating the two panes. When one pane increases in size, the other pane decreases by the same amount. The overall sizes of the window frame and the client area *do not* change as the panes are resized.

Users should be able to adjust the sash using either a mouse or a keyboard operation. Moving the sash with the mouse is typically a button-press, drag, button-release operation like moving a window using the title area. Moving the sash with the keyboard typically requires a selection and the use of the navigation keys.

Choosing the Appropriate Control

Radio buttons, option menus, and list boxes can all be used to choose one option from a list of multiple options. Choosing the right control for the job depends on the number and nature of the options in the list.

For choosing a single option from among a small number of mutually exclusive options, a radio button is usually the easiest for users to operate. For more options, an option menu push button takes up a small amount of space and is relatively easy to use. For many options, the list box is the easiest for people to use; it also allows multiple items to be chosen at one time.

Deciding Between a Pop-Up Menu and Push Buttons

Pop-up menus provide users with quick access to application functions. So do control panels containing push buttons. Generally, pop-up menus are preferable when users are focused on their work areas. In these situations, moving the mouse between a control panel and the work area would be distracting.

Push buttons and a control panel are preferable when users make frequent selections, need to make several selections at the same time, or are already manipulating the mouse primarily in the control panel area.

Deciding Between Dialog Boxes and Menus

You should design your application so that it is consistent with other OSF/Motif applications. To do so, you should understand when to use dialog boxes and when to use some other method of control. In particular, you should know the difference between dialog boxes and menus.

As you design your application, you will encounter many instances in which the same objective can be accomplished with either a dialog box or a menu. The menu selections act similar to the controls used in most dialog boxes. However, there are differences.

A menu is short-lived. It appears quickly, but exists only while a selection is being chosen. As soon as the selection is made, the menu disappears. A dialog box, on the other hand, can be displayed until told to go away, but usually takes up more workspace. While the dialog box is displayed, users can make several different selections.

Additionally, a menu is usually modal in nature. Until a menu goes away, users can't interact with any other part of the application. Dialog boxes, on the other hand, are frequently modeless. Users can still interact with other parts of the application while the dialog box is displayed.

Thus, if a modeless state were required, a modeless dialog box would be the appropriate solution. In the case of users browsing current settings or making a single selection, a menu would be faster. However, when several selections need to be made, a dialog box would be a better design choice.

Aligning Columns of Controls

While push buttons are usually placed in a row along the bottom of the dialog box, check buttons and radio buttons are frequently placed in columns. When using columns, align the check buttons or radio buttons vertically so that the location cursor doesn't bounce around as users tab through the selections. Proper vertical alignment also enables users to slide the mouse pointer in a single direction rather than having to zigzag through a slalom course of misaligned controls.

Using Defaults

Your application should use default values for common settings or obvious selections. A default selection should be easily distinguishable from other selections. Typically this is accomplished with an extra border around the default selection.

Default check buttons need only display with a check in them. The default selection in a set of radio buttons also displays selected. A default push button is typically indicated by a double border. Users activate the default push button by double-clicking the Select mouse button with the mouse pointer in the area containing the push button or by pressing <*Enter*> on the keyboard. Figure 18 shows a default push button.

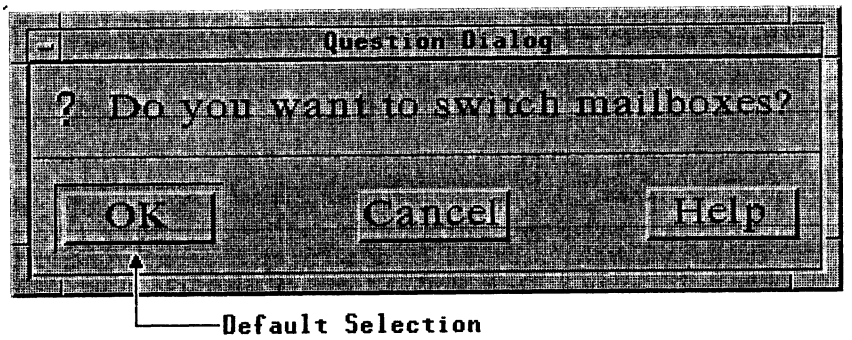


Figure 18. A Default Push Button.

Chapter 5
Providing Controls: Buttons, Boxes, and Valuators

Users control applications in the OSF/Motif environment using a number of graphical controls. These controls are of the following three types:

Buttons	Like the control buttons in real life, users generally operate OSF/Motif graphical control buttons by pressing them with the Select mouse button or <*Select*> key.
Boxes	Like boxes in real life, graphical control boxes generally contain groups of related items. of interest to users.
Valuators	Like some analog gauges in real life, valuators provide users with a way to specify or control incremental changes.

Types of Buttons

OSF/Motif applications currently use three types of buttons: push buttons, radio buttons, and check buttons. Which button you use for your application depends on the situation you wish to control. Buttons should be of a size large enough so that users can easily position the mouse pointer on them.

Push Buttons

A **push button** consists of two parts:

- A graphical image that represents the button.
- A label or icon describing the action invoked by the button.

When users position the mouse pointer anywhere on a push button and click the Select mouse button, or position the location cursor on the push button and press <*Select*>, the action represented by the push button occurs.

The label of a push button should be short, usually a verb for action push buttons, such as Cancel or Apply. Response push buttons can have text such as OK or Yes, however, the question that prompts the response should be carefully worded to avoid ambiguity.

A push button can be used to display another dialog box. This is the case with an option menu in a dialog box. A push button used this way should provide a visual cue to its functionality by following its button label with an ellipsis (. . .).

Radio Buttons

A **radio button** consists of two parts:

- The graphic image that visually represents the button.
- A label that describes the choice represented by the graphic image.

Each radio button represents a single-choice selection. Radio buttons are always in a fixed set of at least two buttons and always represent *mutually exclusive choices*.

Conceptually, radio buttons work like the buttons on a car radio (from which they derive their name). Users select or unselect a radio button by clicking the Select mouse button when the mouse pointer is over the radio button or by pressing <*Select*> when the location cursor is over the radio button. Like a car radio,

when one radio button is selected, the previously selected button is unselected.

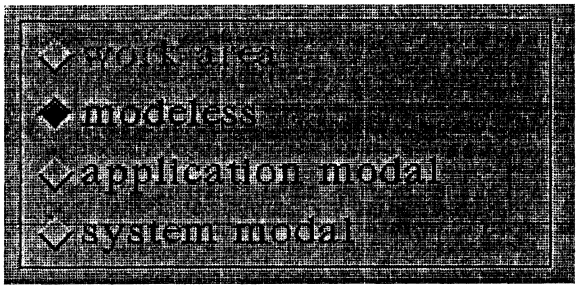


Figure 19. A Typical Set of Radio Buttons.

Radio buttons that refer to similar kinds of options should be grouped in sets. Sets of radio buttons that refer to different kinds of options should be grouped separately. Sets can be arranged in either rows or columns. Use white space to visually separate multiple sets of radio buttons into a control panel.

Radio buttons are usually circles in 2-dimensional environments and usually diamonds in 3-dimensional environments to distinguished them visually from check buttons which are usually square.

Check Buttons

A **check button** consists of the following two parts:

- A square box or "button" that is empty when unselected but that is filled in or contains some other visual cue when selected.
- A label that identifies the purpose of the check button. The label is at the same level as and to the right of the check button.

Check button enable users to select choices that are *not mutually exclusive*. Users select or unselect a check button by clicking the Select mouse button when the mouse pointer is on the check button or by pressing *<Select>* when the location cursor is over the check button.

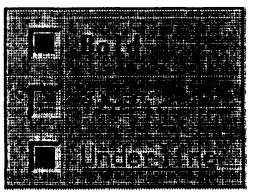


Figure 20. A Typical Set of Check Buttons.

Types of Boxes

OSF/Motif applications currently use two types of control boxes: list boxes and entry boxes. Like buttons, which box you use depends on the situation.

List Boxes

A **list box** typically consists of the following parts:

- A title that describes the purpose or contents of the list box. The title generally appears above the list box.
- A window containing the listings.
- Vertical and horizontal scroll bars, as needed. The scroll bars enable users to view the listings.

List boxes enable users to select from an existing list of items that is either long or variable in length.

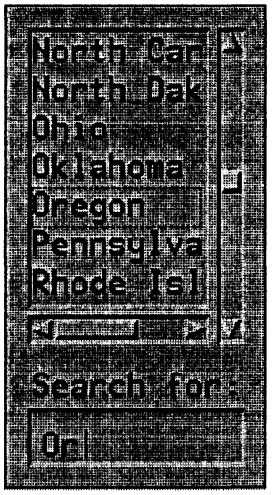


Figure 21. A Typical List Box.

Users move the slider on the scroll bar to change their current view of the list. To choose a selection, users position the mouse pointer on the selection and click the Select mouse button or position the location cursor on the selection and press <*Select*>. The standard methods for range and additional non-contiguous selections are also supported.

List boxes do not support mnemonics. Instead, they have a speed-search function that works as follows: When users press the first letter of an item in a list box, the box scrolls to the first occurrence of an item that begins with that letter. For example, if the list box contained an alphabetical listing of the states in the United States, a user would press "O" to view the states beginning with Ohio.

A list box can also be combined with an entry box to provide an incremental search function. For example, if the list box contains an alphabetical listing of the states in the United States (see Figure 21), you press "O" to view the states beginning with Ohio. Then, to select "Oregon", you press "OR."

Double-clicking the Select mouse button in a list box chooses a selection and activates the default push button in the dialog box.

Entry Boxes

An **entry box** consists of the following parts:

- A title or label.
- The box in which text is entered.

Entry boxes enable users to enter text. The entry box may scroll horizontally if the text entered is longer than the box. The entry box may also be more than one line high, in which case it can have a vertical scroll bar like a list box.

The title describes what is to be entered in the entry box. Titles generally appear above or to the left of the box (in western countries).

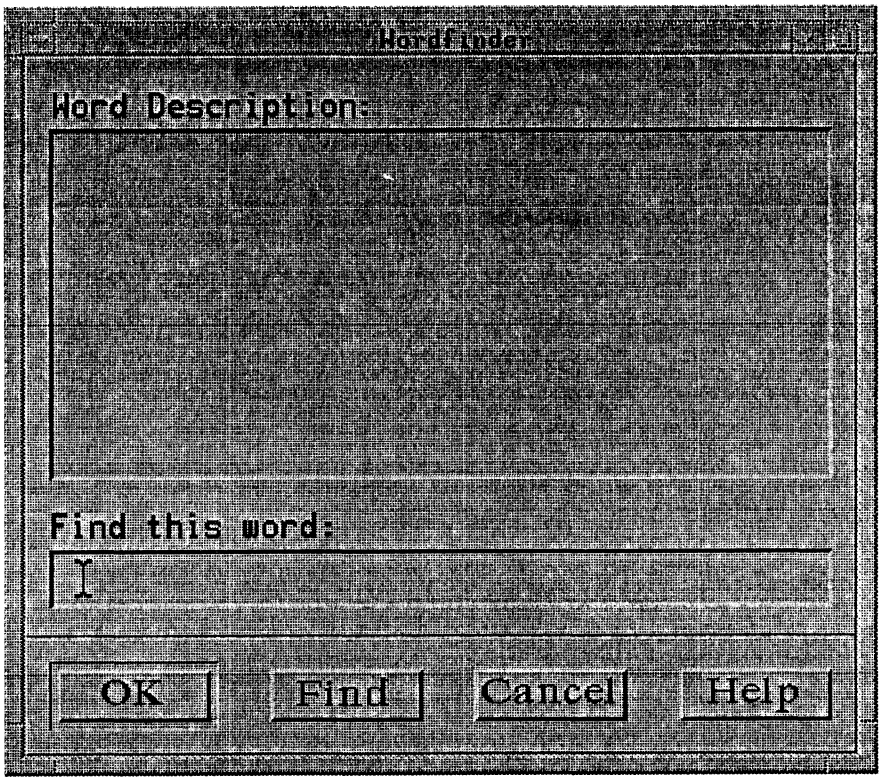


Figure 22. A Typical Entry Box.

When users move the mouse pointer into an entry box, the mouse pointer changes from the default shape to the shape of the text insertion cursor. Entry boxes follow the rules for basic text editing.

Pending Delete

Entry boxes include a function known as "pending delete." Using this function, users can select a range of text to be overwritten in an entry box and then simply begin typing the replacement text. The selected text is deleted and the new text is inserted in its place as typed.

Text Cursor Shapes

A text insertion cursor shows where text will be inserted or overstruck. Insertion cursors should always provide users with a visual cue to the current text mode, insert or overstrike.

In insert mode, the cursor typically appears as a vertical bar or pipe (|) between two text characters. When users press a new character, the character appears to the left of the cursor. The cursor moves one character space to the right. In an inactive window or entry box the insertion cursor is de-emphasized.

In overstrike mode, the cursor typically appears as a block or underline located at the text character that will be replaced. When users press a new character, the cursor replaces the existing character with the new character and moves to the right to the next character. In an inactive window or entry box the block is de-emphasized.

Figure 23 shows typical text cursor shapes.

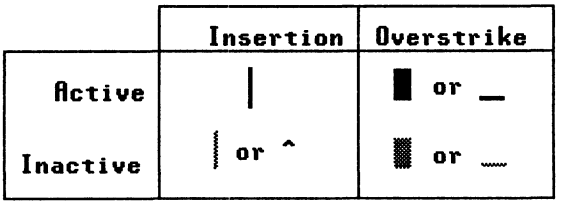


Figure 23. Insertion Cursor Shapes.

Your application should allow users to control whether insertion cursor blinks or not.

While several insertion cursors may appear on the workspace, only *one* cursor, the one in the window with the input focus, can be active; all other insertion cursors are inactive and have a de-emphasized shape.

An insertion cursor can change size and should be set to the height of the current font.

Pre-formatting Entry Areas

Where possible, the entry boxes of your application should be pre-formatted. Typical instances are entry boxes for supplying phone numbers or social security numbers. Pre-formatted entry boxes increase the uniformity of the data entry while easing the burden of remembering and correctly typing formats.

When the text entered in an entry box is all the same length (for example, phone numbers or social security numbers), you can implement **auto tabbing**. Auto tabbing speeds data entry in fixed-length fields by automatically moving the cursor to the next field as users finish making an entry in the current field. This saves moving the mouse or pressing the navigation keys.

Types of Valuators

The OSF/Motif includes several types of valuators that enable you to provide users with analog-style controls.

Using a Scale

Your application could use a **scale** valuator. A scale enables users to enter a value from a range of values by adjusting in analog fashion a sliding arrow to a specific position along a line.

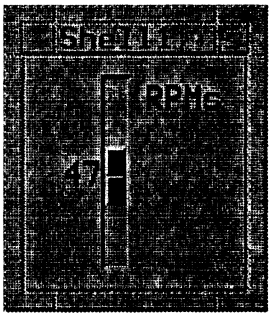


Figure 24. A Typical Scale.

A scale consists of the following components:

Scale bar	The scale bar may contain tick marks and represents the range of available values.
Slider	The slide arrow marks the currently chosen scale value.
Digital readout	The digital readout is an optional number directly opposite the slider that is the digital representation of the currently chosen analog scale value.

Using a Scroll Bar

People use scroll bars to scan rapidly through the contents of a window or to choose from a continuously variable set of values such as color intensity. The current location or setting of the scroll bar is shown by the position of the slider in the scroll bar for text windows or by example for variable choices such as color intensity.

If users try to scroll beyond the end of the text, nothing should happen.

Scroll Bar Components

Scroll bars have the following components:

Scroll region	The scroll region is the "background" of the scroll bar and represents visually the length of the area that users can scroll.
Slider	The slider represents the window through which users look at the displayed data. Put another way, the position of the slider box on the scroll region provides a visual cue that marks the location of users' viewpoint in relation to the total scrollable area.
Stepper arrows	The stepper arrows enable users to scroll incrementally through the data and provide a visual cue to the direction of the scrolling movement.

Your application can use either horizontal or vertical scroll bars or both. The slider moves back and forth in the scroll region showing the position of the currently displayed section relative to the entire contents.

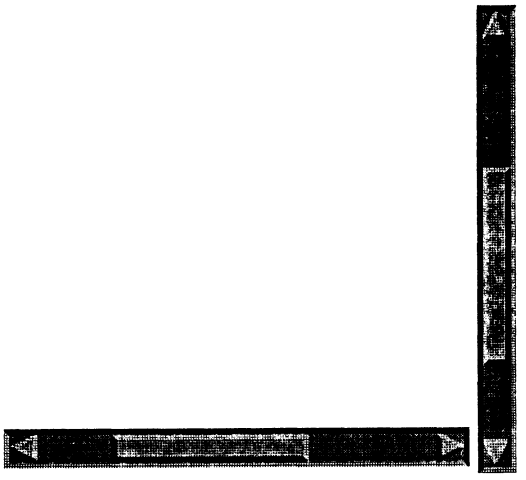


Figure 25. Scroll Bars Can Be Either Vertical or Horizontal.

Operating Scroll Bars

Viewing text or graphical information through a window is like viewing the stars through binoculars. To change the view of the sky, users move the binoculars, not the stars. When the binoculars move up, the stars appear to move down; whichever direction the binoculars move, the stars appear to move in the opposite direction. Similarly, when people use a scroll bar to view a file, the file appears to move in the direction opposite to the movement of the slider. For example, in a text window, if the slider of a vertical scroll bar moves up, a text display seems to move down as previous lines in the file appear at the top of the window.

Table 4 discusses the different ways users can operate a scroll bar.

TABLE 4. Operating Scroll Bars.

Action	Description
Performing a selection operation on a stepper arrow.	Highlights the stepper arrow and moves the window through the underlying file by a single unit, in the direction indicated by the arrow.
Performing a press selection operation on a stepper arrow.	Highlights the stepper arrow and causes a continuous scroll, in unit steps, in the direction indicated by the arrow.
Performing a selection operation on the scroll region.	Moves the window through the underlying file by one window length minus one unit for overlap.
Performing a press selection operation on the scroll region.	Continuously moves the window through the underlying file by one window length minus one unit for overlap.
Performing a drag selection operation on the slider.	Moves the slider and continuously moves the window to a location consistent with the new slider location.

Automatic Scrolling

When users drag the mouse pointer (with a mouse button pressed) beyond the top or bottom of the window, your application should continue the selection by scrolling in the direction of the mouse pointer. This automatic scrolling operates at the same speed as when users press on the directional arrows. The automatic scrolling ends when users move the mouse pointer back into the window or release the Select mouse button.

Slider Size

The slider itself may vary in size to represent the proportion of the entire contents currently covered by the window.

Scroll bars are usually located to the right or on the bottom of the area to be scrolled.

Application Extras

Besides the controls supplied by the OSF/Motif toolkit, other dialog box controls can be supplied by the application as needed. One such application extra is the stepper button. A **stepper button** typically consists of the following parts:

- A title or label.
- A group box or panel separating the stepper button visually from the other parts of the dialog box.
- A text box displaying the current value of the stepper button.
- A scroll bar for stepping through the list of values to find a new value.

A stepper button enables users to select a value by scrolling through a circular list of possible values. The stepper button is similar in operation to the digital read-out of a stereo tuner where pressing the button steps

the read-out through the available radio stations.

The values that read out as people use the stepper button can be either letters or numbers, but they should be in consecutive order, either alphabetical or numerical, as opposed to random order. Users should be able to anticipate the appearance of values.

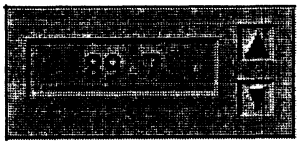


Figure 26. A Typical Stepper Button.

Combining Controls

Entry boxes and list boxes are often used in combination to provide users with particular capabilities. Some common examples are the following:

incremental searching	As users type each letter of an entry in the entry box, the list box moves to the part of the list that matches what has been typed so far. When the desired item appears in the list box, users click the Select mouse button to select the item. The list box can still be used in the normal way.
automatic entry	Users scan the items in the list box and click the Select mouse button to put the current item in the entry box. The entry box can still be used in the normal way.

List boxes used in incremental searches or in automatic entry must be single selection.

Chapter 6
Designing Menus

A **menu** typically consists of a title and a list of selections. Similar to restaurant menus, menus in the OSF/Motif environment display a list of selections from which users choose an appropriate action. Menus provide users with a simple means to quickly access the functions in your application. This chapter discusses the following aspects of menus:

- What types of menus there are.
- What components make up menus.
- How users operate menus.
- What are the standard OSF/Motif menus.
- How to design OSF/Motif menu extensions.

What Types of Menus There Are

The OSF/Motif environment has the following four types of menus:

Pull downs	Menus that pull down from a fixed location in the menu bar.
Pop ups	Menus that pop up at the current pointer location, wherever that may be.
Cascading	Menus that cascade to the right from another menu, providing more detailed selections related to the original menu selection.
Option	Menus that display from a an option button in a window.

Pull-down Menus Are Always Available

In most cases, pull-down menus provide an important part of the communication between users and application programs.

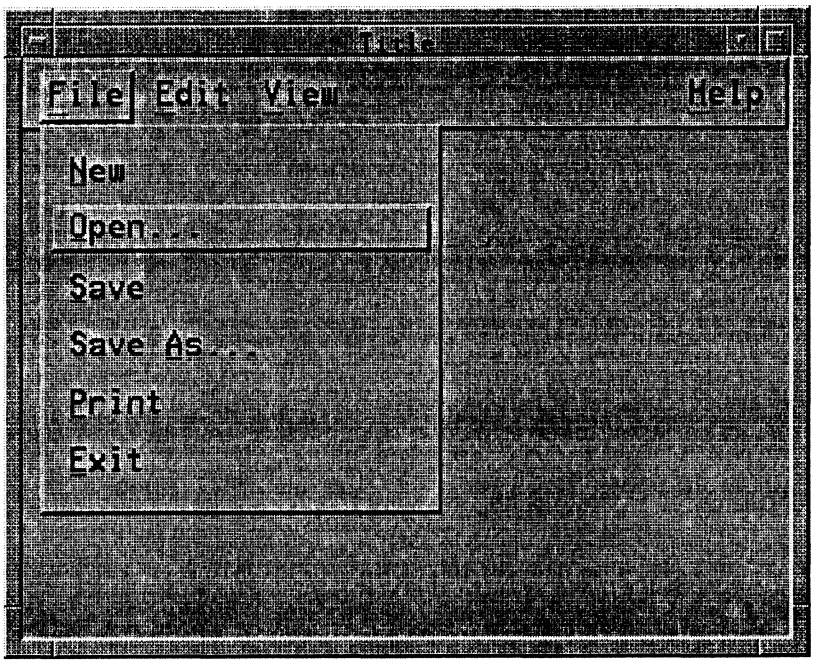


Figure 27. Pull-down Menus Provide Easy Access to Functionality.

Pull-down menus, with the exception of the window menu, are always associated with an application's menu bar. The result is that, if you design your application to include a menu bar, the titles of available pull-down menus are always visible to the users running your application. Thus, the major portion of the functionality of your program is only a point-and-click (or point-and-drag) away from users' fingertips.

Additionally, users soon learn that they can drag the pointer across the titles in the menu bar, displaying each pull-down menu in turn, thus providing a handy table of contents to your application's functionality. This is another way of empowering users because it enables them to browse your program's functionality, refreshing their memories, rather than forcing them to remember the arcane command-line syntax of a particular function.

Option Menus

Windows which include selections from a list, but where space is at a premium are candidates for option menus. Only the option button is usually visible. The button shows the current value of the control. When users select the option button, the menu appears, showing the list of choices.

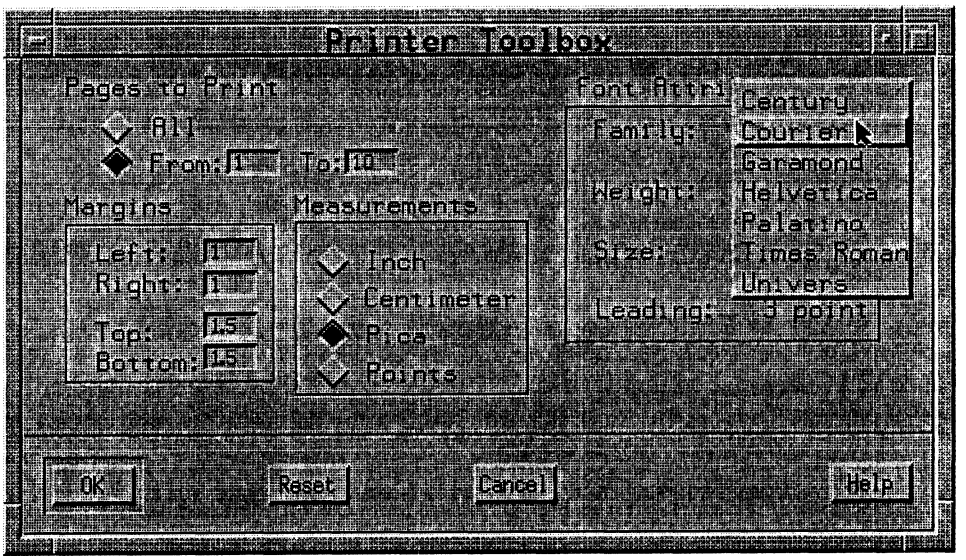


Figure 28. A Typical Option Menu.

Pop-Up Menus Save Space

Pop-up menus (also called "context menus") have the advantage that they take up no permanent screen space. Not being associated with a menu bar, they simply pop up at the current pointer location. A workspace menu is an example of a pop-up menu.

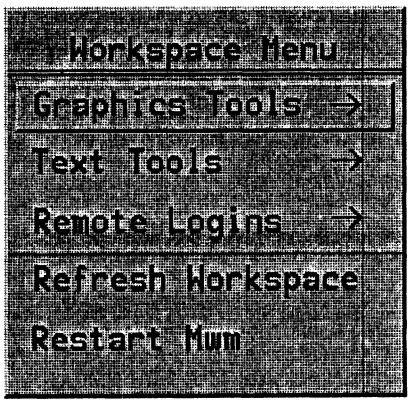


Figure 29. Pop-up Menus Save Space and Mouse Travel.

A second advantage of the pop-up menu is that it requires a minimum of mouse movement. To display a pull-down menu, users must position the pointer on a title somewhere in the menu bar; to display a pop-up

menu, users need only press the Menu mouse button.

Pop-up menus are related to the context of the area in which they are selected. For example, a workspace menu that pops up when users position the pointer over the workspace and click the Menu mouse button may be associated with system-wide functions.

However, pop-up windows, by their very nature, do not provide a visual cue to their availability. Users must learn *and remember* that a pop-up menu is associated with a certain area. Hence, your application design should not use pop-up menus too casually.

Cascading Menus Provide Further Selection Detail

Cascading menus add detail to pull-down and pop-up menus. You can think of them as submenus or child menus of other menus. Cascading menus provide you with a mechanism to organize menu selections in a tree structure, thus simplifying the presentation of complex selection lists. To maintain ease of use, the menu section tree should be no more than three levels deep. A cascading menu appears when users select or drag the pointer onto or across its title on the parent menu.

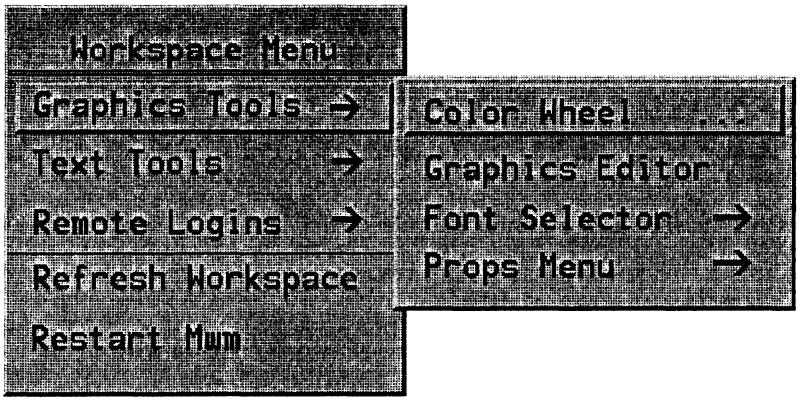


Figure 30. Cascading Menus in Two Tiers.

Cascading menus typically appear to the right of their parent menu selection (in countries where users read from left to right). While cascading menus differ from the two other types of menus in the method of their appearance, cascading menus behave just like pull-down and pop-up menus as far as the choosing of a selection.

What Components Make Up Menus

All menus, regardless of type, have the same components.

Menus Have Titles

Menus have titles that name them. A menu's title should be unique to eliminate the possibility of confusion. The title should clearly indicate the purpose of the menu.

The title of a pull-down menu is on permanent display in the menu bar. The optional title of a pop-up menu displays at the top of the menu. The title of a cascading menu displays as a selection in the parent menu.

The titles of pull-down menus that appear on the menu bar employ single-character mnemonics as memory aids to increase the efficiency of more experienced users. Mnemonics are explained later in this section.

Menu titles are visually distinct (that is visually separated in some way) from the menu's selections. Typically, this is accomplished by placing a separator line below the title.

Menus Have Selections

Menu selections are listed below the menu title and, like the titles of pull-down menus, can also employ mnemonics. Additionally, a menu selection lists any keyboard accelerator associated with the selection. Selections can be text or graphics. Selections can also be grouped with a separator to provide a visual cue of similarity or related functionality.

OSF/Motif menu selections can be one of three types. Figure 31 shows a sample menu containing the three selection types.

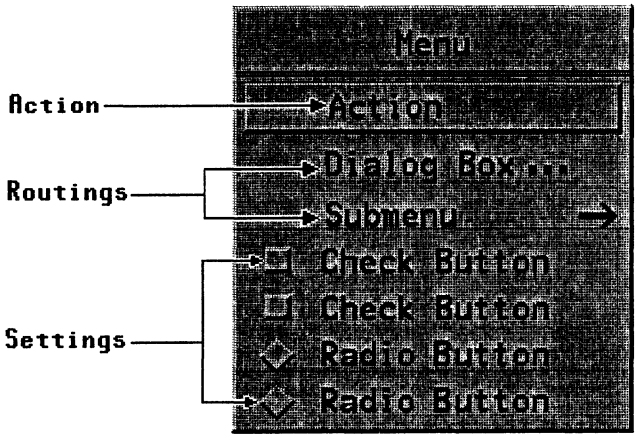


Figure 31. Menu Selections Have Three types.

TABLE 5. Menu Selection Types

Type	Description
Actions	Issue a command or carry out an action.
Routings	Display a dialog box (dialog menu items are indicated by an ellipsis (. . arrow (→)).
Settings	Set an application state using check buttons (multiple selections) or radio buttons (mutually exclusive selections).

Menu selections that are currently not available (disabled) are visually de-emphasized.

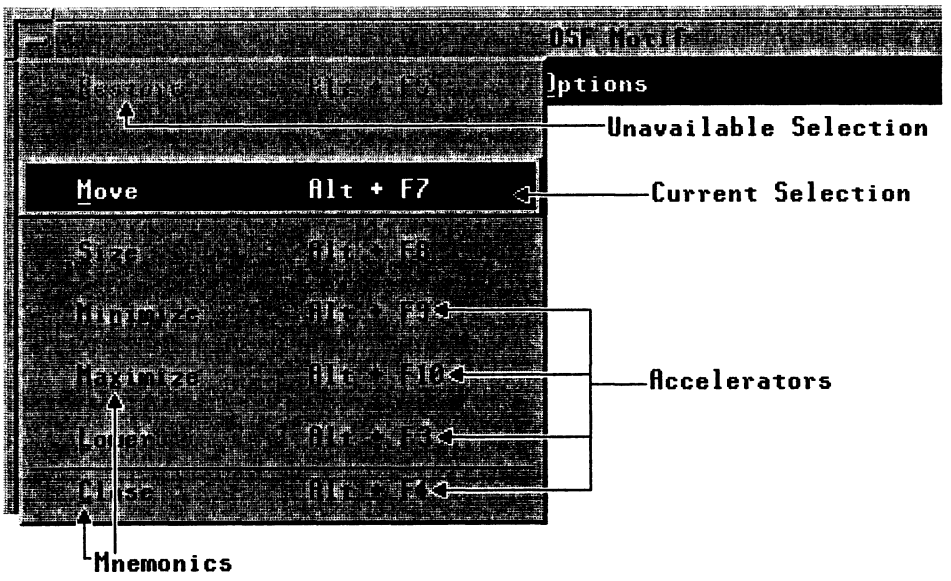


Figure 32. The Anatomy of a Menu.

Menus Have Mnemonics

A **mnemonic** is a single character that provides a shortcut for making selection from the keyboard. Users press the mnemonic for a selection rather than using the navigation and select keys. In the OSF/Motif environment, all menus (and the titles of pull-down menus on the menu bar) have mnemonics associated with their selections. OSF/Motif menu mnemonics are a single letter, usually the initial letter of the selection. Typing the mnemonic for a selection chooses that selection, the same as if the selection were chosen with a mouse operation. If the title of a cascading menu is selected, the menu is displayed. To display a pulldown menu while the location cursor is outside the menu bar, press <*Alt*>+mnemonic.

Typically, an underline visually designates a letter as the mnemonic for a selection, as in Figure 32 above. When an initial letter cannot be used, as in the case where two selections begin with the same letter, some other letter in the selection name, preferably with some mnemonic value, should be chosen instead. If the mnemonic letter does not appear in the selection's text, it appears in parentheses after the text.

Mnemonics are only accessible when the menu containing them is displayed. The pull-down menus in a menu bar are always accessible (by pressing *<Alt>+mnemonic*) since the menu bar is always displayed.

Menus Have Keyboard Accelerators

Menu selections can also have **keyboard accelerators**, a key or key sequence that invokes a menu selection *without* displaying the menu. Keyboard accelerators do not have to be associated with each and every menu selection. In most cases, it is sufficient to provide accelerators only for frequently used functions. Providing accelerators should be a matter of utility, not design conformity.

If a keyboard accelerator exists for a menu selection, it appears right justified on the same line as and separated from the selection's text by enough space to make it visually distinct. Other than this, keyboard accelerators provide no visual cue to their existence and so users must memorize them. This is why frequently used functions make the best candidates for accelerators.

How Users Operate Menus

Users operate menus in two steps: first, they display the menu; second, they choose a selection. When a selection has been chosen (unless it leads to a cascading menu), the menu disappears.

Displaying Menus

In the case of pull-down menus, mouse users move the pointer to the title of the menu and press the Select mouse button. Keyboard users move the location cursor to the menu bar by typing *<F10>*, then press the pull down menu's mnemonic or use arrow keys to select the menu's title. An alternate method of displaying a pull down menu is to press the *<Alt>* modifier key and press the menu mnemonic.

In the case of pop-up menus, mouse users move the pointer over the pop-up area (for example, the workspace) and press the Menu mouse button. Keyboard users press the *<Menu>* key to display a pop-up menu.

In the case of cascading menus, mouse users move the pointer to the title of the cascading menu on the parent menu and click (or continue to press) the Select mouse button. Keyboard users use the navigation keys and *<Select>* or *<Enter>* to achieve the same effect as the mouse user's actions.

Additionally, users can access application menus using either of the following two methods:

Dragging	Users can drag the menu, pressing *and holding down* the mouse button to maintain the display. They release the mouse button to make their choice.
Clicking	Users can click the appropriate mouse button or press the appropriate key to display the menu. The menu remains displayed until either a selection is made or the display is canceled. A selection is made by moving the pointer to the selection and clicking the mouse button or by moving the location cursor to the selection and typing the *<Select>* or *<Enter>* key.

Browsing the Menu Bar

Users can browse the menus listed on the menu bar by pressing the Select mouse button and dragging the pointer across the menu titles. As the pointer crosses each title, the menu associated with the title pulls down. To browse the menu bar from the keyboard, users display a menu and use the left and right arrow

keys to move laterally across the bar.

Choosing Menu Selections

To choose a menu selection, users position the pointer or location cursor on the selection and make a selection action.

People using the drag method drag the pointer onto the selection they desire and release the mouse button they pressed to initiate the drag process.

Mouse users using the click method slide the pointer onto the selection and click the appropriate mouse button. Keyboard users using the click method use the navigation keys to position the location cursor on the selection they desire and press the <*Select*> key.

A menu item that has been selected provides a visual cue of its selection. Typically, this cue is a change in color, either highlighting or reversed video. In a 3-D implementation, the current selection not only changes color, it also has a raised, 3-D appearance.

Releasing the mouse button or keyboard key selects the item under the mouse pointer or location cursor. If the release occurs on a command (action), check button, or radio button, the specified action takes place, and any menus displayed disappear. If the release occurs on the title of a submenu, the menu is displayed.

Avoiding Menu Selection

To avoid making a menu selection, mouse users dragging the menu drag the pointer off the menu and release the mouse button. Mouse users clicking the menu slide the pointer off the menu and click the Select button. Keyboard users press the <*Esc*> key.

The Standard OSF/Motif Menus

The following five pull-down menus provide general functions common to most applications. They are a part of most menu bars in the OSF/Motif environment:

- File
- Edit
- View
- Options
- Help

Figure 33. A Typical Motif Menu Bar.

The File and Edit menus have recommended contents. The View and Option menus are highly specific to each application, so only sample implementations are provided. This chapter also contains a sample Help menu. You can create additional menus for your application. The File and Edit menus (if used) should be the first two menus in your action bar. The Help menu is always the last.

While it is recommended that you include the standard menus in the menu bar of your application, your choice of menu titles and items depends on the nature of your application. Should your application require it, you should design more relevant titles and selections for your application's menu bar and menus, but do not change the meanings of words used in the standard menus.

A Look at the File Menu

The File menu presents actions that deal with files in their entirety. All applications that deal with files should provide the File menu.

The selections in the menus are divided into four groups:

- Selecting actions that connect files to your application.
- Saving actions that transfer changes made to a storage medium.
- Output actions that send the changed file to an output device.
- Other actions.

These groups are important as you design extensions to the menu. Suppose you wanted to add an action called Plot to your application. You would place it in the File menu because it deals with the file as a whole. Within the File menu, you would place it with Print in the output group because Plot writes the entire file to an output device.

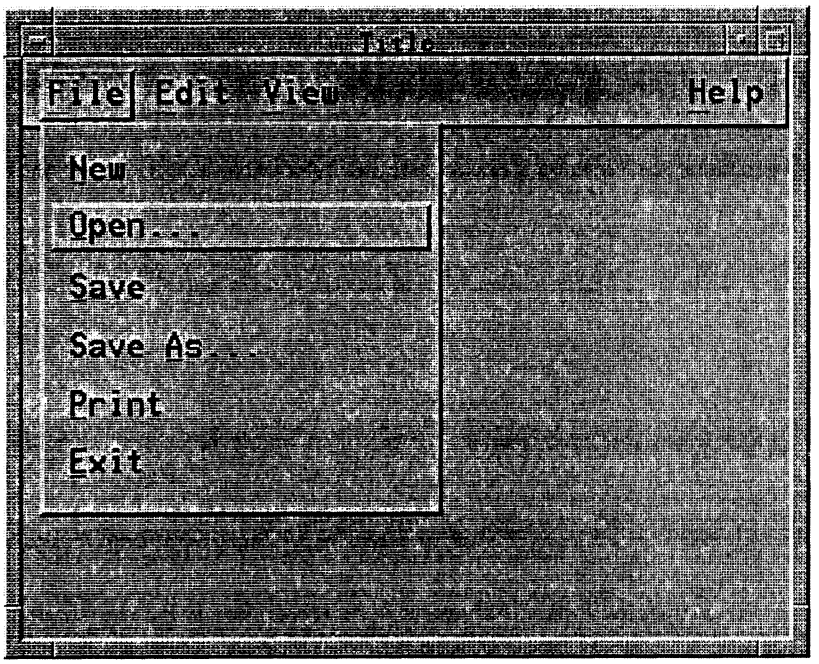


Figure 34. The File Menu And Its Selections.

For consistent operation with other OSF/Motif applications, if you include the File menu in the menu bar of your application, it should appear as the first title, placed on the far left, and with "F" as its mnemonic.

The File menu contains the following selections:

New	Creates a new file. The New operation clears existing data from the client area and replaces the current filename in the title bar with "Untitled" or some other application-generated name. If completion of the operation will obliterate current changes to the file, you must display a message box asking users whether they want to save their changes.
Open . . .	Opens an existing file. The Open operation prompts users for the name of the file by displaying a dialog box for the purpose. The title bar is updated with the name of the newly opened file. If completion of the operation will obliterate current changes to the file, you must display a message box asking users whether they want to save their changes.
Save	Saves the currently opened file to a storage device without removing the existing contents of the client area. If the currently opened file has no name, Save prompts users for a file name by displaying a dialog box for the purpose. Saved along with the file should be the current state of the file, including: window size, window location, and scrolling position within the file. This enables users to reopen the file and begin their work where they left off.
Save As . . .	Saves the currently opened file under a new name. The Save As operation prompts users for the name of the file to be saved by displaying a dialog box for the purpose. If users try to save the new file under an existing name, the Save As operation displays a warning message alerting them of a possible loss of data.

Print	Schedules a file for printing. If your application requires specific printing information before printing, the Print . . . selection displays a dialog box in which to gather it.
Exit	Ends the current application and closes all windows associated with it. This action is equivalent to closing all primary windows of the application. If completion of the operation will obliterate current changes to the file, you must display a message box asking users whether they want to save their changes. You are encouraged to include this action even though it duplicates much of the functionality of the Close action in the window menu. This assures users have a way to end the application even if they are not running the Motif window manager. If your application does not have a File menu, put Exit at the end of the first pulldown menu.

A Look at the Edit Menu

The Edit pulldown menu contains actions that modify the contents of the data which which your application is currently dealing. Many of the actions relate to cut and paste functionality.

The standard selections are grouped as follows:

- Undo actions that reverse the effect of previous actions.
- Actions that relate to the system-wide clipboard.
- Other Actions.

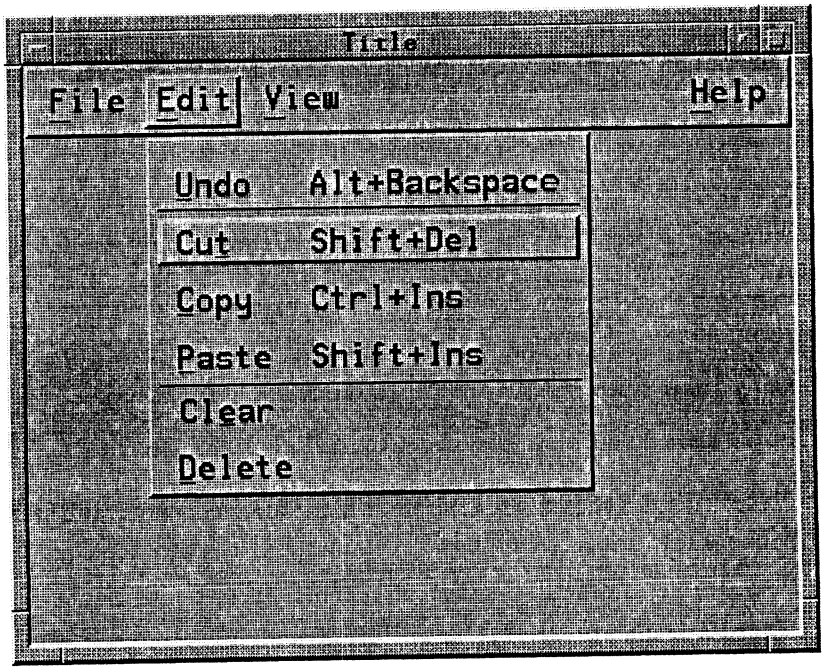


Figure 35. The Edit Menu And its Selections.

For consistent operation with other OSF/Motif applications, if you include the Edit menu in the menu bar of your application, it should appear as the second title from the left and with "E" as its mnemonic. The

Edit menu contains the following selections:

Undo Alt+Backspace	Reverses the most recently executed action. To provide a visual cue to users, the Undo selection should be dynamically modified to indicate what is being undone. If the most recently executed action were a paste, the action name would be Undo paste. Your application should be able to undo all of the actions in the Edit pulldown. Text applications should also support Undo typing which restores text after it has been selected and replaced by newly typed text.
Cut Shift+Del	Removes a selected portion of data from the client area to the clipboard buffer. Your application determines whether the area that used to be occupied by the removed data is left blank or whether the remaining data is compressed to fill in the space. Usually graphics applications leave the space blank while text application compress the remaining text to fill in the space.
Copy Ctrl+Ins	Copies a selected portion of data to the clipboard without removing the original data from the client area.
Paste Shift+Ins	Pastes the contents of the clipboard into a client area at the selected location. Your application determines whether the pasted data is reformatted to fit in the client area and whether existing data moves to create room for the pasted data. Text applications will typically reformat the pasted text to fit into the margins of the text field and they will move the existing text to make room for the new text. Graphics application might do neither. They might insert the graphics unmodified and overlay existing data.
Clear	(Optional) Removes a selected portion of data from the client area without copying it to a clipboard buffer. Clear erases the data, but leaves the space formerly occupied by the data.
Delete	(Optional) Removes a selected portion of data from the client area without copying it to a clipboard buffer. The remaining data is compressed to fill the space formerly occupied by the deleted data.

A Look at a View Menu

A View menu enables users to control how data is displayed. The data itself remains unchanged. Entries in the menu control such aspects of presentation as:

- Appearance of the data. Examples may be iconic or text-based.
- How much of the data is presented.
- In what order the data is sorted.
- To what level the data is summarized.

For consistent operation with other OSF/Motif applications, if you include a View menu in the menu bar of your application, it should have the "V" as its mnemonic.

The content of View menus is very application specific. Figure 36 shows a sample View menu for a hypothetical file browser.

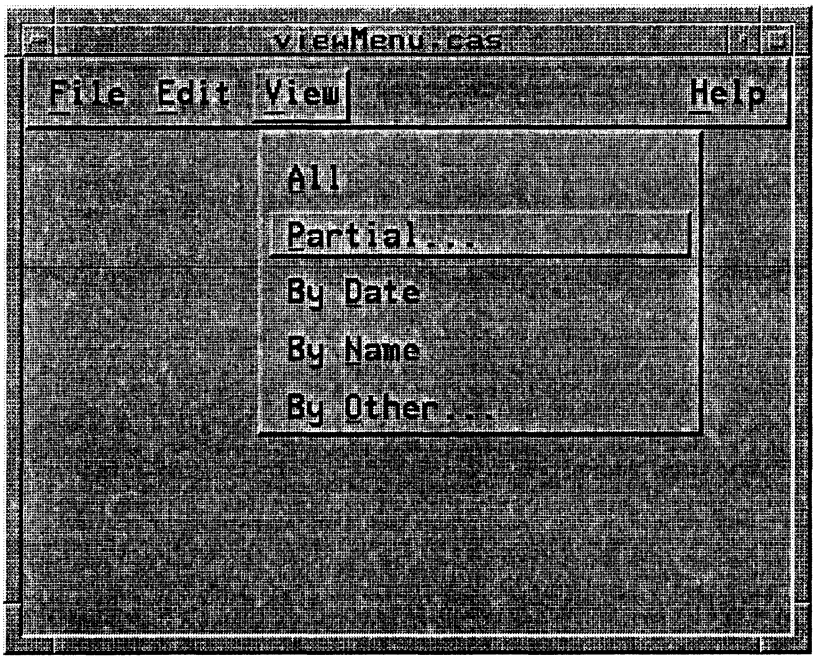


Figure 36. A Typical View Menu and its Selections.

The entries in the sample View menu are:

All	Views an entire list of items.
Partial . . .	Views a partial list of items. The selection criteria are determined by a dialog box.
By Date	Views a list ordered by date of entry.
By Name	Views a list ordered by item name.
By Other . . .	Views a list ordered by selection criteria determined by a dialog box.

Remember that the menu above is only an example. The needs of your application control what entries you place in your View menu.

A Look at an Options Menu

An Options menu enables users to customize various aspects of your application. Just like the View menu, the Options menu's content depends entirely on the needs of your application.

For consistent operation with other OSF/Motif applications, if you include a Options menu in the menu bar of your application, it should have the "O" as its mnemonic.

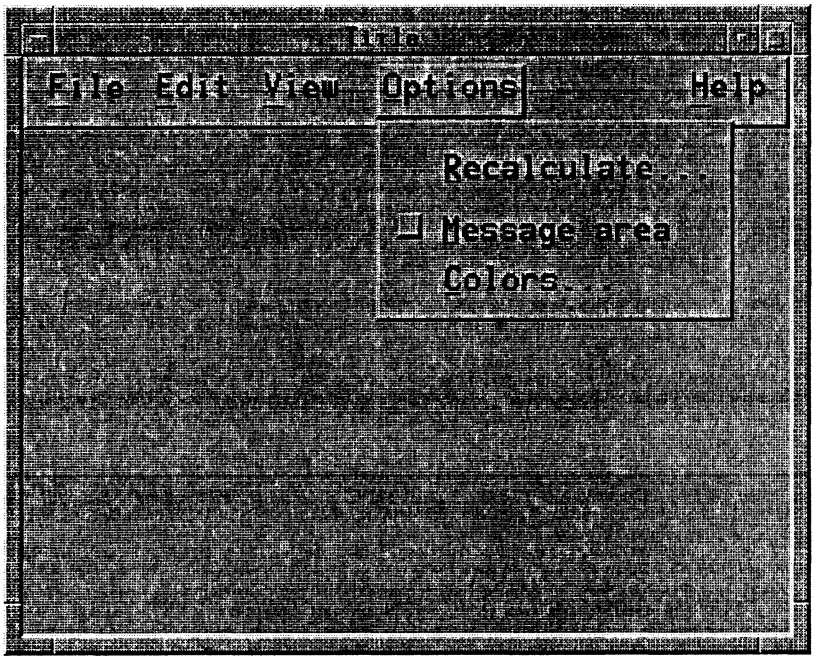


Figure 37. A Typical Options Menu and its Selections.

The content of Options menus is very application specific. Entries in the menu for a hypothetical spreadsheet application might include such entries as:

Recalculate	Displays a dialog box to control when cell formulas are recalculated.
Message area	A check box that controls whether a message area is displayed. The message area shows helpful explanations of commands for novice users.
Colors . . .	Displays a dialog box used to select colors used by the application.

Remember that the menu above is only an example. Select entries appropriate to your application.

A Look at the Help Menu

Good applications provide help facilities for people to use when the need arises. No matter how intuitive you design your application to be, sometimes users get stuck.

For consistent operation with other OSF/Motif applications, if you include a Help menu in the menu bar of your application, it should appear as the last title, placed on the far right of the menu bar, and with "H" as its mnemonic.

Below appears a sample help menu. Note that you should only include those functions actually supported by your application.

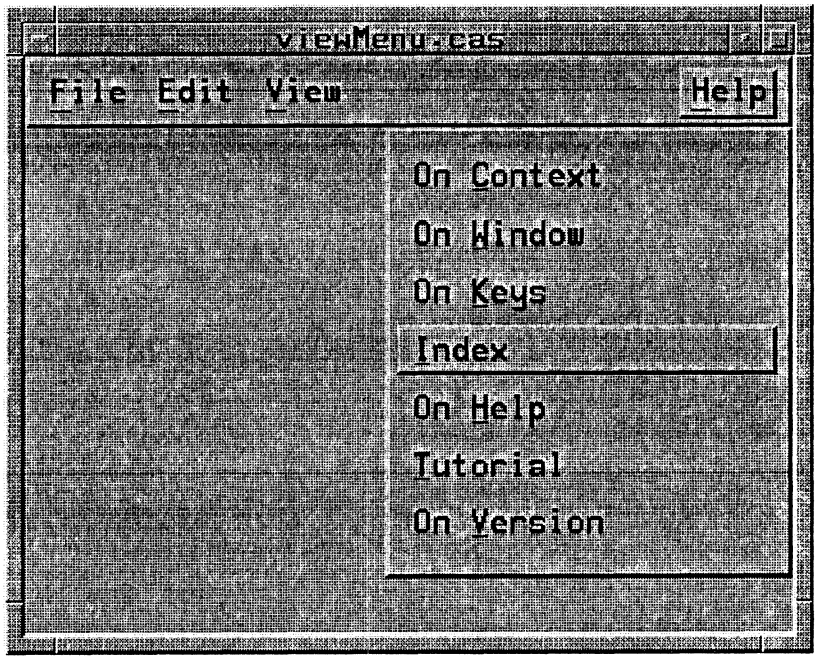


Figure 38. The Help Menu And its Selections.

The sample Help menu contains the following selections:

On Context	Provides context-sensitive help about the specific situation that exists when the help was requested.
On Help	Provides information on how to use the application's help facility.
On Window	Provides general information about the operation of the window from which the help was requested.
On Keys	Provides information about the application's use of function keys, mnemonics, and keyboard accelerators.
Index	Provides an index, with a search capability, for all help information in the application.
Tutorial	Provides access to the application's tutorial if such a tutorial exists.
On Version	Provides the name, version, and date of the application.

Help windows your application displays generally are normal secondary windows whose parent window is the window from which users requested help. The title in the menu bar is the name of the application (or name of the object in object-oriented systems) followed by the word "Help." Help menus may also have a title in the client area that describes the topic for which help is being provided.

How to Design Motif Menus

You will most likely have to design menus specific to your application. The following sections describe some considerations in menu design.

Group Like Menu Selections Together

As a general rule, wherever possible, organize the selections on your menus into logical groups.

However, it's not a good idea to put a destructive command (such as Delete or Quit) next to other frequently chosen selections. This is because of a common problem of menus called the off by one error: Users mistakenly choose the selection next to the one they intended to choose. So be aware of the result if a user mistakenly chooses the menu selection above or below the intended selection.

Use a visual cue such as a separator line to divide menu selections into logical groups.

Sets of related items, such as radio buttons or check buttons, should be located together and separated from other menu items.

List Selections in Order of Frequency

Order menu selections according to the frequency of usage, positioning the most frequently used selections near the top of the menu.

As much as the logical grouping of the menu allows, order the selections of your menu in decreasing frequency of usage. Within logical groups, the same principle applies: List menu selections according to the frequency of usage with the most frequently used selections at the top.

As you order menus, maintain a global perspective. Consistency across your entire application is generally more important than frequency of use in a particular menu. Evaluate frequency of use over the entire environment faced by people who use your application.

Keep Menu Structures Simple

If your application requires submenus, keep the menu structure simple. While it is possible to create menu structures that start from a single pull-down or pop-up menu and cascade down several levels of submenus, it is seldom necessary to do so. The awe of users who see submenu after submenu cascading down the screen quickly turns to consternation over such poor design.

Avoid using many submenus in your application. It is better to use a few more pull-down and pop-up menus, or to have more selections per menu, than to have multiple cascading submenus.

As a general guideline, use as few menu levels as possible with three levels as a maximum. A dialog box is a good alternative to menu complexity. So is redesigning your menu structure to eliminate complexity.

Provide Accelerators and Mnemonics

Keyboard accelerators and mnemonics enable users who have become familiar with your application to take short cuts, increasing their efficiency, while not affecting those users who are still learning the basics.

Mnemonics require the display of a menu but are preferred by some users because they allow those users to operate the mouse with one hand and make the selection with the other (as the mouse hand returns to the keyboard). Mnemonics can be made more memorable by carefully choosing the mnemonic letter.

Keyboard accelerator sequences don't require the display of a menu, hence they are active whenever your application's window is active. They are designed for menu entries that people use very frequently. To make the accelerators for your application more memorable, design your application so that accelerator sequences are consistent and progress logically.

Control Availability of Menu Selections

During use of your application, the application will enter a state where some menu selections will not make sense. In that case, make them unavailable while that state occurs. This avoids excessive message boxes noting a selection error.

Making menu selection unavailable provides a temporary constraint. Menu selections that are *never* applicable to your application should *not* be included in the menu.

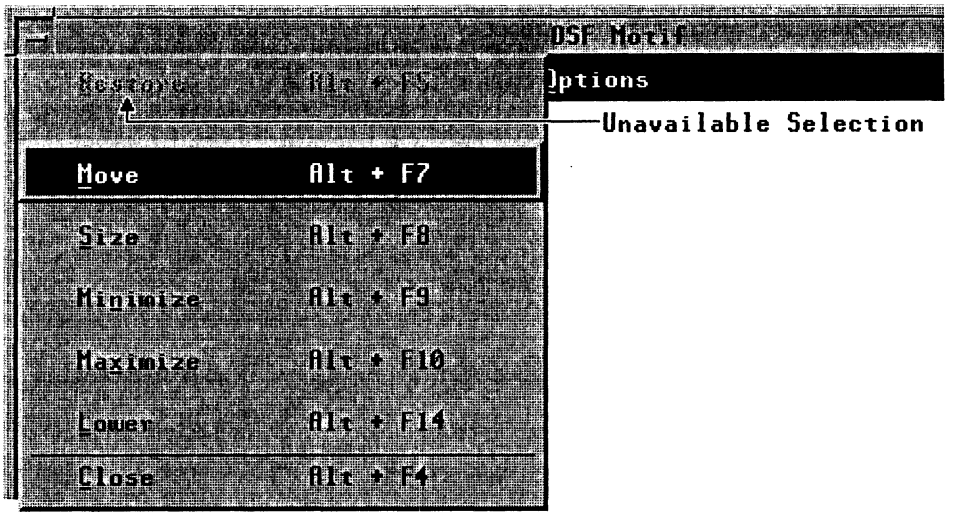


Figure 39. Disabled Menu Selections Give a Visual Cue.

Unavailable menu selections provide a visual cue, such as being dimmed, that they are not functional in the current context. But even if all selections in a menu are disabled, users should still be able to display the menu (but not choose any of the selections) and get help.

Consider Use of Graphic Images

An important option to consider is the use of graphic images (bitmaps) for selections. A good picture can be more readily understood and easily remembered than a line of text. Graphics also ease the task of localizing your application for other countries.

Keep Menu Selections Stable

You should generally keep menu selections the same for the duration of an application's invocation. Settings in menus may be set or unset and any selection may become unavailable, but the selections themselves should not change.

Do not replace menu selections during an application's use. Entries that are *temporarily* unavailable should be disabled and should visually appear as such.

You may want to reword some menu selections slightly in order to better reflect their meaning during the current state of the application. The Undo entry in the Edit menu does this.

Adding menu entries dynamically is discouraged. If your application does require this, however, add them at the end. Number the selections and use the number as the mnemonic. This is the only case where menu selections should be numbered.

The above discussion of dynamic changes in menus applies only to changes made by the application in response to changing conditions in the application. They do not apply to any changes due to user customization.

Allow Users to Customize Menus

Should you choose to, you can further empower the people using your application by allowing them to create their own menu titles and selections and associate them with their own choice of functions.

Chapter 7
Designing Dialog Boxes

A **dialog box** is a window that contains graphical controls that people use to converse with your application. While the OSF/Motif toolkit supplies you with graphical controls for most occasions, you must select the appropriate controls and create the dialog boxes for your application.

This chapter discusses the following subjects:

- The characteristics of dialog boxes.
- Dialog box actions.
- The anatomy of a dialog box.
- Common dialog boxes.
- How users converse with dialog boxes.

The Characteristics of Dialog Boxes

People use dialog boxes to control the operation of your application. From a technical point of view, a dialog box is any window that solicits or displays information or instructions. While dialog boxes, because they enable users to control your application, are similar to menus, they can be much more flexible than menus. It is important that you design dialog boxes with the needs of your application and your users in mind.

The Purpose of Dialog Boxes

Dialog boxes have a variety of purposes. Some display information. These message boxes may be rather plain and relatively simple. Other dialog boxes solicit data from users. These may include elaborate combinations of text and graphics and a variety of controls (entry boxes, list boxes, radio buttons, etc.) through which users can convey information to your application.

You are not restricted to these purposes. In fact, many of your dialog boxes may serve a combination of purposes. However, you will probably notice that many of your dialog boxes require the same or similar actions. The *Dialog Box Actions* section of this chapter describes standard dialog actions. Your application should take advantage of these when appropriate.

Ending a Dialog

Many dialog boxes disappear after users have acknowledged the information, provided the information requested, or selected an action. This is another area in which dialog boxes are similar to menus. Unlike menus however, you may want a particular dialog box to remain visible after a selection. This enables users to continue their dialog without having to redisplay the dialog box. The "Find String" dialog box of some editor programs is an example. It remains visible until users explicitly close it.

Making Controls Unavailable

As people use your application, the operational context that develops may make the use of certain controls inappropriate. For example, the use of the Minimize selection in a window menu is inappropriate when the window is already minimized. In such cases, you should make the inappropriate controls unavailable. This is also called disabling the controls. Unavailable controls are visually de-emphasized. The unavailable controls can be in any window of your application. While unavailable controls do not operate, users should still be able to get help on them.

If a control can *never* be used, that control should not appear at all. Applications that have various authorization levels, for instance, should only show those controls that users are authorized to use.

Some dialog boxes require users to complete the interaction with them before the application continues. In essence, these dialog boxes temporarily disable *all* other controls in the application. This type of dialog box is sometimes called **application modal**. Others take over the entire display and make all other windows on the screen unavailable. These windows are called **system-modal**. Message boxes that require users to perform some immediate action before processing can continue are examples of modal windows. Use the disabling feature sparingly, since it compromises the basic premise of the OSF/Motif interface: empowering the user.

Dialog Box Actions

Most often dialog boxes present information or solicit data. Typically, they also provide several actions from which users select the one desired.

While on occasion your application may require special dialog box actions, most of your dialog boxes will share common actions. The OSF/Motif user interface has identified actions that are likely to occur in many dialog boxes and has given them common names and definitions. No dialog box will contain all of the common actions listed below. Select the ones appropriate to your application or determine your own actions using the guidelines in this section. The list is ordered in the approximate sequence they appear in dialog boxes.

Action	Performs the action specified by the label. You specify actions that are appropriate to your application.
Yes	Indicates an affirmative response to a question posed in the dialog box and closes the window. See note below.
No	Indicates a negative response to a question posed in the dialog box and closes the window. See note below.
OK	Combines Apply and Close actions (both are described below) in one convenient pushbutton.
Apply	Applies any changes made to controls in the dialog box.
Retry	Causes the task in progress to be attempted again. This action is commonly found in message boxes reporting some sort of hardware error.
Stop	Ends the task in progress at the next possible breaking point. This action is commonly found in progress message boxes.
Text...	Opens another dialog box that extends the current dialog.
Reset	Resets any changes to controls in the dialog box to the values they had when the dialog box was opened.
Cancel	Combines Reset and Close actions in one convenient pushbutton.
Close	Removes the dialog box. Users explicitly close dialog boxes that remain open after each use. This function duplicates the Close action in the window menu of the dialog box. You are

encouraged to include it in your dialog box in case users run your Motif application without the OSF/Motif window manager.

Help Enters the help subsystem.

While Yes and No are not actions, they imply "do" and "don't" and are used in dialog boxes in that context. However, you should be careful to avoid ambiguity. Only if a question is very simple, phrased without negatives, and results in an action that is not damaging should you use Yes and No. Otherwise, use an action.

The Anatomy of a Dialog Box

Dialog boxes are composed of combinations of the controls described in Chapter 5.

Dialog boxes differ from one another, not because their controls differ. Controls *don't* differ in behavior. A push button, for example, is always pushed; a check button is always checked. This is fundamental to common behavior. Rather, dialog boxes differ because your choice of controls and combinations of controls differs depending on the needs of your application.

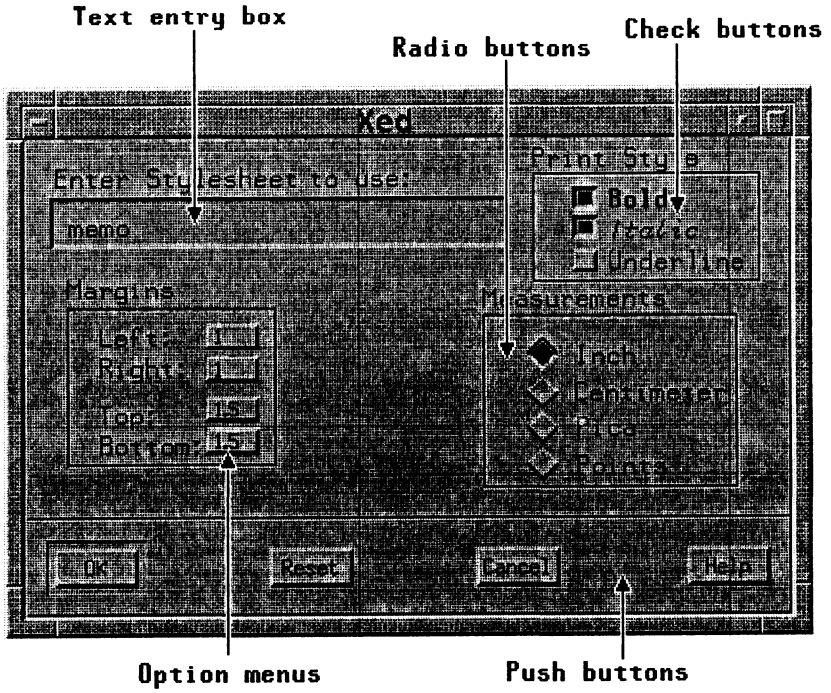


Figure 40. Dialog Boxes Can Contain A Variety of Controls.

While the OSF/Motif Toolkit provides a variety of standard controls, you can create additional controls should the need arise. A stepper button is an example of an application-supplied control.

Arranging Push Buttons in Dialog Boxes

You should arrange push buttons in your dialog box in in an order that supports users' natural progress through the controls in the dialog box. This guideline usually results in pushbuttons either in a row at the bottom of the window or in a column at the right .

Of the two positions, in a row along the bottom of the dialog box is preferable because push buttons are frequently used to end the dialog and thus are the last thing users encounter as they scan the contents of the dialog box.

Push buttons are commonly found in the following combinations and sequences:

- Close
- OK Cancel Help
- OK Reset Cancel Help
- Apply Reset Close Help
- Yes No Help
- Retry Stop Help

In this order, the positive (confirming) selections are toward the left, the selections negating change are toward the right, and Help is consistently placed as the right most push button. If you choose to arrange push buttons in a column, the positive selections should be toward the top with Help on the bottom.

Default Push Buttons

A **default push button** enables users to easily select the most likely response to a dialog box query. The default push button is always visually distinguishable from other push button selections. The OK push button is frequently the default push button in dialog boxes.

If possible, the action performed by the default push button should be reversible. If an action is potentially destructive (for example, if loss of data could occur), its button should not be made the default push button.

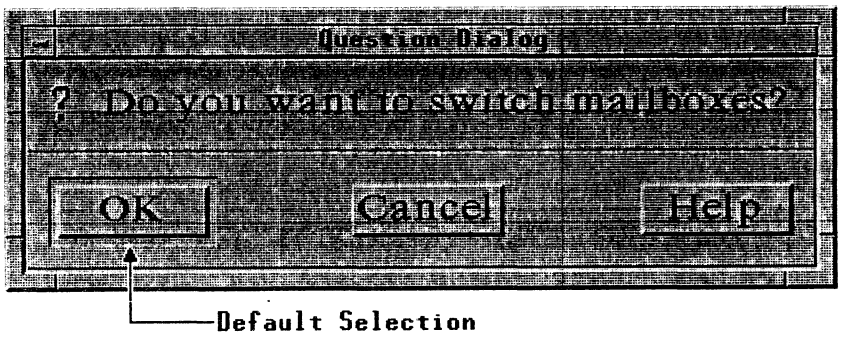


Figure 41. The Default Push Button Is Always Visually Distinct.

Users can start the default push button action in either of two ways:

- By double-clicking when making a selection in the dialog box.

- By pressing *<Enter>* after making a selection in the dialog box.

When people use the keyboard to navigate through the push buttons, the button with the location cursor always becomes the default push button. This ensures that pressing *<Enter>* over a push button invokes that push button. When the location cursor leaves the push buttons, the original default button once again becomes the default.

Message Dialog Box Types

Some dialog boxes provide users with a message and ask for an acknowledgement or response or some sort. These dialog boxes, called message boxes, are not requested by users, but are displayed by application as a result of some event. They can be divided into five general types depending on the nature of their message: information, progress, question, warning, and action.

Information

Some dialog boxes simply convey information. They inform users. They do not interrupt any tasks. For example, they may display the fact that a user has newly arrived electronic mail. When users acknowledge the information, the information box goes away. Figure 42 illustrates a typical dialog box that conveys information.

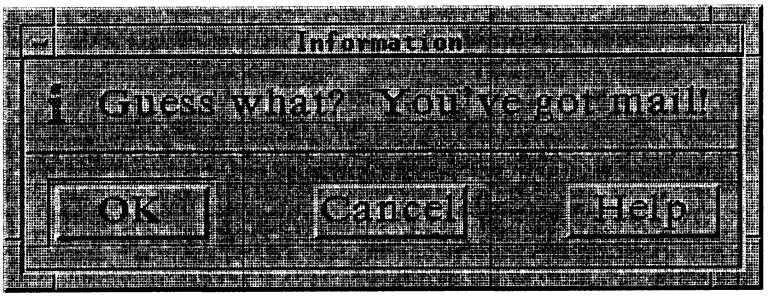


Figure 42. A Typical Information Box.

Progress

Other dialog boxes convey current progress. Their message tells users of the progress of an operation and allows users to choose between continuing the operation or canceling it. Figure 43 illustrates a typical dialog box that conveys the status of work in progress. You may want to show in your application's progress box how much of the total work has been completed.

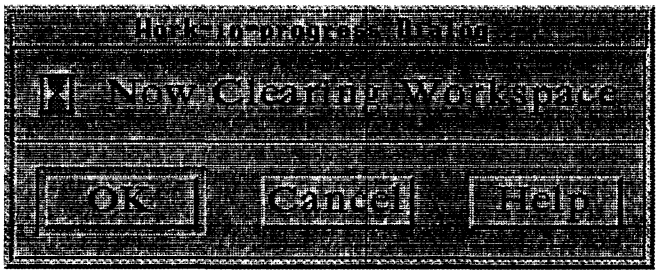


Figure 43. A Typical Progress Box.

Question

Some dialog boxes clarify a previous response by asking a question. The work does not proceed until the question has been answered. The question briefly explains the situation and is phrased so that the reply is a choice between mutually exclusive alternatives. Figure 44 illustrates a typical dialog box that asks a question.

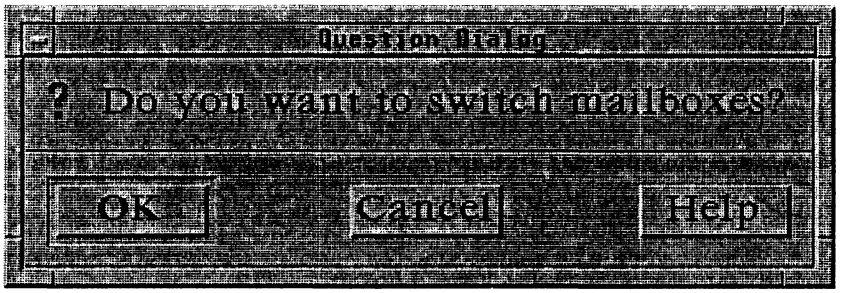


Figure 44. A Typical Question Box.

Warning

Other dialog boxes convey a warning. They alert users to some eminent danger (for example, the potential loss of data) and allow users to choose between ignoring the warning and continuing the operation or heeding the warning and canceling the operation. Figure 45 illustrates a typical dialog box that warns of possible danger.

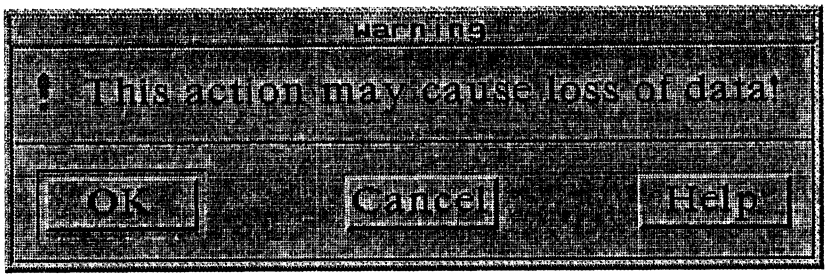


Figure 45. A Typical Warning Box.

Action

Still other dialog boxes convey a message that requires immediate action. They alert users that some action is required (or that some condition exists that requires action) by constraining the operation of the application until the action has been completed. Figure 46 illustrates a typical dialog box that conveys an action message.

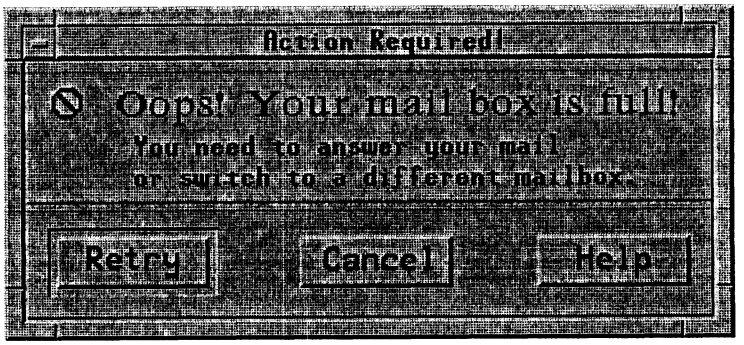


Figure 46. A Typical Action Message Box.

Common Dialog Boxes

Some dialog boxes perform an operation based on supplied input. Usually, these dialog boxes are application specific, but some operations are common to a wide variety of applications. Examples include operations to save a file, enter a command, and make a selection.

File Selection

The File Selection dialog box is used by users to enter the name of a file which they want to save. Figure 47 illustrates a sample File Selection dialog box. The box is most commonly used in conjunction with the common File menu described in the previous chapter.

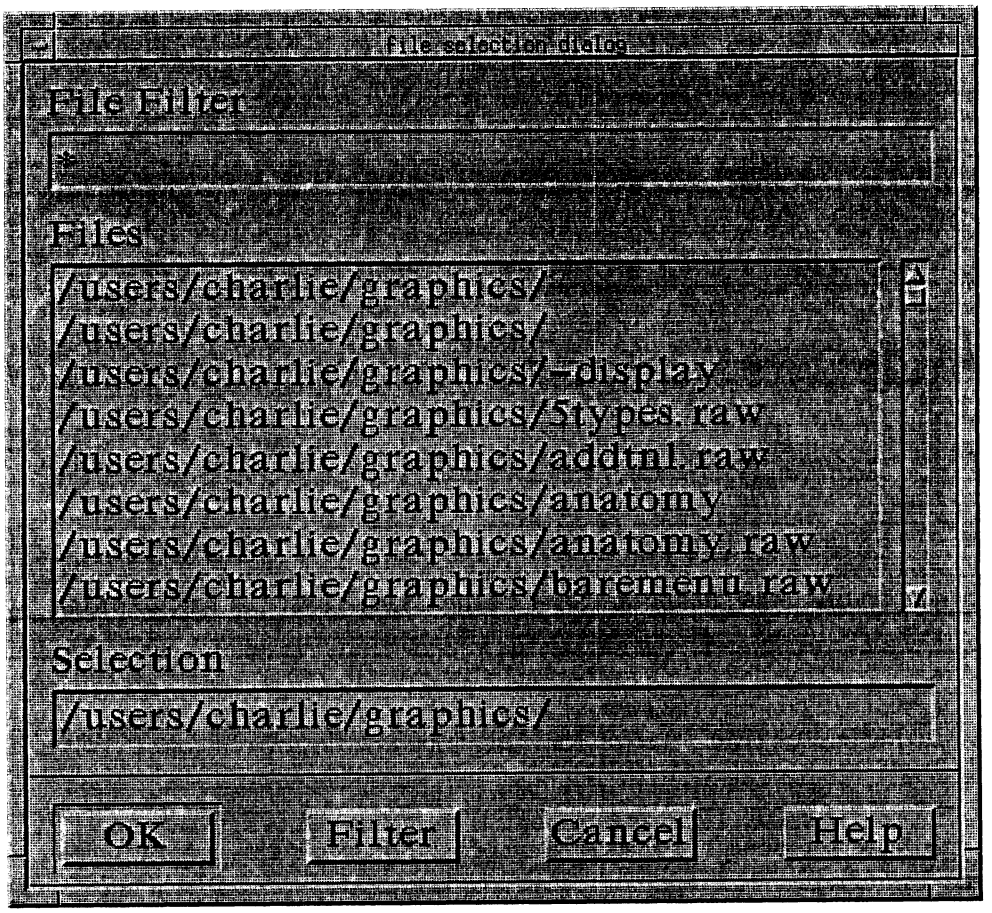


Figure 47. A File Selection Dialog Box.

Command

The Command dialog box is used by people to enter a command either to the application or to the computer operating system. The Command dialog box provides the same function as a command line prompt and provides a history list from which previous commands can be reselected. Your application should use either the command line or the command dialog box. Figure 48 illustrates a typical Command dialog box.

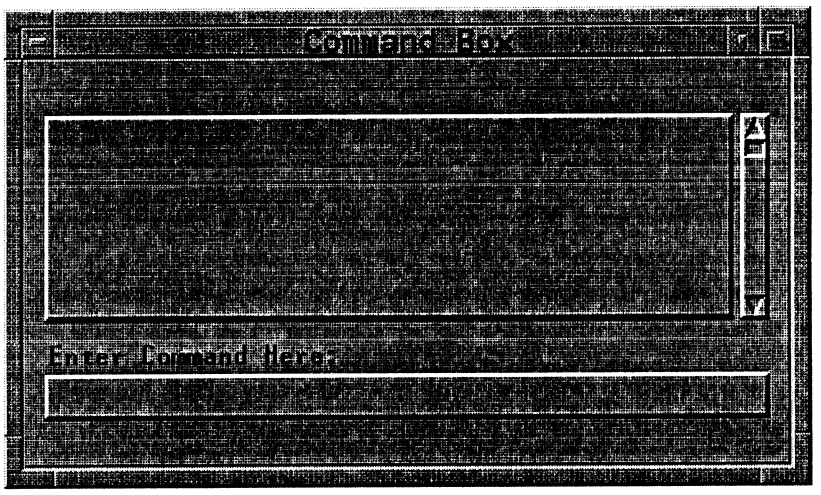


Figure 48. A Typical Command Dialog Box.

Selection

The Selection dialog box is used by people to choose from a list of several possible selections. Figure 49 illustrates a typical Selection dialog box.

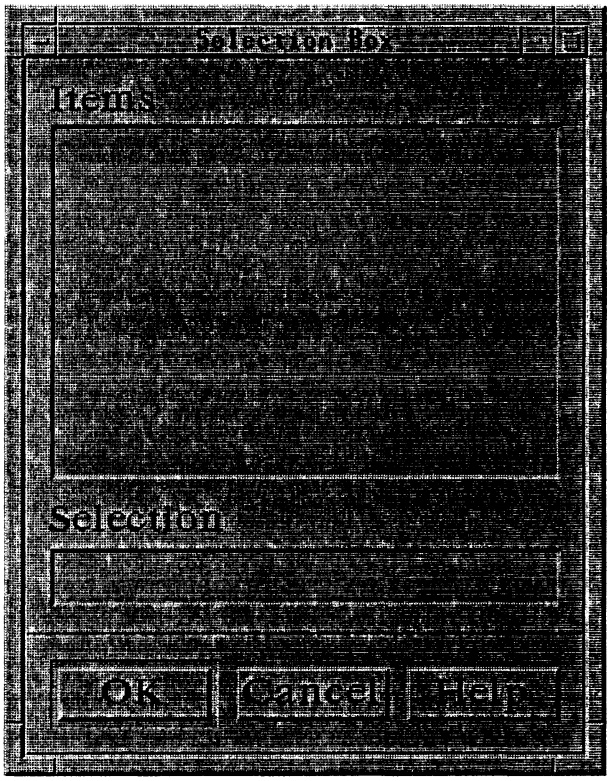


Figure 49. A Typical Selection Dialog Box.

Starting a Dialog

In general, dialog boxes are started by users choosing selections from controls such as menus. Menu selections that lead to dialog boxes follow the selection name with ellipsis (. . .) as a visual cue.

In the case of message boxes, the opposite is true. The application starts the dialog box based on some event. In the case of information and progress boxes, work progresses unimpeded by the presence of the dialog box. For the others, work on a task cannot proceed until a question has been answered, a warning of potential damage acknowledged, or some specific action taken.

Navigating Through a Dialog Box

In many cases, users don't need to carry out a long-winded conversation with a dialog box to get what they want. Rather, they will call the dialog box from the menu, make a few changes to some settings, and send the dialog box away. Availability of an appropriate default action is key to ease of use.

Your application should provide users with the ability to carry on a selective conversation with a dialog box and to easily navigate through the box. Mouse users can slide the pointer directly to a control. Keyboard users press the Tab and Arrow keys to step through a dialog box.

Determining Dialog Box Location and Size

Your application determines the size and location of its dialog boxes.

Location

Whenever possible, don't place a dialog box so that it obscures important information in the underlying window. A dialog box should be horizontally offset from the underlying window. This enables users to see the selections they have chosen.

When a dialog box relates to an item in an underlying window, position the dialog box next to the item. As much as possible, avoid covering information in the underlying window that users might refer to in conversing with the dialog box.

If it is necessary to display two dialog boxes, offset the top dialog box (the one that was called last) to the right and below the title of the underlying dialog box (the one that was called first).

Obviously, there is only a limited amount of screen real estate in which to place a dialog box. While the above suggestions seem simple enough, they can not always be followed completely. Therefore, dialog boxes, once displayed, should be movable so that users can relocate them as needed to see information in underlying windows.

Size

The initial size of a dialog box should be large enough to contain the dialog controls without crowding or visual confusion, but otherwise as small as possible.

Chapter 8
Providing Online Help

Good applications provide help facilities for the people to use when the need arises. No matter how intuitive you design your application to be, there will be times when some users get stuck. Paper documentation is important, but on-line help is often more readily available (manuals can easily get lost) and more appropriate. This is particularly true of context-sensitive help.

Users can call for help using the Help menu from the menu bar as described in Chapter 6. Since dialog boxes usually don't have menu bars, they have a Help push button which opens a help window. Help windows in turn have an action bar which contains the Help menu.

Types of Help

Motif provides for several types of Help information. Each corresponds to an entry in the Help menu. If your application does not support all forms of Help, be sure that your Help menus only include the forms you do support.

Help On Context

Context-sensitive help may be the most important type of Help information for people using your application. Context-sensitive help describes the nature of a specific control and how people use that control.

The help message is unique to the field. Context-sensitive help for a field specifying printer baud rate may be:

> Baud Rate is the speed at which your printer can accept data. Acceptable speeds are 1200, 2400, and 9600. Enter the speed in bits per second.

Users request context-sensitive help from the keyboard by moving the selection cursor to the desired field. Pressing the Help key will then display help for that field.

Using the mouse, people can use the On Context entry in the Help menu. The pointer shape changes to the help pointer shape. Selecting a field with the Help pointer displays help on that field. After the selection, the Help pointer returns to its previous shape. To cancel the context-sensitive help mode, users press *<Esc>*.

Typing the Help key while the Select button is pressed on a field will also display context help for that field.

Help On Window

This help provides information about the current window. It describes the function of the window and the tasks that can be performed using that window.

Users request help on the window via the Help menu.

Help On Keys

Applications frequently have special uses for keys, particularly function keys. Help on keys shows the meanings assigned by this application to special keys.

Users request help on keys via the Help menu.

Help Index

Users often have a certain topic in mind when they request help. This option lists alphabetically a series of topics on which help is available. Users select the topics of interest and help on those topics is then displayed in the window. Applications may provide an entry field to allow users to type in a topic of interest.

Help on Help

This option provides information on how to use the Help facility.

Help Tutorial

Complex applications may provide on-line Help tutorials. This option provides access to such a capability.

Help On Version

This option provides information about the specific version of the application. Information typically includes the name, version, and date of the application as well as copyright data.

Help Windows

Help windows are standard application secondary windows. They contain an action with at least the Help menu. Their title is the title of the window from which they were requested followed by the word "Help." They may also have a title in the client area to specify which field in the window the help window describes. Place Help windows so that they do not obscure the objects they are describing.

Chapter 9
Designing International Software

Designing software for the international market requires producing applications that are easily translatable. In a typical scenario, when an application is translated into another language, some portion of it remains constant, while other portions change.

The constant portion forms the **base** of the product. The base contains all parts that are internationally invariable. The portion that changes from country to country is the **localization** portion. Identifying and separating the base from the localization portion is important in creating applications that are efficient to translate.

This chapter discusses the following localization issues:

- Collating sequences.
- Country-specific data formats.
- Icons and pointer shapes.
- Scanning direction.
- Text translation.
- Messages.
- Copyrights.

Collating Sequences

To produce an alphanumeric list, printable characters are sorted according to a **collating sequence**. Printable characters include letters, numbers, punctuation characters, and other symbols such as asterisks (*) and ampersands (&). The collating sequence defines the value and position of a character relative to the other characters.

Many applications make frequent use of collating sequences to produce alphanumeric lists. Examples of alphanumeric lists include the following:

- A directory listing of file names.
- The output from a sorting utility.
- An index produced by a text-processing application.
- The lists produced by a database application, such as lists of names or addresses.

If your application contains sorting functions, it needs to be flexible enough to accommodate individual country-specific collating sequences. Collating sequences must not be hard-coded into your application. At run-time, your application should refer to a table holding the collating sequences.

Country-Specific Data Formats

The formats for many types of data vary from country to country. When designing data formats, use the same format for input and display. If the format changes, it should change for both. Do not make formats dependent upon other features of your application or dependent upon each other.

Country-specific data includes the following:

- Thousands separators.

- Decimal separators.
- Positive and negative values.
- Currency.
- Dates.
- Time formats.
- Time zones.
- Telephone numbers.

Thousands Separators

For separating units of thousands, the comma, period, space, and apostrophe are considered to be valid separators. Your application should allow for arbitrary separators or no separators at all. The thousands separator should be definable by people who use your application or at the time of localization.

Decimal Separators

For separating decimal fractions, the period, comma, and the center dot are considered to be valid separators. Do not use the same character for both the thousands separator and the decimal separator. The decimal separator should be definable by people who use your application or at the time of localization.

Positive and Negative Values

The plus (+) and minus (–) signs are valid either before or after a number. In applications such as a spreadsheet, allow negative numbers to be enclosed in parentheses.

Currency

For currency, the comma, period, and colon are valid separators. They should be modified independently of the separators used for other data formats.

The currency symbol is a valid separator. It can be placed before or after the numerical value, or be used as the decimal separator. Allow for one or no space between the currency symbol and the amount. The currency symbol can be changed.

Dates

If your application displays or prints the date, the date should be in the format and language of the people using your application. Properly formatting the date requires the following:

- Alphanumeric date formats should allow sufficient storage and display space to accommodate date names in other languages.
- The position of each component of an alphanumeric date should be able to be varied or omitted.
- The hyphen, comma, period, space, and slash are all valid separators for the day, month, and year.
- Numeric date formats should allow the month and day fields to be reversed. For example, the 4th of August, 1990, can be displayed as either 4/8/90 or 8/4/90.

- The format for entering the date should match the display format.

Time Formats

For time formats, hour, minute, and second components always appear in that order. Hours and seconds are never used without minutes.

The colon, period, and space are valid separators for hours, minutes, and seconds. In 24-hour notation, the four-digit format may use a separator.

Time Zones

For time zone displays, allow up to three characters. Also, allow for the time offsets to be fractions, because time offsets are not always an integer number of hours from Greenwich Mean Time (GMT).

Telephone Numbers

Telephone numbers differ in the number of digits and the format used. The space, hyphen, period, comma, and square brackets are all valid separators.

For international numbers, the plus sign is frequently used in Europe to indicate that a number is a country code. For national numbers, it is common (but not universal) to put the city code or area code in parentheses.

Icons and Pointer Shapes

It may not always be possible to design an icon, pointer shape, or other graphical symbol that adequately represents the same object or function in different countries. Culture is inherent even in seemingly universal symbols. For example, sending and receiving mail is a commonly understood function, but representing that function with an icon of a mail box may be inappropriate because the appearance of mail boxes varies widely among countries. An envelope may therefore be a more appropriate icon.

When used correctly, graphical symbols offer the following advantages:

- They are language independent and do not need to be translated.
- They can be used instead of computer terms that have no national-language equivalent.
- They may have more impact when used with text as warnings than the text alone.

Scanning Direction

Western languages scan left to right across the page (or display screen) and from top to bottom. In Eastern languages, however, this is not the case. The scanning direction of the country of localization may have an impact on the location of controls in dialog boxes, the order of selections in menus, and other areas.

Translating Screen Text

Well-written screen text makes an application easier for users to understand. It also makes translation easier.

In particular, relationships between nouns become very explicit in other languages. You should avoid stringing three or more nouns together. Use prepositions to clarify the relationship of the nouns.

Also, text translated from English is likely to expand 30% to 50%. If text space is limited, for example an error message restricted to one screen line, shorten the original version to allow for the translation.

While common words are easy to understand and translate, jargon, when translated into other languages, remains jargon and serves only to confuse and intimidate people who use your application. Avoid using jargon whenever the jargon is not a part of the working vocabulary of the people using your application.

Use the following guidelines to write screen text for translation:

- Brief and simple sentences are easy to understand and aid in translating.
- Affirmative statements are easier to understand than negative statements.
- Active voice is easier for both application users and application translators to understand.

Messages

Languages have different grammatical structures that must be accommodated when a product is adapted for a local environment.

Do not build messages dynamically; that is, do not store separate subsections of a message and assemble them for output. This will avoid embarrassing juxtapositions of words and phrases. Store each message as a complete string of variable length in separate modules or files. Do not embed messages in your code.

Appendix A
Default Keyboard and Mouse Bindings

Different keyboards may have different numbers of keys or different key labels. Similarly, different mice have different numbers of buttons. Nevertheless, the OSF/Motif environment is able to employ a standard of consistent behavior because it prescribes that the same function always be invoked in the same way.

In particular, common and frequently performed functions are associated with (bound to) particular sequences of keyboard key presses and mouse button operations. These default keyboard and mouse actions, and the functions they invoke, are presented in this appendix in tabular form.

Default Keyboard Assignments

Modifier Keys The typical keyboard also has the following **modifier** keys:

- *<Shift>*
- *<Alt>* or *<Meta>*
- *<Ctrl>*

Users use these keys to modify the meanings of other keys or mouse button operations. In other words, modifier keys are *always* used in conjunction with other keys or mouse actions. Accelerators are an example of modifier usage.

Keyboard Function Assignments The following table includes navigation, function, and special-purpose keys and their associated functions.

TABLE 6. Default Key Assignments for Functions.

Key	Function			
	No Modifier	**Shift**	**Control**	**Alt**
←	Left	Range selection	Word left	
→	Right	Range selection	Word right	
↓	Down	Range selection		Lower window to bottom of stack.*
↑	Up	Range selection		Raise window to top of stack.*
Delete	Delete	Cut	Erase to end of line	
End	End of line		End of data	
Esc	Cancel	Window menu		Next Application
F1	Help			
F2				
F3				Lower*
F4	Prompt/Pop-up menu†			Close window*
F5				Restore window*
F6	Switch window panes			Switch windows*
F9				Minimize window*
F10	Switch to menu bar			Maximize window*
Home	Beginning of line		Beginning of data	
Page Down	Page down		Page right	
Page Up	Page up		Page left	
Space bar	Select†			Window Menu‡
Tab				Next application‡
* An optional assignment and should remain available for user				
convenience.				
† Used in substitutions when special purpose key is not available.				
‡ Provides compatibility with MS-Windows.				

Default Keyboard Assignments for Window Manager Functions The following table contains the default key assignments used by the OSF/Motif Window Manager for the window menu mnemonics and accelerators:

TABLE 7. Window Manager Functions.

Menu Selection	Mnemonic	Accelerator
Restore	R	*<Alt>+<F5>*
Move	M	*<Alt>+<F7>*
Size	S	*<Alt>+<F8>*
Minimize	n	*<Alt>+<F9>*
Maximize	x	*<Alt>+<F10>*
Lower	L	*<Alt>+<F3>*
Close	C	*<Alt>+<F4>*

Note on terminology: *<Alt>+<F5>* means to hold down the *<Alt>* key and press *<F5>*.

Default Key Assignments for Text Keys The following table contains the default key assignments used by text entry boxes and other editing windows:

TABLE 8. Default Assignments for Text Entry Keys.

Key	Function			
	No Modifier	Shift	Control	Alt
<Backspace>	Delete to left			Undo last action
<Ins>	Insert/Replace	Paste	Copy	
<Del>	Delete	Cut		

Default Key Assignments for Cursor Navigation The following table contains the default key assignments used by the OSF/Motif window manager for cursor navigation:

TABLE 9. Default Assignments for Cursor Navigation Keys.

Key	Description
<Next Field>	Moves the cursor forward to the next field going from left to right and top to bottom. At the last field, the cursor wraps to the first field.
<Previous Field>	Moves the cursor back to the previous field going from right to left and bottom to top. At the first field, the cursor wraps to the last field.
<Home>	Moves the cursor to the beginning of the current line.
<Ctrl>+<Home>	Moves the cursor to the beginning of the data.
<End>	Moves the cursor to the end of the current line.
<Ctrl>+<End>	Moves the cursor to the end of the data.
<↑>	Moves the cursor to the previous choice in the client area. At the first choice, the cursor wraps to the last choice.
<↓>	Moves the cursor to the next choice in the client area. At the last choice, the cursor wraps to the first choice.
<←>	Moves the cursor to the previous choice. At the first choice, the cursor wraps to the last choice on the line above. At the top first choice, the cursor wraps to the bottom last choice.
<→>	Moves the cursor to the next choice. At the last choice, the cursor wraps to the first choice on the line below. At the bottom last choice, the cursor wraps to the top firs choice.
<PageUp>	Scrolls a window, displaying the previous page of information.
<PageDn>	Scrolls a window, displaying the next page of information.
<Ctrl>+<PageUp>	Scrolls a window, displaying information previously out of view to the left.
<Ctrl>+<PageDn>	Scrolls a window, displaying information previously out of view to the right.
<F6>	Moves the cursor from one pane to another in a split window. Precedence is left to right, top to bottom.
<F10>	Moves the cursor to the menu bar.
<Ctrl>+<←>	Moves the cursor to the previous word in a field.
<Ctrl>+<→>	Moves the cursor to the next word in a field.

Substitutions for Special Purpose Keys Not every keyboard contains all the special purpose keys mentioned in this style guide. Where a special-purpose key is available, it should be used as described. Thus keyboards with a *<Select>* key use it for selection. Some keyboards may not include all special-purpose keys. The following table presents some suggested substitutions.

TABLE 10. Special Purpose Keys and their Substitutions.

Key	Substitution
<Select>	*<Space Bar>* or *<Enter>*.
<Menu>	*<F4>*
<Help>	*<F1>*
<Alt>	*<Extend>*
<Esc>	*<F12>*
<Next Field>	*<Tab>* or *<Ctrl>+<Tab>*
<Previous Field>	*<Shift><Tab>* or *<Ctrl>+<Shift>+<Tab>*
<Enter>	*<Return>*

If neither a special-purpose key nor a recommended substitution is available, function key equivalents should be used. Applications that do not need a function key for one of the standard functions may use that function key for application-specific purposes.

Default Mouse Assignments

Default Button Bindings OSF/Motif assumes a three-button mouse. The following figures illustrate the default OSF/Motif mouse button locations and button assignments for right- and left-handed mice:

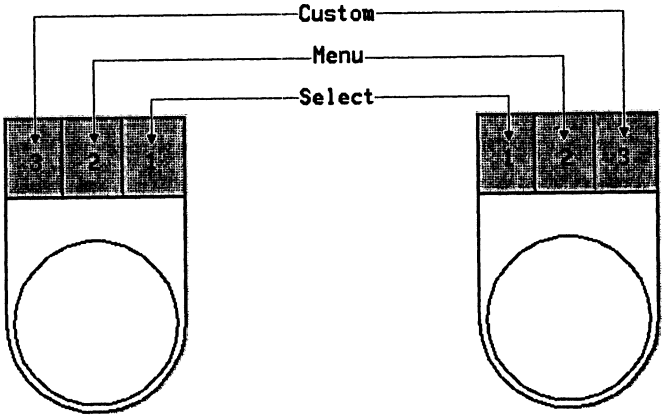


Figure 50. Default OSF/Motif Mouse Button Assignments.

Operating Scroll Bars with a Mouse

TABLE 11. Operating Scroll Bars with a Mouse.

Mouse Action	Response
Clicking the Select button with the pointer on stepping arrows.	Highlights the stepper arrow and moves the window through the underlying file by a single unit, in the direction indicated by the arrow. You must determine the appropriate unit to be scrolled in your application. For example, a unit in a spreadsheet may be a single row or column.
Pressing the Select button with the pointer on stepper arrows.	Highlights the stepper arrow and causes a continuous scroll, in unit steps, in the direction indicated by the arrow. The default step speed is approximately 30 milliseconds and is timed by the completion of the scroll operation.
Clicking the Select button with the pointer in the scroll region.	Moves the window through the underlying file by one window length minus one unit for overlap. Clicking below (or to the right of) the slider advances to the next window's worth of information. Clicking above (or to the left of) the slider moves back to the previous window's worth of information.
Pressing the Select button with the pointer in the scroll region.	Continuously moves the window through the underlying file by one window length minus one unit for overlap until the user releases the select button or until the slider moves under the pointer.
Dragging the slider with the Select button pressed.	Moves an outline of the slider. Releasing the select button repositions the window to a location consistent with the new slider location. The user can release the select button when the pointer is anywhere in the window, not just within the scroll region. This action can be canceled by clicking any of the other buttons before releasing the select button.

Object Selection

TABLE 12. The OSF/Motif Object-Action Selection Model.

Selection	Mouse Selection	Keyboard Selection
Task	**Operation**	**Operation**
Change selection focus.	Move the mouse to position the pointer.	Press the navigation keys to position the selection cursor.
Select a single object.	Click Select	Press <*Select*>
Select a range of objects.	Drag Select	Press <*Select*>+navigation keys.
Select all objects between current location and previous location.	Click <*Shift*>+Select	Press <*Shift*>+<*Select*>
Select an additional object.	Click <*Ctrl*>+selection operation	Press <*Ctrl*>+selection operation.
Deselect an object.	<*Ctrl*>+selection operation on a selected object.	<*Ctrl*>+selection operation on a selected object.

Incremental Text Selection

In addition to text selection using the press-and-drag method, your application can use an incremental selection method. In this method, the user can perform multiple clicks of the select button to select successively larger (or smaller) portions of text.

TABLE 13. Incremental Text Selection.

Number of Select Button Clicks	Text Block Selected
1 click	Null selection.
2 clicks	Word.
3 clicks	Line.
4 clicks	Paragraph.
5 clicks	Module.

Glossary

accelerator	A key or sequence of keys (typically a modifier key and some other key) that provides a shortcut, immediately accessing a program function.
active window	The terminal window where what you type appears. If there is no active window, what you type is lost. Only one terminal window can be active at a time.
button	A button on a mouse pointing device; mouse buttons can be mapped to the keyboard. A graphical control on a window frame or in a dialog box that works by pressing it.
button binding	Association of a mouse button operation with a window manager or application function.
cancel	A label given to a push button in some dialog boxes that performs the action of closing the dialog box without implementing any changes.
cascading menu	A submenu that provides selections that amply the parent selection on a pull down or pop up menu.
check button	A control used to select settings that are not mutually exclusive. The visual cue to the selection is frequently that the button is filled in or checked.
click	To press *and release* a mouse button. The term comes from the fact that pressing and releasing the buttons of most mice makes a clicking sound.
client	An application program written specifically for the X Window System. Some clients make their own windows. Other clients are utility programs.
client area	The area within the borders of a primary window's frame that is controlled by an application.
clipboard	An device used to store text or graphics during cut-and-paste operations.
close	A label given to a push button in some dialog boxes that performs the action of closing the dialog box. Close is also used as a selection in menus to close the window associated with the menu.
<*Ctrl*>+click	Clicking the mouse while pressing and holding down the <*Ctrl*> key.
<*Ctrl*> key	The keyboard key usually labeled <*Ctrl*> and used as a modifier key.
control panel	An area of a window similar to the control panels in real life that is used to hold push buttons and other graphical controls.
<*Ctrl*>+selection operation	Pressing and holding down <*Ctrl*> and then pressing the Select mouse button or the <*Select*> key to toggle the selection state of an additional selection.
cursor	A graphical image, usually a pipe (|) or block, that shows the location where text will appear on the screen when keys on the keyboard are pressed or where a selection will be made when the Select mouse button is clicked or the <*Select*> key is pressed.

default (selection)	An object or action that is specified for selection if no other selection is specified.
desktop	Another term for workspace.
dialog box	A secondary window that displays under the direction of the user and contains application controls.
dimmed selection	A selection that is not currently available.
double-click	To press *and release* a mouse button twice in rapid succession.
drag	To press *and hold* a mouse button while moving the mouse.
entry box	An area used for text entry. Typically, an entry box is part of a dialog box.
graphical user interface	A form of communication between users and computers that uses graphics-oriented software such as windows, menus, and icons, to ease the burden of the interaction.
grayed selection	A menu selection that is not currently available and so has been dimmed.
group box	A graphical rectangle drawn or etched around a set of controls that provides a visual cue that the controls contained within the box are logically related.
Help key	A special-purpose key on some keyboards that is used to display the Help menu.
Help on Context	Provides specific, context-sensitive help.
Help on Window	Provides general help on an application.
Help on Keys	Provides help on which keyboard keys have been assigned to active functions.
Help Index	Provides entry into the Help index function.
Help Tutorial	Provides entry into the Help tutorial.
Help on Version	Provides version and copyright information for an application.
highlight	A graphic technique used to provide a visual cue to the current selection or to the current location of the input focus. Highlighting is frequently accomplished by reversing the video of the selection.
hotspot	The area of a graphical image used as a pointer or cursor that is defined as the point of the pointer or cursor.
hourglass	A graphical image used to symbolize the passage of time and provide a visual cue that the application is currently performing an operation.
I-beam	A graphical image used to represent the location of the mouse pointer in a text entry box and which provides a visual cue that text can be entered in an area.
Icon	A small graphical image used to represent a window. Windows can be turned into icons or minimized to save room or unclutter the workspace.
inactive	A window that does not have the input focus.
insertion cursor	The graphical symbol that provides the visual cue to the location of the insertion point.

insertion point	The point in a text entry area or graphics entry area, shown by the presence of the insertion cursor, where text or graphics will appear when keys are pressed on the keyboard.
keyboard	One of many input devices; the traditional method of entering text into an application.
label	The text part of an icon or graphical control.
list box	A control that provides users with a scrollable list of options from which to choose.
location cursor	A graphical symbol that marks the current location of the keyboard input focus for selection. Typically, this symbol is a box that surrounds the current object. The location cursor is sometimes known as the selection cursor.
lower	To move a window to the bottom of the window stack on the workspace.
maximize	To enlarge a window to its maximum size.
maximize button	A control button placed on the MWM window frame and used to initiate the maximize function.
menu	A list of available selections from which a user chooses.
menu bar	A rectangular area at the top of the client area of a window that contains the titles of the standard pull down menus for that application.
message box	The generic name for any dialog box that provides information, gives the current state of a work in progress, asks a question, issues a warning, or draws attention to an error.
minimize	To turn a window into an icon. The term "iconify" is sometimesnused instead of minimize.
minimize button	A control button placed on the MWM window frame and used to initiate the minimize function.
mnemonic	A single character (frequently the initial character) of a menu selection which, when the menu is displayed and the character is pressed on the keyboard, initiates the selection.
modifier key	A key that, when pressed with another key, changes the meaning of the other key. *<Ctrl>*, *<Alt>*, and *<Shift>* are modifier keys.
mouse	A pointing device commonly used in conjunction with a keyboard in point-and-click, object-oriented user interfaces.
mouse button	One of the buttons on a mouse pointing device. Mouse buttons can be pressed, released, dragged, clicked, and double-clicked.
open	To start an action or begin working with a text, data, or graphics file.
paste	Inserting data into an area. Pasting is commonly used in reference to text files where a block of text is cut from one area and pasted into another area.
point	To position the pointer or location cursor.
pointer	The graphical image that appears on the workspace and represents the current location of a mouse or other pointing device.

pointing device	A device such as a mouse, trackball, or graphics tablet that allows users to move a pointer about on the workspace and point to graphical objects.
pop-up menu	A menu that provides no visual cue to its presence, but simply pops up when users perform a particular action. Pop up menus area associated with a particular area of the workspace, such as the client area of an application, and users must memorize where these areas are.
press	To hold down a mouse button or a key. Note that to hold down a mouse button *and move* the mouse is called dragging.
primary window	A top-level window of an application. Primary windows can be minimized.
pull-down menu	A menu that is pulled down from a client application's title bar.
push button	A graphic control that simulates a real-life push button. People use the pointer and mouse to push the button and start some action.
radio button	A graphic control that simulates the buttons on a real-life car radio. Each button represents a mutually exclusive selection. Radio buttons are typically used for setting states or modes.
release	To let up on a mouse button or key that has been pressed. Sometimes it is the press that initiates the action; sometimes it is the release.
resize	To change the height or width of a window.
resize border	The MWM frame part that surrounds the client area of an application and that is used to change the height or width of the window.
restore	To return an icon or maximized window to its normal size.
root menu	See *workspace menu.*
root window	See workspace.
save	To write changes to a data file to a storage device for safekeeping.
screen	The physical CRT (Cathode Ray Tube) that displays information from the computer. In the OSF/Motif environment, in most cases screen and workspace are synonymous.
scroll bar	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.
selection cursor	See location cursor.
select	To choose an object to be acted upon or an action to be performed.
selection	The object or action that is selected. Menus are composed of selection items. Dialog boxes contain controls each of which represents a selection.
Select button	The mouse button used to make a selection.
<*Select*> key	The special-purpose keyboard key used to make a selection. Keyboards without a Select key use a substitute to provide the select functionality.
setting	A parameter that does not cause an operation, but rather influences the outcome of related operations. Once set, a setting influences

	subsequent operations.
<*Shift*> key	One of the modifier keys on the keyboard.
<*Shift*>+click	A modified mouse button operation performed by pressing the <*Shift*> key and clicking a mouse button.
<*Shift*>+select	A modified mouse button or keyboard operation performed by pressing the <*Shift*> key and then doing a selection operation with either the mouse (clicking the Select mouse button) or keyboard (typing the <*Select*> key.
<*Shift*>+drag	A modified mouse button or keyboard operation performed by pressing the <*Shift*> key and then doing a drag operation with either the mouse (pressing a mouse button and moving the pointer) or the keyboard (typing the cursor navigation keys).
size	Used as a verb to describe changing the size of a window on the workspace.
slider	One of graphical components of a scroll bar or scale. The slider is the object that is dragged along the scroll area to cause a change.
state	A generic term used to describe the condition or mode of an object or action.
submenu	A cascading menu.
system menu	See window menu.
switch	To change from one window to another or from one application to another.
text cursor	See insertion cursor.
title area	The area at the top of the window frame immediately beneath the resize border. The title bar has two functions: it contains a title or name that identifies the window and it can be grabbed and dragged to relocate the window.
title bar	The bar across the top of an MWM managed window that consists of the window menu button, the title area, and the window control buttons.
transient window	A window of short duration such as a dialog box. The window is only displayed for a short time, usually just long enough to convey some information or get some operational directions.
type	As a verb, to press and release a keyboard key.
underlined letter	A letter used as a mnemonic. The underline provides the visual cue to the mnemonic function.
window	A data structure that represents all or part of the CRT display screen. Visually, a window is represented as a rectangular subset of the display screen.
window decoration	The frame and window control buttons that surround windows managed by the a window manager such as the OSF/Motif Window Manager.
window frame	The area surrounding a window. A window frame can consist of a resize border, a window menu button, a title bar, and window control buttons.

window manager	A program that controls the size, placement, and operation of windows on the workspace. The window manager includes the functional window frames that surround each window object and may include a separate menu for the workspace.
window menu	The menu that displays when the window menu button is pressed. The window menu typically contains selections for restoring, moving, sizing, minimizing, maximizing, and closing the window.
window menu button	The graphical control button that appears at the left side of the title bar in the window frame.
workspace	The CRT screen. The area on which the windows of a user's environment display. The workspace is sometimes called the desk, desktop, or root window.
workspace menu	An optional pop up menu associated with the workspace.

Index

<Alt> 2-2, 2-3, 6-7
  + <Esc>, A-1
  or <Meta>, A-1
<Backspace> A-1
<Ctrl> A-1
  + <End>, A-4
<Ctrl>+selection operation, 5-3
<Del> A-1
<End> A-4
<Enter> 7-5
<Esc> 2-3, 6-8, A-4
<Extend char>, 2-2
<Extend> 2-2
<F10> 6-7, A-4
<F6> A-4
<Help> 2-3, A-4
<Home> A-4
<Ins> A-1
<Menu> 2-3, 6-7, A-4
<Meta> 2-2
<PageDn> A-4
<PageUp> A-4
<Select> 2-3, 5-1, A-1, A-4
<Shift> A-1
  + <Tab>, A-4
<Space Bar>, A-1
<Tab> A-4
<→> A-4
<←> A-4
<↓> A-4
<↑> A-4

a

accelerators 6-16, 6-7, A-1
action 6-8
  dialog box, 7-5
actions 6-5, 7-2
active point, 2-6
  voice, 9-3
  window, 2-1
acts requiring actions, 1-3
additional (non-contiguous) selection operation, 2-12
  menus, 6-8
  selection, 2-12
adjust the client area, 4-8
affirmative statements, 9-3
align columns of controls, 4-9
alignment 4-9
alphanumeric date formats, 9-2
  lists, 9-1
alternate method, 6-7
alternatives 6-1, 7-5
analog-style controls, 5-5
anatomy 3-3
  of dialog box, 7-3
ANSI keyboards, 2-2
application developers, 1-1
  extras, 5-8
  menus, 6-7
  program, 1-4, 2-1
  title, 3-6
application-modal 7-2
application-oriented programs, 3-3
application-supplied controls, 7-3
Apply 7-2
appropriate control, 4-8
area 4-1, 4-3
arrangement 4-8
arranging controls in a dialog box, 7-4
  push buttons, 7-2
arrow 2-2
arrows 5-6
assisting memory, 6-6
associated functions, A-1
auto tabbing, 5-5
auto-selection 2-14
automatic entry, 5-9
  scrolling, 5-6
avoiding menu selection, 6-8

b

bar 6-3
base 9-1
behavior 2-1
bitmaps in menus, 6-17
box 3-8, 5-3, 5-4, 7-1
  controls, 7-3
boxes 5-1, 5-3, 5-5, 7-3
browsing menu bar, 6-7
button 3-4, 5-1
  assignments, A-5
  bindings, A-5
  defaults, 7-4
buttons 2-5, 3-6, 5-1, 7-3

c

Cancel 7-2
canceling a selection, 2-13
cascading 4-4
  menu, 6-1
characteristics of dialog boxes, 7-1
check buttons, 4-7, 5-2, 6-5, 7-5
child menus, 6-4
choosing menu selections, 6-8
clear 6-11
click-to-type 2-1
clicking 2-5
  menus, 6-7
client area, 3-7, 4-1, 4-8
  controls, 4-3
Close 7-2
close 3-5, A-1
collating sequences, 9-1
columns 4-9
combining controls, 5-9
  mouse and keyboard input, 2-2
command line, 4-3
common dialog boxes, 7-7
  settings, 4-9
commonly performed functions, A-1
components 5-6, 6-4
consistency 1-2
  of behavior, 2-1
context-sensitive help, 1-4
contiguous objects, 2-9
control buttons, 3-5
  panel, 4-2, 4-5
  panels, 4-8
controlling program operation, 7-1
controls 4-3, 4-4, 4-6, 4-7, 4-8, 5-1, 5-5, 5-9, 7-1, 7-2, 7-3
copy 6-11
country-specific data formats, 9-1
cue 2-10, 4-7, 4-8
currency 9-1, 9-2
current progress, 7-5
cursor 2-1, 2-3, 2-4, 5-5
  shapes, 5-5
cut 6-11

d

data 7-1
  formats, 9-1, 9-2
date formats, 9-2
  formatting, 9-2
decimal separators, 9-1
default action, 2-13
  assignments, A-1, A-4
  button bindings, A-1
  keyboard, A-1
  mouse assignments, A-5
  push button, 4-9
  selection, 7-4
delete 5-4
deselecting 2-13
design 4-1, 4-8
  a graphical symbol, 9-3
  alternatives, 6-1
  an icon, 9-3
  conformity, 6-7
  menus, 6-15
designing 4-6
  mouse pointers, 2-6
desk 3-1
desktop 3-1
destruction 1-3
device 2-2
dialog box, 4-7, 6-16, 7-1, 7-10, 7-11, 7-3, 7-4
  box actions, 7-2
  boxes, 4-5, 4-7, 7-1, 7-5, 7-7
differences 7-5
digital readout, 5-5
direct manipulation, 1-3, 2-4
direction of scan, 9-3
directional arrows, 4-3, 5-6
disabling a menu selection, 6-16
  controls, 7-2
displaying information, 7-1
do-it-yourself' menus, 6-18
double-clicking 2-13, 2-5, 7-5
dragging 2-5
  menus, 6-7
dual accessibility, 2-2

e

edit menu, 6-11
editing windows, A-1
elements 3-3
ellipsis 7-10
empowering the user, 1-2, 7-2
ending a dialog, 7-1
entry 5-9
  box, 5-4
  boxes, 5-5
errors 1-3
exclusive 4-8
exit 6-9
explicit destruction, 1-3

focus, 2-1

f

file menu, 6-9
flexibility 1-3
focus 2-1
format 2-2
formats 9-1, 9-3, 9-4
frequency of menu selections, 6-16
frequently performed functions, A-1
function 2-2, A-1
  key equivalents, A-1
functions 3-3, 3-5, A-1

g

gain 2-6
general functions, 2-2
graphical controls, 4-4, 5-1, 7-1
  user interface, 2-1
group 6-16
  box, 5-3, 5-8
  boxes, 4-5
  like menu selections, 6-16
grouping like controls, 4-5
groups 4-6, 6-16
Help 7-2, 8-1

h

help 1-4
  menu, 6-14
horizontal panes, 4-3
hot spot, 2-6

i

icon box, 3-8
icons 9-3
  in menus, 6-17
immediacy of response, 1-3
implementation-dependent 3-5
implicit focus, 2-1
incremental searching, 5-9
  text selection, A-7
index 6-14
information dialog box, 7-5
input 2-2
  devices, 2-2
  focus, 2-1, 3-6
  focus model, 2-1
  selection model, 1-4
insertion cursor, 2-3, 2-4, 5-5
instructions 7-1
interface 3-1
international 9-1
  numbers, 9-3
internationalization 9-1

k

key labels, 2-2
keyboard 2-2
  accelerator, 6-5
  accelerators, 6-7
  function assignments, A-1
  input, 2-2
  key presses, A-1
  navigation, 2-7
  selection, A-7
  substitutions, 2-3
keys 2-2, 6-8

l

label 5-4, 5-8
labels 5-1
language independent, 9-3
layout 4-8
left-handed operation, 2-5
like menu selections, 6-16
line A-7
list box, 5-3
  box with scrollbar, 7-5
  boxes, 4-8
listing selections, 6-16
localization 9-1
location 2-6, 7-11
  cursor, 2-1, 2-3
logical groups, 6-16
lower 3-5, A-1

m

main window, 3-1, 3-2, 4-5
manipulation 1-3
manipulations 1-4
maximize 3-5, A-1
  button, 3-6
memory 6-6
menu 3-4, 6-1, 6-11, 6-12, 6-13, 6-14, 6-9
  bar, 4-1, 6-3, 6-7

toggling a selection, 2-12
track listener, 2-1
  pointer, 2-1
translating text, 9-3
traversal 2-2
tutorial 6-14
types of boxes, 5-3

u

undo 1-4, 6-11
undoing a selection, 2-13
user 7-2
  interface, 2-1, 3-1
using a scale, 5-5

v

valuator 5-5
valuators 5-1, 5-5
values 9-2
vertical alignment, 4-9
  column, 4-7
  panes, 4-3
view menu, 6-12
visual cue, 2-1, 2-10, 4-7, 4-8, 6-8
  cues, 6-5
visually distinct, 6-4

w

warning dialog box, 7-6
warnings 1-3
widget developers, 1-1
window 2-1, 3-1, 3-2, 3-3, 3-7, 4-5
  control buttons, 3-5
  frame components, 3-6
  manager, 1-4, 3-3, 3-5, 3-6
  manager developers, 1-1
  menu bar, 4-1
  panes, 4-3
  types, 3-1
Windows 3-1
windows 3-3, 3-6, 4-1
word A-7
workbench 3-1
workspace 3-1

y

Yes 5-1

z

zoom feature, 2-6

press-and-drag method, A-7
pressing 2-5
primary window, 3-1, 3-2
print 6-9
printing of controls, 4-8
priority 4-8
program 2-1
programs 3-3
progress dialog box, 7-5
pull down menu, 4-4
pull-down menu, 4-1, 6-1, 6-16
purpose of dialog boxes, 7-1
push button, 5-1, 7-4
  buttons, 3-5, 4-6, 4-7, 4-9, 7-2, 7-5

q

question dialog box, 7-6

r

radio button, 5-1
  buttons, 4-7, 4-8, 6-5, 7-5
range selection, 2-10, 2-11, 2-9
real estate driven, 2-1
recommended substitution, A-1
related functionality, 6-5
  items, 6-16
releasing 2-5
required actions, 7-5
Reset 7-2
resize border, 3-7
response immediacy, 1-3
restore 3-5, 3-6, A-1
right-handed operation, 2-5
routings 6-5

s

sash 4-8
save 6-9
  as, 6-9
  space, 6-3
scale 5-5
  valuator, 5-5
scanning area, 4-8
  direction, 9-3
scroll bar, 4-3, 5-6, 5-8
  bars, 5-3
  incrementally, 5-6
  region, 5-6
searching 5-3, 5-9
secondary windows, 3-3, 4-1
select mouse button, 2-5
selecting the default action, 2-13
selection 2-12, 2-13, 2-9, 5-1, 5-2, 7-4
  action, 6-8
  cursor, 2-3
  model, 1-4, 2-7
selections 6-5
separator 9-1
sequence of controls, 4-8
sequences 9-1
sets 4-6
settings 6-5
shapes 2-6
shortcuts to selection, 6-16
single selection, 2-8
  selection with a keyboard, 2-9
single-character mnemonics, 6-4
size 3-5, 7-11, A-1
slider 4-3, 5-5, 5-6
  size, 5-6
soliciting data, 7-1
special purpose keys , A-4
special-purpose 2-2
  keys, A-1
speed searching, 5-3
speeding data entry, 5-5
standard actions, 7-2
  pushbuttons, 7-2
starting a dialog box, 7-10
stepper button, 5-8
  buttons, 7-3
structures 6-16
sub-areas 4-1
submenus 6-16, 6-4
substitutions A-4
symbol 9-1

t

telephone numbers, 9-3
text block, A-7
  box, 5-8
  cursor shapes, 5-5
  entry boxes, A-1
  selection, A-7
  translation, 9-3
text-formatting keys, 2-2
thousands separators, 9-1
time formats, 9-3
  zones, 9-3
title 5-4, 5-8, 6-5
  area, 3-6
titles 6-4

components, 6-4
mouse button, 2-5
selection, 6-16, 6-8
selection types, 6-5
structures, 6-16
title, 6-5
menus 4-4, 4-7, 6-1, 6-15, 6-17, 6-3, 6-4, 6-6, 6-7, 6-8
message area, 4-3
boxes, 7-1
dialog boxes, 7-5
messages 9-4
minimize 3-5, A-1
button, 3-6
mnemonic 4-1
mnemonics 6-16, 6-4, 6-5, 6-6, A-1
modal 4-7
dialog box, 7-2
model 2-7
modeless 4-7
modifier 2-2
keys, A-1
usage, A-1
modify 2-2
module A-7
motif window manager, 2-1
mouse button, A-1, A-5
button bindings, A-1
buttons, 2-5
input, 2-2
movement, 6-3
navigation, 2-7
pointer, 2-6
pointer shapes, 2-6
selection, A-7
speed, 2-6
travel, 4-7
move 3-5, A-1
mouse button, 2-5
multiple cascading submenus, 6-16
selection, 2-7, 5-2
windows, 3-6
mutually 4-8
exclusive alternatives, 7-5
MWM 1-4, 3-3

n

national numbers, 9-3
navigating a dialog box, 7-10
navigation 2-2, 2-7, A-4
new 6-9
non-contiguous selection, 2-12
null selection, A-7
numeric 2-2
date formats, 9-2

o

object 2-7
action, 1-4
deselection, 2-13
selection, A-7
object-action selection model, 2-7
object-oriented programs, 3-3
obvious selections, 4-9
off by one, 6-16
OK 5-1, 7-2
open 6-9
operating scroll bars, 5-6
operation 2-2, 2-5
operational models, 2-1
operations 2-5
option menus, 4-8
options 4-8
menu, 6-13
ordering menu selections, 6-16
organizing 4-1
OSF/Motif design principles, 1-2
window manager, 3-3
overstrike cursor, 5-5

p

panels 4-5, 4-8
panes 4-3
paragraph A-7
parent menus, 6-4
windows, 3-3
parts 3-3
paste 6-11
pending delete, 5-4
pipe (/), 2-4
point-and-click 1-4, 6-1
point-and-drag 6-1
pointer 2-1, 2-6
shapes, 2-6, 9-3
speed, 2-6
pointing devices, 2-4
pop ups, 4-4
pop-up 6-16
area, 6-7
menu, 4-7, 4-9, 6-1, 6-3
position 2-6, 4-8
positive and negative values, 9-1
pre-formatting entry areas, 5-5
presenting multiple controls, 4-7

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name _____ Your Title _____

Company _____ Phone _____

Street _____

City _____ State _____ Zip _____

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title _____ Manual No. _____

Who are you?

- ☐ EDP/MIS Manager
- ☐ Analyst/Programmer
- ☐ Other _____
- ☐ Senior Systems Analyst
- ☐ Operator
- _____
- ☐ Engineer
- ☐ End User

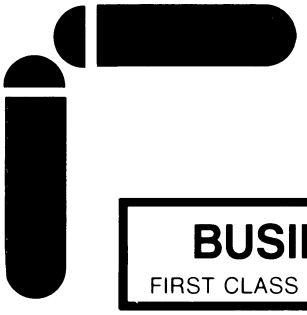
How do you use this manual? *(List in order: 1 = Primary Use)*

- ___ Introduction to the product
- ___ Tutorial Text
- ___ Other
- ___ Reference
- ___ Operating Guide
- _____

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:



NO POST
NECESS
IF MAIL
IN TH
UNITED ST

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

DataGeneral

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

OSF/Motif™
Style Guide

069-100323-00

Cut here and insert in binder spine pocket

DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

069-100323-00