DECwindows Companion to the OSF/Motif Style Guide

Order Number: AA–Q8YUA–TE

August 1994

This document is a supplement to the *OSF/Motif Style Guide*. It does not repeat the *OSF/Motif Style Guide*, but rather provides additional information concerning the style of OSF/Motif user interfaces. This document also provides guidelines concerning widgets included in DECwindows Motif.

Revision/Update Information: This is a revised manual.

Operating System: DEC OSF/1 Version 3.0 or higher
Microsoft Windows NT Version 3.1 or higher
ULTRIX Version 4.4 or higher

Software Version: DEC OSF/1 Version 3.0 or higher
eXcursion (SDK) for Windows NT Version 1.2 or higher
ULTRIX Worksystem Software Version 4.4 or higher

Digital Equipment Corporation
Maynard, Massachusetts

August 1994

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from Digital or an authorized sublicensor.

© Digital Equipment Corporation 1994. All Rights Reserved.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: Alpha AXP, AXP, Bookreader, CDA, DDIS, DEC, DECdecision, DEC FUSE, DECnet, DECstation, DECsystem, DECterm, DECUS, DECwindows, DECwrite, Digital, DTIF, eXcursion, LN03, MASSBUS, MicroVAX, OpenVMS, PrintServer, Q–bus, ReGIS, ULTRIX, ULTRIX Mail Connection, ULTRIX Worksystem Software, UNIBUS, VAX, VAXcluster, VAX DOCUMENT, VAXserver, VAXstation, VMS, VT, XUI, and the DIGITAL logo.

Adobe, Display PostScript, and PostScript are registered trademarks of Adobe Systems Incorporated.

Helvetica and Times are trademarks of Linotype AG or its subsidiaries, or both.

ITC Avant Garde Gothic is a registered trademark of International Typeface Corporation.

Open Software Foundation, OSF, OSF/1, OSF/Motif, and Motif are trademarks of the Open Software Foundation, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

NT and Windows NT are registered trademarks and Windows is a trademark of Microsoft Corporation.

ZK5637

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

About This Manual xi

1 Designing Dialog Boxes

1.1 Determining the Type and Placement of a Dialog Box 1–2
1.1.1 Types of Dialog Boxes 1–2
1.1.2 Placing Dialog Boxes 1–7
1.1.2.1 Placing a Single Dialog Box 1–7
1.1.2.2 Nesting Multiple Dialog Boxes 1–9
1.2 Using Window Decorations 1–14
1.2.1 Titles in Dialog Boxes 1–15
1.2.2 Resizing Dialog Boxes 1–16
1.3 Creating Push Buttons 1–21
1.3.1 Default Push Button 1–21
1.3.2 Cancel Button 1–22
1.3.3 Labels in Push Buttons 1–22
1.3.4 Push-Button Arrangement 1–22
1.3.4.1 Horizontal Push-Button Arrangement 1–23
1.3.4.2 Vertical Push-Button Arrangement 1–28
1.3.5 Push-Button Dimensions 1–28
1.3.6 Help Push Button 1–30
1.4 Using Standard Message Dialog Boxes 1–31
1.4.1 Question Dialog Box 1–32
1.4.2 Error Dialog Box 1–34
1.4.3 Information Dialog Box 1–35
1.4.4 Warning Dialog Box 1–36
1.4.5 Working Dialog Boxes 1–37
1.5 Creating a Logical and Appealing Layout of Controls 1–38
1.5.1 Group Related Choices Graphically 1–38
1.5.2 When to Avoid Using Punctuation in Dialog Boxes 1–39
1.5.3 Keep Dialog Boxes Simple 1–40
1.5.4 Use Uniform Spacing Within the Dialog Box 1–42
1.5.5 Align Labels and Text Fields 1–45

1.5.6 Promote Scanning from Left to Right, Top to Bottom 1–48
1.5.6.1 Use the Top Left Corner 1–49
1.5.6.2 Present Related Controls in a Clearly Visible Series 1–49
1.5.6.3 Clarify Structure 1–50
1.5.6.4 Provide Appropriate Defaults 1–51
1.5.6.5 Use Option Buttons to Conserve Space 1–52
1.5.6.6 Initializing Values in Dialog Boxes 1–53
1.5.6.7 Provide a Variety of Selection Controls 1–54
1.5.7 Creating Tab Groups 1–54
1.5.7.1 Navigating Between Tab Groups 1–55
1.5.7.2 Navigating Within Tab Groups 1–59
1.6 Presenting Text in Dialog Boxes and Other Screen Objects 1–61
1.7 Creating Title Bars 1–62
1.8 Using the Working Dialog Box and the Wait Cursor 1–63

2 Designing Menus and Menu Items

2.1 Naming Menus and Menu Items 2–1
2.2 Organizing Menus and Menu Items 2–3
2.3 Using Ellipses in Menu Items 2–5
2.4 Disabling Menu Items 2–6
2.5 Designing Pop-up Menus 2–8
2.5.1 Characteristics to Give to Pop-up Menus 2–9
2.5.2 Determining the Actions of Pop-up Menus 2–11
2.5.3 Choosing Between a Pop-up Menu or a Control Panel 2–11
2.5.4 Designing Pop-up Menus with Submenus 2–11
2.6 Designing Tear-off Menus 2–12
2.7 Items in the File Menu 2–13
2.8 Items in the Selected Menu 2–21
2.9 Items in the Edit Menu 2–22
2.10 Items in the View Menu 2–26
2.11 Items in the Options Menu 2–27
2.12 Items in the Help Menu 2–27
2.13 Items in the Windows Menu 2–29

3 Customizing the Interface

3.1 General Guidelines for Providing Customization Features 3–1
3.2 Designing Options Menu Items and Dialog Boxes 3–4
3.2.1 Adding Command Menu Items 3–5
3.2.2 Adding Dialog Boxes 3–5
3.3 Guidelines for Customizing Windows, Color, and Text 3–7
3.3.1 Window Positions and Sizes 3–8

3.3.2 Colors 3–8
3.3.3 Size and Rendition of Text 3–8

4 Using Color

4.1 Why Use Color? 4–1
4.2 General Guidelines and Specific Recommendations 4–2
4.2.1 Before You Select Colors 4–2
4.2.2 After You Select the Colors 4–3
4.3 Using Digital's ColorMix Widget 4–5

5 Designing Help

5.1 Methods of Accessing Help 5–1
5.1.1 Help Pull-Down Menu 5–1
5.1.2 Help Command 5–2
5.1.3 Help Push Button 5–2
5.1.4 Help Key 5–2
5.2 Guidelines for Designing and Writing Online Help 5–2
5.2.1 Planning Help Topics 5–2
5.2.1.1 Organizing Help Topics 5–5
5.2.1.2 Writing Help Topics 5–11
5.2.2 Remember Translation 5–14
5.3 Digital Tools for Creating Online Help 5–14
5.3.1 Using Digital's Help Widget 5–14
5.3.1.1 Title Bar 5–15
5.3.1.2 Menu Bar 5–16
5.3.1.3 Help Topic 5–17
5.3.1.4 Additional Topics 5–17
5.3.1.5 Push Buttons 5–17
5.3.2 Using the DECwindows Motif Help System 5–18

6 Using Digital-Specific Widgets

6.1 Using Digital's Print Widget 6–1
6.2 Using the Structured Visual Navigation Widget 6–5
6.2.1 Determining the Components of an Entry 6–10
6.2.2 Designing the Appearance of Your Hierarchy 6–11
6.2.2.1 Showing Icon States 6–11
6.2.2.2 Aligning Entries 6–11
6.2.2.3 Using Fonts Within a Hierarchy 6–12
6.2.2.4 Choosing Selection Modes 6–12
6.2.2.5 Choosing Selection Line Length 6–14

6.2.3 Providing Items in the View Menu 6–15

A Keyboard and Mouse Bindings for DECwindows Motif

A.1 Keyboard Bindings A–1
A.2 Mouse Bindings A–11
A.3 Standard Accelerators A–13

B Common Motif Terms Translated into European Languages

B.1 Danish and Dutch Terms B–2
B.2 Finnish and French Terms B–7
B.3 German and Italian Terms B–12
B.4 Norwegian and Portuguese Terms B–17
B.5 Spanish and Swedish Terms B–22

C Sources of Further Information

C.1 Books C–1
C.2 Periodicals C–3

Index

Figures

1–1 Modeless Dialog Boxes Allow Users to Work in Other Windows 1–3
1–2 Caution Pointer with a Modal Dialog Box 1–5
1–3 Place Dialog Boxes Appropriately 1–8
1–4 Using an Options... Button to Indicate a Secondary Dialog Box 1–10
1–5 A Dialog Box with Too Many Steps 1–11
1–6 Nesting Used to Redesign a Dialog Box with Too Many Steps 1–12
1–7 Nesting Dialog Boxes 1–13
1–8 Window Decorations on Primary and Secondary Windows . . . 1–15
1–9 Appropriate and Inappropriate Titles in Dialog Boxes 1–16
1–10 Use Resize Borders with Text-Entry Fields or List Boxes . . . 1–17
1–11 Move Push Buttons Closer Together 1–18
1–12 Do Not Truncate Push Buttons 1–19

1–13	Do Not Truncate Labels in Push Buttons	1–19
1–14	Do Not Overlap Push Buttons	1–20
1–15	Increase Size and Layout of Dialog Box Proportionally	1–21
1–16	Equally Spaced Push Buttons at the Bottom of a Dialog Box	1–23
1–17	Vertically Arranged Push Buttons at the Side of a Dialog Box	1–28
1–18	How to Handle a Push Button That Is Larger Than Others	1–30
1–19	Examples of Question Dialog Boxes	1–33
1–20	Error Dialog Box	1–35
1–21	Information Dialog Box	1–36
1–22	Warning Dialog Box	1–37
1–23	Working Dialog Box	1–38
1–24	Group Related Choices Using Space or Bevels	1–39
1–25	When to Avoid Punctuation in Dialog Boxes	1–40
1–26	Simplifying Dialog Boxes	1–41
1–27	Use Uniform Spacing	1–43
1–28	Appropriate and Inappropriate Spacing Between Labels and Fields	1–44
1–29	Vertical Left Justified and Horizontal Arrangements	1–45
1–30	Align Labels with the Baseline of Text	1–45
1–31	Left Justified Alignment of Labels and Text Fields	1–46
1–32	Right Justified Alignment of Labels and Text Fields	1–46
1–33	Appropriate and Inappropriate Labeling of Radio Buttons	1–50
1–34	Provide Appropriate Defaults	1–52
1–35	Use Option Buttons to Conserve Space	1–53
1–36	Using Two Option Buttons	1–53
1–37	Provide a Variety of Selection Controls	1–54
1–38	Use the Tab Key to Move Between Tab Groups	1–57
1–39	Use the Tab Key to Move Between Tab Groups–continued	1–58
1–40	Use the Arrow Keys to Move Within Tab Groups	1–60
1–41	Putting Information in Title Bars	1–63
2–1	Order of Menus in the Menu Bar	2–2
2–2	Sample Menu Bar with Application-Specific Items	2–2
2–3	A Menu Map	2–5
2–4	Ellipses with Cascade Buttons	2–6

2–5	A Menu with a Disabled Menu Item	2–7
2–6	Pop-up Menu	2–9
2–7	Pop-up Menu with a Submenu	2–12
2–8	A File Menu and Its Menu Items	2–14
2–9	A ‘‘Save File’’ Question Message Box	2–16
2–10	An ‘‘Open in New Window’’ Check Box in an Open File Dialog Box	2–17
2–11	A Save As Dialog Box That Allows Users to Save in Different File Formats	2–19
2–12	A Revert Dialog Box	2–20
2–13	A Selected Menu and Its Menu Items	2–22
2–14	An Edit Menu and Its Menu Items	2–23
2–15	A View Menu and Its Menu Items	2–27
2–16	A Help Menu and Its Menu Items	2–28
2–17	A Windows Menu	2–30
3–1	An Example Options Menu	3–2
3–2	Range of Customization	3–3
3–3	Feedback Area Labeled "Sample Text"	3–7
3–4	Using One Menu to Customize Text	3–9
3–5	Using Several Menus to Customize Text	3–10
3–6	Using a Dialog Box to Customize Text	3–11
4–1	Color Picker Model	4–6
5–1	Create a Hierarchical Tree	5–3
5–2	Create Task-Oriented Topics	5–5
5–3	Write an Overview for the Application	5–6
5–4	Additional Topics in a Help Window	5–8
5–5	Put Task-Oriented Information First	5–9
5–6	Number Steps	5–10
5–7	Use Short Sentences and Paragraphs	5–12
5–8	Help Window	5–15
5–9	Comparison of Help Widget and DECwindows Motif Help System Windows	5–19
6–1	Print Widget’s Primary Dialog Box	6–2
6–2	Print Widget’s Secondary Dialog Box on OpenVMS Systems	6–3
6–3	Erasing Tab Groups Users Do Not Need	6–5
6–4	Using the Outline Format to Show SVN Hierarchy	6–6

6–5 SVN Tree Format 6–7
6–6 SVN Column Format 6–8
6–7 SVN Index Window 6–9
6–8 Components in an SVN Entry 6–10
6–9 Expanded and Collapsed SVN Icon States 6–11
6–10 SVN Selection with a Full Selection Line 6–13
6–11 SVN Selection with a Limited Selection Line 6–14
6–12 SVN with a Variable-Length Selection Line 6–15
6–13 SVN Menu Items in the View Menu 6–16

Tables

5–1 Comparison of Help Widget and DECwindows Motif Help System Features 5–18
A–1 Differences Between LK201 and LK401 Keyboards A–1
A–2 Modifiers for DECwindows Motif Keyboard Bindings A–2
A–3 General Bindings A–3
A–4 Bindings for General Navigation A–5
A–5 Bindings for Menu Navigation A–6
A–6 Bindings for Text Navigation A–7
A–7 Bindings for Window Navigation A–8
A–8 Bindings for Activation and Editing A–9
A–9 Mouse Bindings for DECwindows Motif (Three-Button Mouse) A–12
A–10 Motif Accelerators for the Window Menu A–13
A–11 Suggested Accelerators for the File Menu A–13
A–12 Motif Accelerators for the Edit Menu A–14

About This Manual

Audience

This manual provides supplemental information to all who have read the *OSF/Motif Style Guide*. It is designed specifically for programmers and interface designers who develop DECwindows Motif applications and who seek to present a uniform and usable software interface that is consistent with other DECwindows Motif applications.

This manual is also for technical communicators such as writers and editors who need to be conversant with the Motif user interface style.

The *OSF/Motif Style Guide* provides a general set of guidelines, while the *DECwindows Companion to the OSF/Motif Style Guide* provides more detailed explanations and more illustrations to help you create a consistent user interface for your application. These detailed explanations and illustrations are recommendations offered to further the consistency and usability of user interfaces. Failing to follow them will not jeopardize the compliance of your user interface to the OSF/Motif style; however, it may jeopardize consistency across applications.

Organization

The *DECwindows Companion to the OSF/Motif Style Guide* covers the following topics:

- Chapter 1 provides guidelines for designing dialog boxes.
- Chapter 2 provides guidelines for designing menus and menu items.
- Chapter 3 provides guidelines for designing an interface to allow customization of an application's user interface.
- Chapter 4 provides guidelines for using color in your application.
- Chapter 5 provides guidelines for designing and writing online help.

- Chapter 6 describes Digital-specific widgets, including ColorMix, Help, Print, and Structured Visual Navigation (SVN).
- Appendix A provides a list of keyboard and mouse bindings and their associated Digital bindings.
- Appendix B provides translations of common interface objects into ten European languages.
- Appendix C provides a list of sources for further information.

Related Documents

For additional information, refer to the following manuals:

- *OSF/Motif Style Guide*
- *OSF/Motif Programmer's Guide*
- *OSF/Motif Programmer's Reference*
- *DEC OSF/1 DECwindows User's Guide*
- *Using DECwindows Motif for OpenVMS*
- *DECwindows Motif Guide to Applications Programming*
- *DECwindows Extensions to Motif*

The printed version of the DEC OSF/1 documentation set is color coded to help specific audiences quickly find the books that meet their needs. (You can order the printed documentation from Digital.) This color coding is reinforced with the use of an icon on the spines of books. The following list describes this convention:

Audience	Icon	Color Code
General Users	G	Teal
System Administrators	S	Red
Network Administrators	N	Yellow
Programmers	P	Blue
Reference Page Users	R	Black

Some books in the documentation set help meet the needs of several audiences. For example, the information in some system books is also used by programmers. Keep this in mind when searching for information on specific topics.

The *DEC OSF/1 Documentation Overview* provides information on all of the books in the DEC OSF/1 documentation set.

Reader's Comments

Digital welcomes your comments on this or any other DEC OSF/1 manual. A Reader's Comment form is located in the back of each printed DEC OSF/1 manual, and online in the following location:

`/usr/doc/readers_comment.txt`

You can send your comments in the following ways:

- Internet electronic mail: `readers_comment@zk3.dec.com`
- Fax: 603-881-0120 Attn: UEG Publications, ZK03-3/Y32
- Mail:

  Digital Equipment Corporation
  UEG Publications Manager
  ZK03-3/Y32
  110 Spit Brook Road
  Nashua, NH 03062-9987

  The Readers Comment form located in the back of each printed manual is postage paid if you mail it in the United States.

If you have suggestions for improving particular sections or find any errors, please indicate the title, order number, and section numbers. Digital also welcomes general comments.

Conventions

The following conventions are used in this manual:

mouse	The term *mouse* is used to refer to any pointing device, such as a mouse, a puck, or a stylus.
MB1 (BSelect) MB2 (BDrag) MB3 (BMenu)	MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button. (The buttons can be redefined by the user.)
Ctrl+x	A sequence such as Ctrl+x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.

numbers	Unless otherwise noted, all numbers in the text are assumed to be decimal. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.
UPPERCASE	Although uppercase letters are used in discussions of accelerators and mnemonics (for example, Ctrl+E), they are not meant to indicate case sensitivity when included in applications.

Note

Your screen objects, dialog boxes, and primary windows may not always appear exactly the same as those in this book because the examples of user interface objects in this document are an artist's rendition, and because user interface objects appear differently depending on whether Motif is being displayed on a monochrome or color terminal and on the colors set for a color terminal.

1
Designing Dialog Boxes

Dialog boxes display messages and solicit user input. This chapter provides guidelines for designing effective dialog boxes.

Have your application provide dialog boxes when the following conditions exist:

- When a task requires more complicated interaction than is available in a menu; that is, when a single menu cannot hold all the options
- When users want to perform related actions likely to be used together, but not necessarily at the same moment (such as changing both font and point size)
- When users want the controls to remain available while they interact with other parts of the application

To design dialog boxes effectively, you must know how to do the following:

- Determine the type and placement of a dialog box
- Use window decorations
- Create push buttons
- Use standard message dialog boxes
- Create a logical and appealing layout of controls

This chapter also discusses the following topics related to implementing dialog boxes:

- Presenting text in dialog boxes and other screen objects
- Creating title bars
- Using the working dialog box and the wait cursor

1.1 Determining the Type and Placement of a Dialog Box

A DECwindows application consists of one or more primary windows and their associated dialog boxes. Dialog boxes are also known as secondary windows. Under the Motif window manager, secondary windows have the following characteristics:

- They cannot be moved behind their parents unless you use specific button and key bindings to move them.
- They are minimized and restored with their parents; they cannot be individually minimized and restored.

All dialog box characteristics are provided by the DECwindows Motif toolkit; that is, you do not have to do any coding to have secondary windows minimize and restore with their parents as long as the dialog box is a subclass of the BulletinBoard or Form widget.

1.1.1 Types of Dialog Boxes

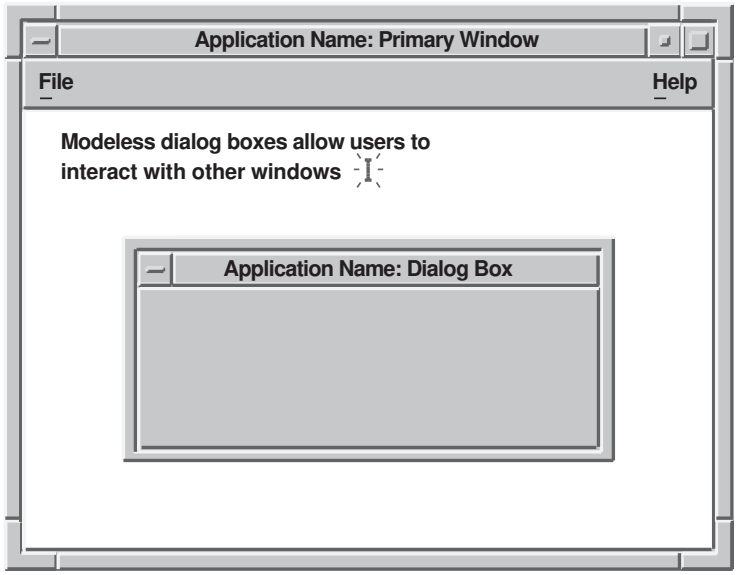
There are two types of dialog boxes: modeless and modal.

Modeless

Modeless dialog boxes allow users to continue working in any of the application's windows. For example, Figure 1–1 shows an application in which users can type in a primary window even though a dialog box is displayed. Modeless dialog boxes allow users the most flexibility in structuring their work.

Make most of your application's dialog boxes modeless.

Figure 1–1 Modeless Dialog Boxes Allow Users to Work in Other Windows



ZK–2592A–GE

Programming Hint

Dialog boxes are modeless by default. To use a modeless dialog box, set the XmDialogStyle to XmDIALOG_MODELESS.

To make a dialog box modal, use an XmForm widget or a subclass of XmForm. Then set the XmDialogStyle to one of the following values:

- XmDIALOG_PRIMARY_APPLICATION_MODAL
- XmDIALOG_FULL_APPLICATION_MODAL
- XmDIALOG_SYSTEM_MODAL

You can also use XmBulletinBoard to make a dialog box modal. However, XmBulletinBoard should be used only in cases that require handling special geometry information.

Note that XmDIALOG_PRIMARY_APPLICATION_MODAL has replaced XmDIALOG_APPLICATION_MODAL. New applications should use only XmDIALOG_PRIMARY_APPLICATION_MODAL.

Modal

Modal dialog boxes prevent certain user interactions while the dialog box is displayed. You can make your dialog boxes either primary application modal or full application modal. Choosing which to use depends on the nature of your application and a given dialog box. An explanation of each dialog box follows:

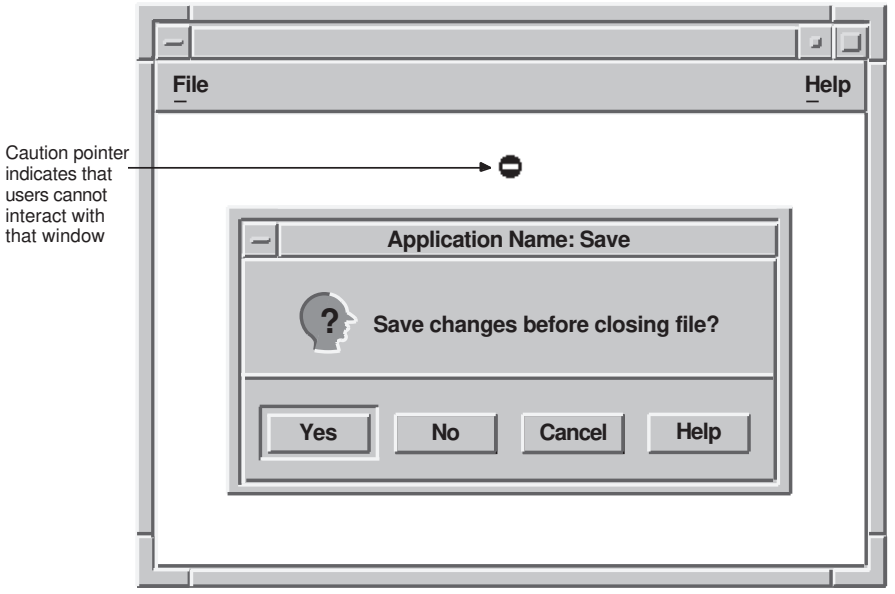
- Primary application modal

  A primary application modal dialog box prevents user activity in any of its ancestors. For example, with Bookreader, the Switch Library dialog box is a primary application modal dialog box; when it is displayed, users cannot interact with the Library window. However, they can still interact with the other Bookreader primary application windows. Specifically, they can still move to the next topic in a book they had previously opened.

  When users move the pointer from the dialog box into any ancestor window, your application should change the pointer to the caution pointer shape to indicate that users cannot do anything in the parent window until they have dismissed the modal dialog box. The caution pointer is the international symbol for *do not enter*. Figure 1–2 shows an example of a caution pointer over the parent window.

  You are responsible for changing the pointer to the appropriate pointer over the corresponding windows.

Figure 1–2 Caution Pointer with a Modal Dialog Box



Programming Hint

To change pointers, you must create each pointer, and set or reset the pointer when it moves from one window to another.

To change the pointer, follow these steps:

1. Include two files.

   On OpenVMS systems, include:

   ```
   decw$cursor.h
   decw$include/cursorfont.h
   ```

   On Digital UNIX systems and Microsoft Windows NT systems, include:

   ```
   Xm/decwcursor.h
   X11/cursorfont.h
   ```

2. Create the cursor. For example, to create the caution pointer you might use the Digital DXmCreateCursor routine and add the following code to your C program:

```
caution_pointer = DXmCreateCursor (toplevel, decw$c_INACTIVE_CURSOR);
XDefineCursor (XtDisplay(toplevel), XtWindow(toplevel), caution_pointer);
XFlush (XtDisplay(toplevel));
```

3. Reset the default cursor.

   For example, add the following code to your C program:

```
XUndefineCursor (XtDisplay (toplevel), XtWindow (toplevel));
```

- Full application modal

  A full application modal dialog box prevents user activity in any other window of the application. For example, the Exit menu item on your application might generate a full application modal dialog box that asks users if they want to save the file before exiting. Users cannot interact with any part of the application until they have answered the question.

  When users move the pointer from a dialog box into any of that application's windows, the application changes the pointer to the caution pointer shape. This shape indicates that users cannot do anything in any of the application's windows until they have dismissed the modal dialog box. With a full application modal dialog box, the caution pointer would appear not only over the parent window, as shown in Figure 1–2, but also over any other window belonging to that application.

  If users try to type or use the mouse in any of the application's windows, the application will emit an audible signal to tell users that they are not allowed to interact with other windows belonging to that application.

  After users dismiss the modal dialog box, they can once again interact in the application's windows.

1.1.2 Placing Dialog Boxes

Once you have decided which of your dialog boxes to make modal and which to make modeless, you must decide where to place them on the screen and whether or not to nest them.

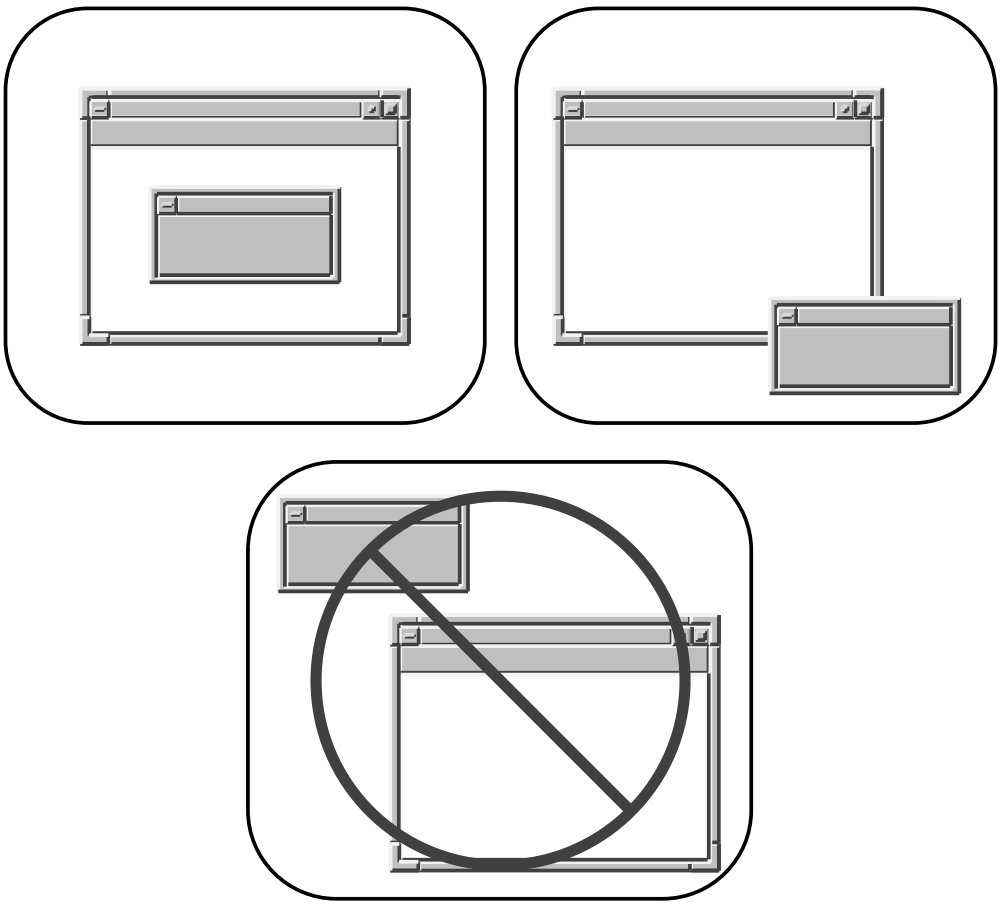
1.1.2.1 Placing a Single Dialog Box

For dialog boxes that do not require users to see or use the information in the parent window, make the dialog box appear in the center of the primary window's work region, as shown in Figure 1–3. This location requires a minimum amount of mouse movement and a minimum shift of eye focus for users.

If, however, the dialog box does require users to be able to see or use the information in the parent window, place the dialog box to one side or below the parent window. The most appropriate location of the dialog box depends on the application.

Do not place dialog boxes in the upper left corner of the screen, that is, at 0,0, as shown in Figure 1–3. From the user's point of view, this placement is arbitrary and has little or no meaningful relationship to where they have placed the application's primary window.

Figure 1–3 Place Dialog Boxes Appropriately



ZK–2600A–GE

Programming Hint

The toolkit places dialog boxes in the center of the parent window if the window is an XmBulletinBoardDialog or an XmFormDialog widget. However, if the window is not a subclass of these widgets, the dialog box is placed in the upper left of the screen at 0,0. In such a case, you must set the x-coordinate and y-coordinate for the dialog box manually.

When users move a dialog box, respect that placement whenever that box is used again. For example, if a dialog box appears in the center of the parent window by default, but users move it below the parent, the next time users call up that dialog box, have it display where they had last placed it; that is, below the primary window.

Save the dialog box placement with respect to its placement on the screen rather than its placement in relation to the parent. For example, if users move the parent window in the previous example from the center of the screen to the top right, and then call up the dialog box again, make the dialog box display where the user last placed it instead of just below the parent window.

To retain these settings, use XtUnmanageChild to unmanage the dialog box when the user dismisses it. Then, when the user performs an action that causes that dialog box to reappear, call XtManageChild to remap it. The dialog box placement and size will be the same as they were before the dialog box was dismissed.

1.1.2.2 Nesting Multiple Dialog Boxes

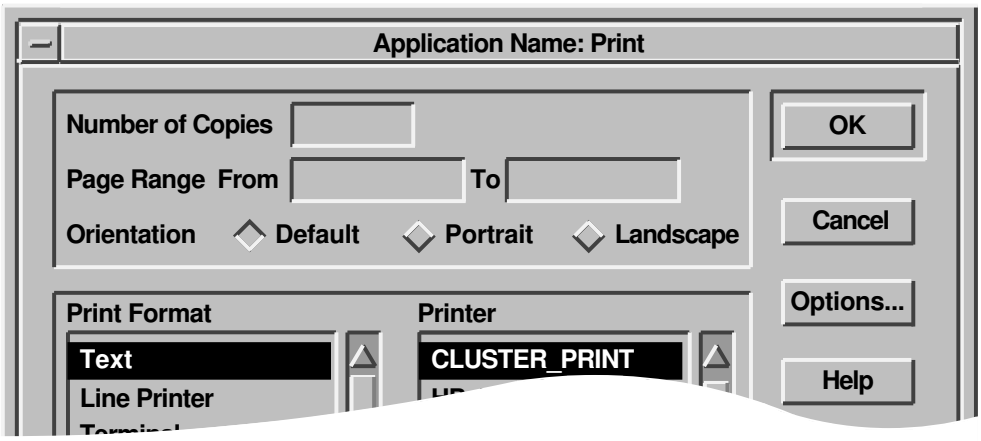
When the amount of information is excessive for a single dialog box, create multiple dialog boxes. This method is called nesting. Organize nested dialog boxes by putting the necessary and common functions in the first dialog box, and the less common functions in subsequent boxes.

For the user's convenience, place the most important and most frequently used controls in the first dialog box. Involve users in the design of dialog boxes so you will know which controls they will use the most. Allow users to make basic choices in the first dialog box; do not require them to go to the secondary dialog box to finish basic operations. Also, make sure that the options available at any time are obvious to users.

If the dialog box contains more than five to seven optional actions in addition to activating a push button, consider nesting. An action can be filling in a text-entry control, choosing a radio button in a set of radio buttons, choosing one or more items in a list box, and so on. Limit the levels of dialog boxes to three.

Invoke the second (nested) dialog box from a clearly labeled push button on the primary box. If your dialog box invokes only one secondary dialog box, use a push button with an Options... label. See Figure 1–4.

Figure 1–4 Using an Options... Button to Indicate a Secondary Dialog Box

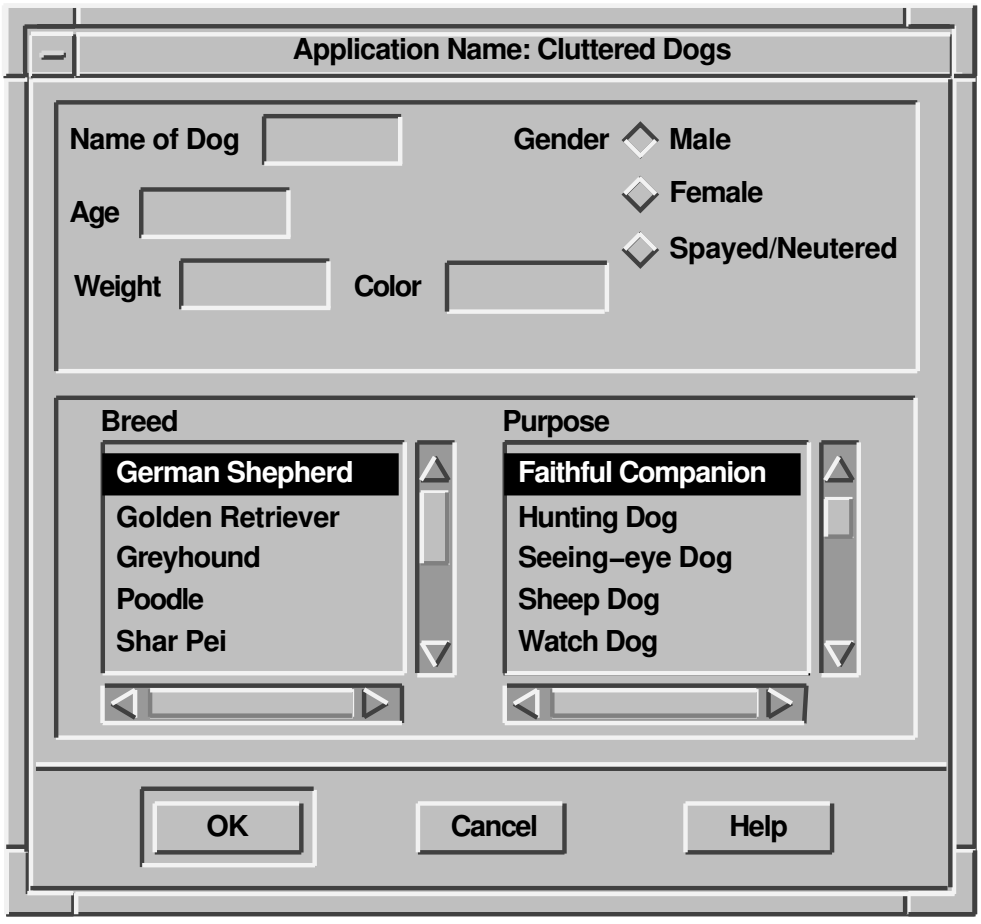


Use an ellipsis (. . .) to indicate that pushing the button invokes a dialog box. By convention, do not use an ellipsis with the Help push button.

You can have your application generate more than one secondary dialog box from one primary dialog box. If you do, however, clearly label the push buttons that invoke the dialogs. For example, you should label one Patterns... and give the other the Templates... label.

When you are deciding which controls to nest, design example dialog boxes and ask potential users what they think is the best arrangement, and which controls belong in the primary dialog box. For example, Figure 1–5 shows a dialog box that contains seven steps in addition to activating the push buttons. (Note that the three radio buttons for gender constitute one step.) It looks cluttered, and could intimidate users.

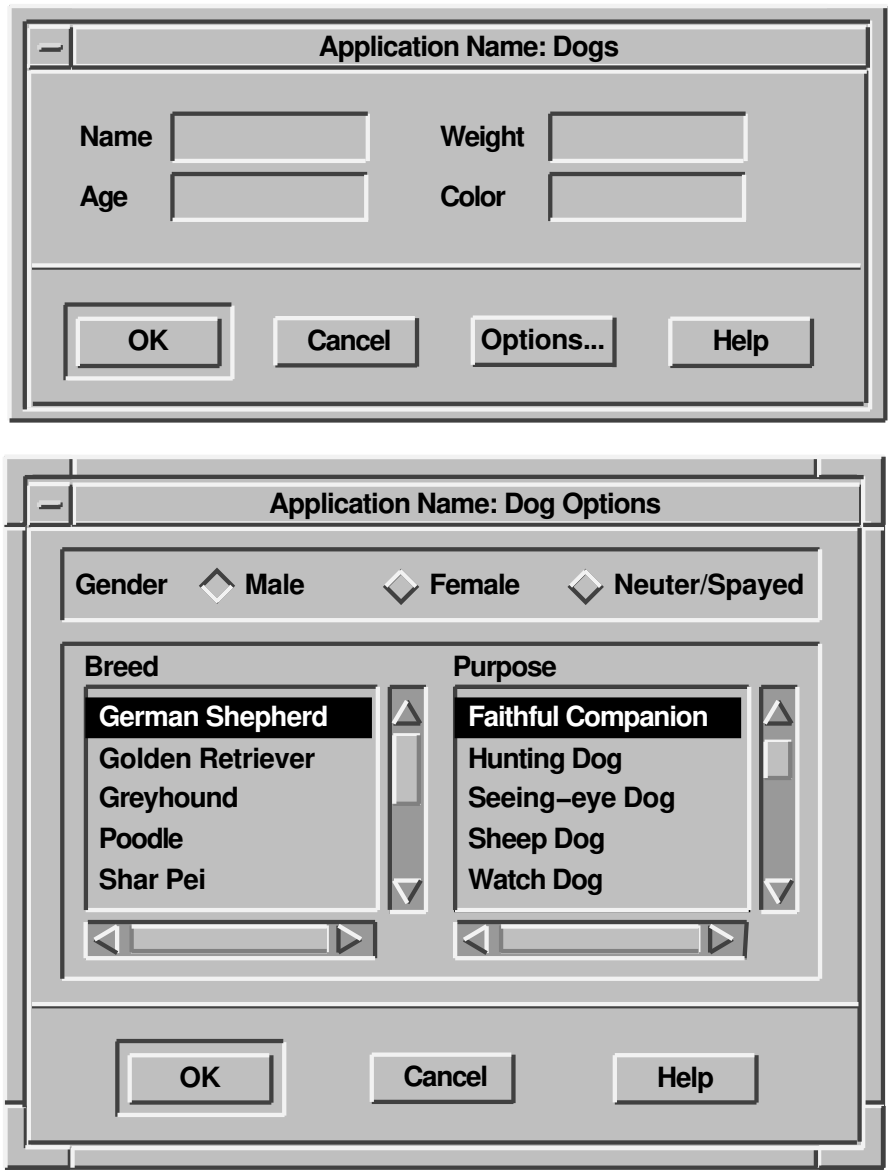
Figure 1–5 A Dialog Box with Too Many Steps



ZK–2769A–GE

You can create a nested dialog box so that the primary dialog box contains only the steps that users perform frequently, and the secondary dialog box contains other, less frequently used steps. An example of the reorganized box is shown in Figure 1–6.

Figure 1–6 Nesting Used to Redesign a Dialog Box with Too Many Steps



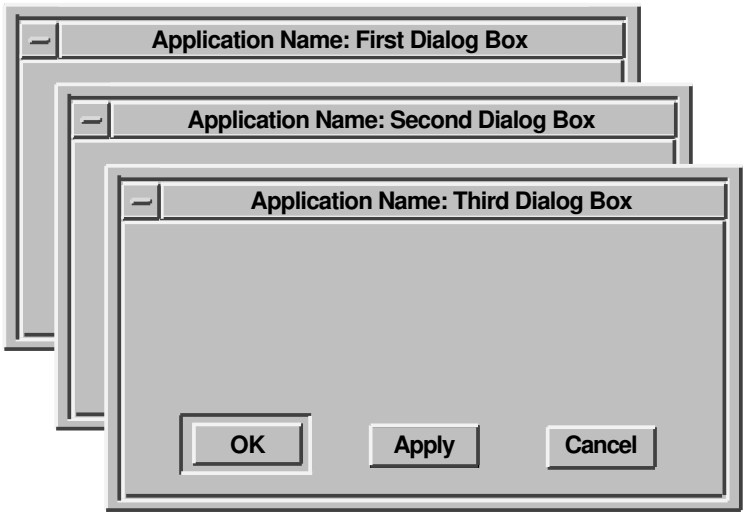
ZK–2770A–GE

There are two techniques for placing nested dialog boxes:

- Align the push buttons of nested dialog boxes one on top of the other. For example, place the nested box's OK button on top of the first box's OK button, so when users are dismissing the dialog boxes, they will not have to move the mouse.
- Place the nested dialog box to the right and below the title of the primary dialog box in a cascading order (in left-to-right environments), as shown in Figure 1–7.

Using both techniques together assumes that the nested dialog box is smaller than the primary dialog box. If it is not possible to make the nested box smaller than the first box, place the push buttons of the nested box as close as possible to the push buttons on the first box.

Figure 1–7 Nesting Dialog Boxes



ZK–2529A–GE

Programming Hint

Remember, if your secondary dialog box is not a subclass of an XmBulletinBoard, you need to specify the x-coordinate and y-coordinate of the dialog box.

1.2 Using Window Decorations

The window manager provides several decorations to any window. All windows receive the following decorations automatically:

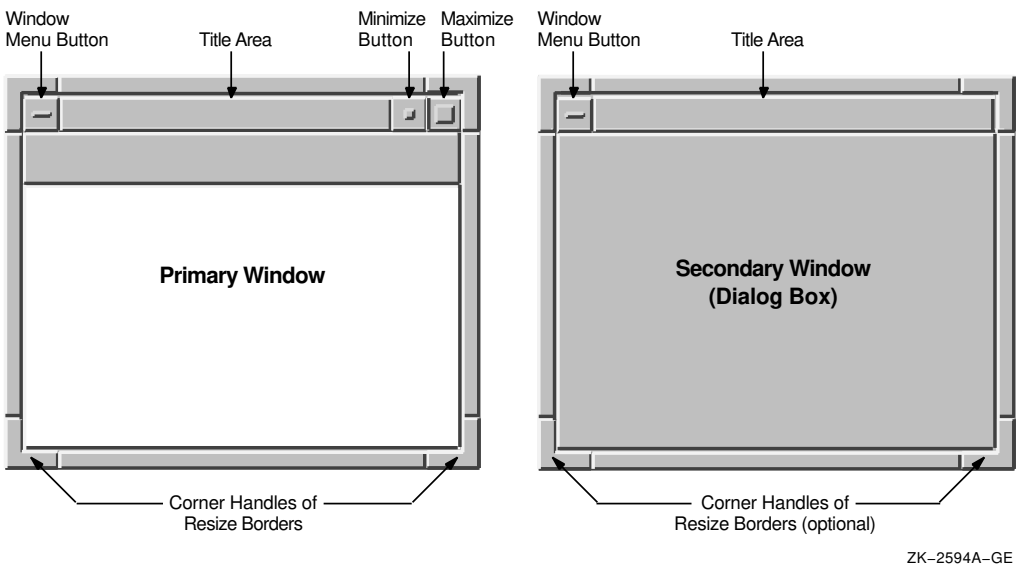
- Resize borders. Notice that the corner handles on the border resize the window in both x-coordinate and y-coordinate, while the side borders resize the window only along one coordinate.
- A window menu button. Pressing the window menu button causes the window menu to appear. The window menu typically contains items for restoring, moving, sizing, minimizing, maximizing, lowering, and closing the window.
- A title area.
- A minimize button. Pressing the minimize button shrinks the window into an icon.
- A maximize button. Pressing the maximize button makes the window as large as it can be.

Give dialog boxes only the following decorations:

- A window menu button.
- A title area.
- Resize borders (optional). See Section 1.2.2.

These decorations are the default. Figure 1–8 compares the window decorations for primary windows to the decorations for secondary windows.

Figure 1–8 Window Decorations on Primary and Secondary Windows



1.2.1 Titles in Dialog Boxes

Give each dialog box a title. The elements of a title are as follows:

- The application name (followed by a colon)
- A space
- The name of the menu item or push button from which it was invoked

See Figure 1–9 for examples of titles.

In some cases you may want to include a menu name with the menu item. For example, if users pull down the Options menu and select the General menu item, give the associated dialog box the following title:

```
Mail: General Options
```

In this example, Mail is the name of the application, General is the name of the menu item, and Options is the name of the menu.

Figure 1–9 Appropriate and Inappropriate Titles in Dialog Boxes

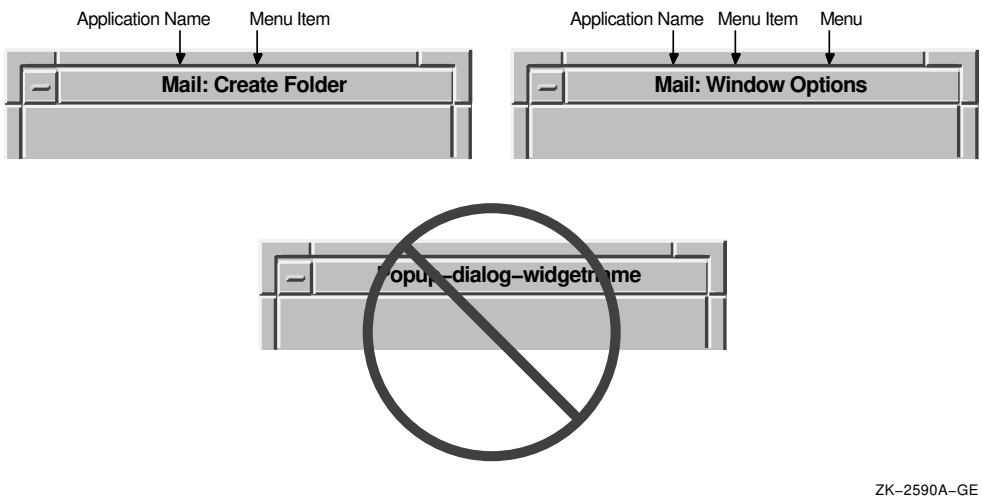


Figure 1–9 also shows an example of a dialog box title that is inappropriate because it is the name of the widget. A widget name means nothing to users, so do not use one as the title of a dialog box.

Programming Hint

Use the XmNdialogTitle resource to set the title of a dialog box.

1.2.2 Resizing Dialog Boxes

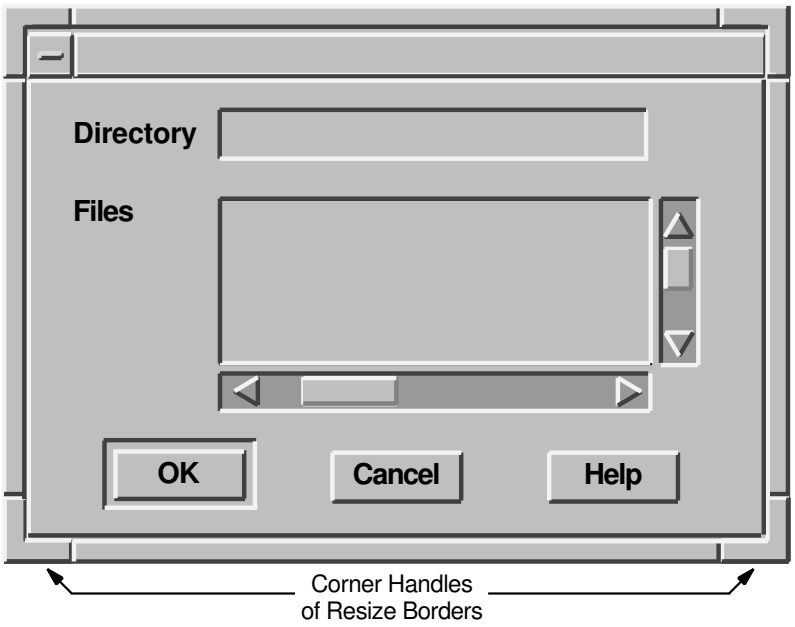
Use the following guidelines to determine when to include resize borders on your dialog box:

- When your dialog box contains text-entry fields, list boxes, or other controls that are not always a fixed size, provide resize borders. The dialog box in Figure 1–10 has resize borders to allow users to see entire file names and to see everything they type in the text-entry field.
- When your dialog box contains only fixed-size controls (such as push buttons, check buttons, and radio buttons), make the dialog box big enough to display all the components. That is, design your boxes so users do not need to resize the box, and then do not include resize borders.

Programming Hint

To create a dialog box without resize borders, set XmNnoResize=True.
To create a dialog box with resize borders, set XmNnoResize=False.
This resource is inherited from the XmBulletinBoard widget.

Figure 1–10 Use Resize Borders with Text-Entry Fields or List Boxes



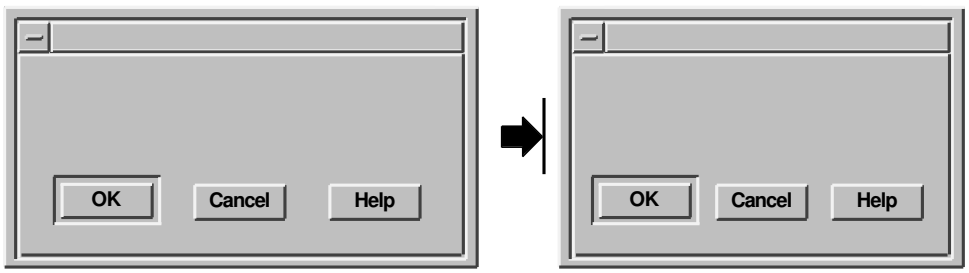
ZK–2668A–GE

If you provide resize borders, here are a few guidelines to follow:

- When users make the dialog box smaller, do the following:
  - Set a minimum size for the dialog box. That is, do not allow users to make the dialog box so small that it is no longer useful. Users should be able to see all of the controls, and activate the push buttons when the dialog box is at the minimum size.

– Move the controls closer together so that they can still be seen and activated. See Figure 1–11 for an example.

Figure 1–11 Move Push Buttons Closer Together



ZK–2601A–GE

Programming Hint

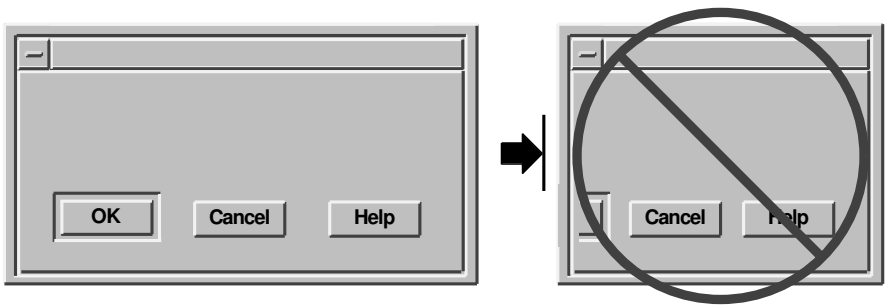
Digital provides a routine (DXmFormSpaceButtonsEqually) that helps to move push buttons closer together. For more information, see Section 1.3.

Use the following code as an example of how to set a minimum size for your dialog box. In the following example, the minimum size is 100; you need to determine an acceptable minimum size for your dialog box:

```
Arg arglist[2];
XtSetArg(arglist[0], XmNminWidth,  100);
XtSetArg(arglist[1], XmNminHeight, 100);
XtSetValues(XtParent(dialog_box), arglist, 2);
```

– Do not truncate the information that was in the box, as shown in Figure 1–12. When the dialog box is truncated, important push buttons or other controls may disappear.

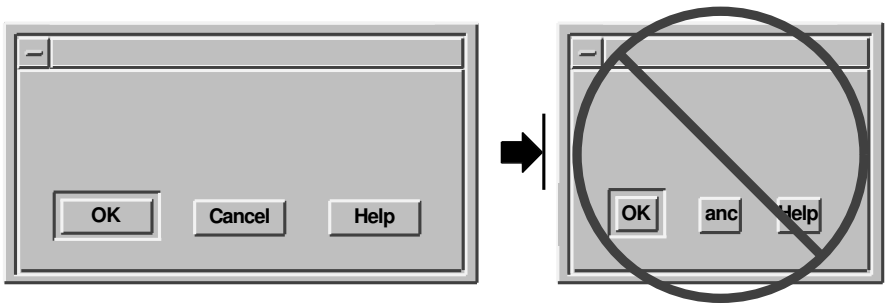
Figure 1–12 Do Not Truncate Push Buttons



ZK–2664A–GE

– Do not truncate the labels in the push buttons, as shown in Figure 1–13. Users cannot read the labels and might not know which push buttons to activate.

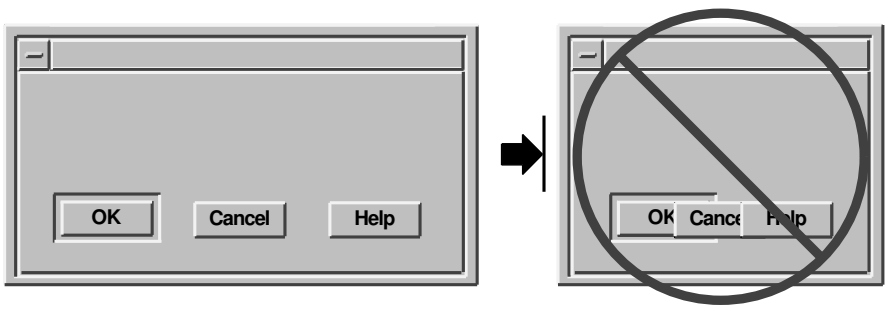
Figure 1–13 Do Not Truncate Labels in Push Buttons



ZK–2665A–GE

– Do not overlap the push buttons in a minimized dialog box, as shown in Figure 1–14. It is difficult for users to tell where to click in order to activate which button.

Figure 1–14 Do Not Overlap Push Buttons



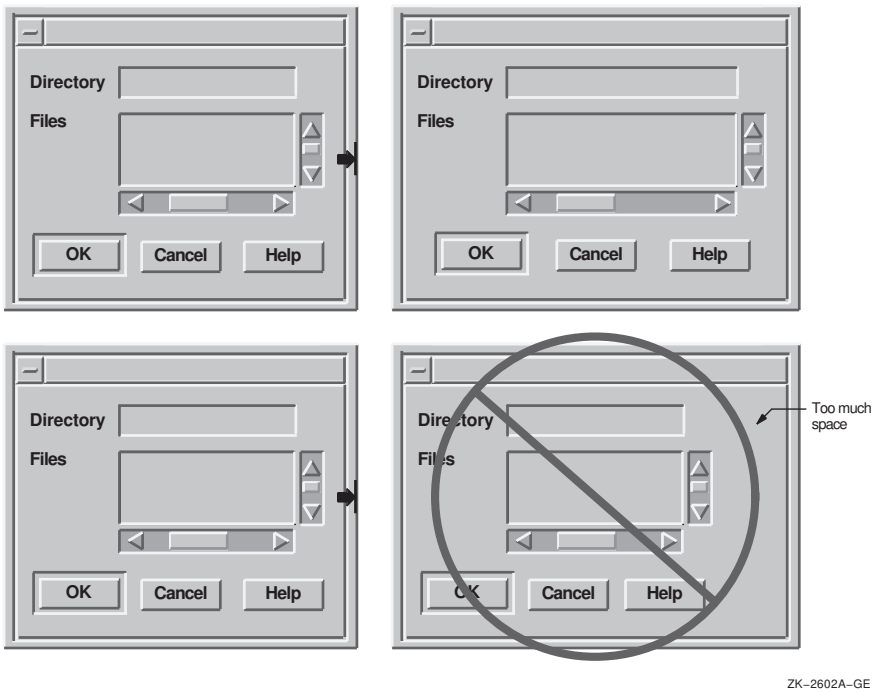
ZK–2666A–GE

- When users increase the size of a dialog box, have the size and layout of the dialog box increase proportionally. Do not add blank space to the area of the dialog box that is larger, as shown in Figure 1–15.

Programming Hint

To increase the size and layout of controls within a dialog box, use an XmForm widget or an XmFormDialog widget and specify both left and right attachments for each control that needs to expand and contract. Use DXmFormSpaceButtonsEqually to make the push buttons expand and contract properly.

Figure 1–15 Increase Size and Layout of Dialog Box Proportionally



1.3 Creating Push Buttons

This section contains guidelines and programming hints to help you create and arrange push buttons for dialog boxes.

1.3.1 Default Push Button

Give every dialog box a default push button.

The default push button allows users to continue working by pressing the Return key (thereby activating the default push button). When you indicate the default push button, Motif automatically gives it extra highlighting as shown in Figure 1–16.

Programming Hint

To enable the default push button, set XmNdefaultButton to the name of your default button widget in the UIL file.

1.3.2 Cancel Button

Give a dialog box a Cancel push button only when appropriate.

Programming Hint

To enable F11 (the Escape key) as the Cancel key, set XmNcancelButton to the name of the cancel button widget in the UIL file.

1.3.3 Labels in Push Buttons

Center labels in push buttons, as shown in Figure 1–16. Centering occurs by default, so do not override it.

When using the standard push buttons in a dialog box, try to preserve one of the following orders:

- OK Apply Reset Default Cancel Help
- OK Cancel Options... Help
- Yes No Cancel Help

The OK button should always be the first button; the Help button should always be last. The Cancel button should be the next to the last button, with one exception. If there is an Options... button or another button that brings up another dialog box, you should place that button between the Cancel and Help buttons.

1.3.4 Push-Button Arrangement

Arrange push buttons horizontally across the bottom of the dialog box (as shown in Figure 1–16) or vertically near the top right corner of the box (as shown in Figure 1–17).

1.3.4.1 Horizontal Push-Button Arrangement

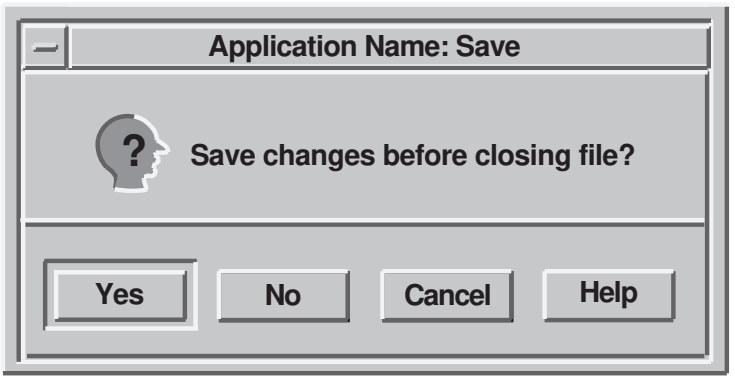
Two methods are recommended for arranging buttons horizontally:

- Space the buttons equally across the entire dialog box.
- Center the buttons together in the dialog box.

If the dialog box is narrow or there are many buttons, either method works well. However, if the dialog box is wide or if there are few buttons, you should center all the buttons, leaving equal left and right margins. This arrangement minimizes the mouse movement necessary to move between the buttons.

In the first method, you leave equal amounts of space between the buttons and the margins. That is, you make the space from the left margin to the first button equal to the space between each of the buttons as well as the space between the last button and the right margin. Figure 1–16 shows equally spaced push buttons at the bottom of a dialog box.

Figure 1–16 Equally Spaced Push Buttons at the Bottom of a Dialog Box



ZK–2540A–GE

Programming Hint

Digital supplies a routine, DXmFormSpaceButtonsEqually, that sets push buttons in a form widget so they are equally spaced and sized. Use the following code example to help you:

```
DXmFormSpaceButtonsEqually (parent, widget_list, num_widgets)

     Widget     my_form;
     Widget     *widget_list;
     Cardinal   num_widgets;
```

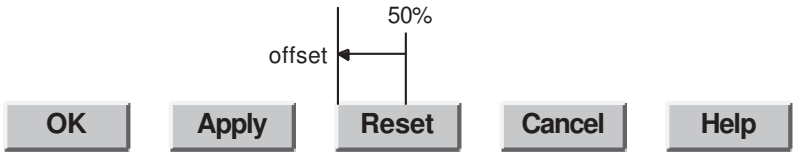
- The parent is the widget ID of the form widget containing the buttons.
- The widget_list is an array of widget IDs of the buttons to be changed.
- The num_widgets parameter is the number of widgets in the widget list.

In the second method, to center the buttons, you use attachments to attach the center button to a relative position in the dialog box.

Use the following Programming Hints for centering an odd number of buttons or an even number of buttons.

Programming Hint

This programming hint works for an odd number of buttons. The example has five buttons: OK, Apply, Reset, Cancel, and Help.



ZK–6374A–GE

To center the buttons, first attach the left side of the Reset button to 50% across the dialog box. Then set XmNleftOffset to be one half of the size of the button:

```
reset_args : arguments
    {
        XmNlabelString = "Reset";
        XmNwidth = 1500;
        XmNleftAttachment = XmATTACH_POSITION;
        XmNleftPosition = 50;
        XmNleftOffset = -750;     /* font units */
    };
```

If you are concerned about internationalization or font customization by the user, you can modify the layout at run time. Follow these steps:

1. Traverse all the buttons and calculate the size of the largest button.
2. Set all button widths to be the size of the largest button plus a little space. This extra space ensures that the text does not run into the border of the button. If the text for all of the buttons is short, you should set the minimum size of the button approximately equal to the Cancel button width.
3. Set the XmNleftOffset of the middle button to half the size of the buttons.

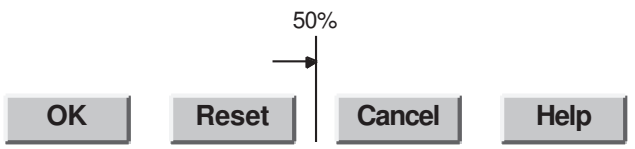
4. Attach the other buttons to the left or right side of the Reset button as is appropriate. You should leave an offset of about 200 font units between each button:

```
left_button_args : arguments
    {
        XmNwidth = 1500;
        XmNrightAttachment = XmATTACH_WIDGET;
        XmNrightOffset = 200;     /* font units */
    };

right_button_args : arguments
    {
        XmNwidth = 1500;
        XmNleftAttachment = XmATTACH_WIDGET;
        XmNleftOffset = 200;     /* font units */
    };
```

   See the Programming Hint for extra-long push buttons in Section 1.3.5 if your buttons are not approximately the same size.

Programming Hint

This programming hint works for an even number of buttons. The example has four buttons: OK, Reset, Cancel, and Help.



ZK–6375A–GE

To center the buttons, first place the Reset button near the center, attaching the right side of the button to 50% across the dialog box. Then set XmNrightOffset to half the size of the space between the buttons:

```
reset_args : arguments
    {
        XmNlabelString = "Reset";
        XmNwidth = 1500;
        XmNrightAttachment = XmATTACH_POSITION
        XmNrightPosition = 50;
        XmNrightOffset = 100;     /* font units */
     };
```

If you are concerned about internationalization or font customization by the user, you can modify the layout at run time. Follow these steps:

1. Traverse all the buttons and calculate the size of the largest button.
2. Set all button widths to be the size of the largest button plus a little space. This extra space ensures that the text does not run into the border of the button. If the text for all of the buttons is short, you should set the minimum size of the button approximately equal to the Cancel button width.
3. Attach the other buttons to the left or right side of the Reset button as is appropriate. You should leave an offset of about 200 font units between each button:

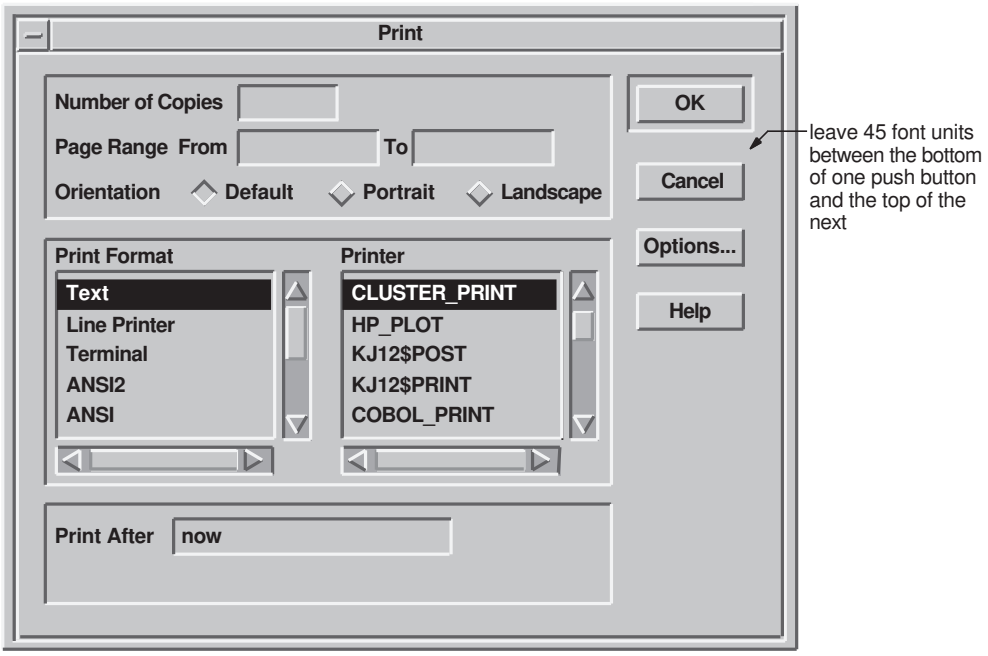
```
left_button_args : arguments
    {
        XmNwidth = 1500;
        XmNrightAttachment = XmATTACH_WIDGET;
        XmNrightOffset = 200;     /* font units */
    };

right_button_args : arguments
    {
        XmNwidth = 1500;
        XmNleftAttachment = XmATTACH_WIDGET;
        XmNleftOffset = 200;     /* font units */
    };
```

   See the Programming Hint in Section 1.3.5 for extra-long push buttons if your buttons are not approximately the same size.

1.3.4.2 Vertical Push-Button Arrangement

If you arrange the push buttons vertically, place the default push button in the top right position. Then align the other push buttons under it. Leave a space of 45 font units of the default menu font between the bottom of one push button and the top of the next. Figure 1–17 shows a vertical arrangement of push buttons.

Figure 1–17 Vertically Arranged Push Buttons at the Side of a Dialog Box



ZK–2516A–GE

1.3.5 Push-Button Dimensions

Give all push buttons in your dialog box the same dimensions, even though some labels might be longer than others.

However, if you have one label that is substantially longer than the others, give the shorter labels smaller push buttons, providing all the smaller buttons will have similar dimensions, as shown in Figure 1–18.

When the buttons are arranged horizontally, make the larger button wider, but not taller, than the others. When the buttons are arranged vertically, make the larger button taller, but not wider, than the others. See Figure 1–18.

Programming Hint

If your application will be internationalized, the extra-long push button will be translated, and may no longer be the longest push button.

For international applications, put the resources for sizing and positioning in UIL include files to simplify translation. For example, if you are using a fixed size for your push buttons, define it in a UIL include file and comment it thoroughly. You could define most buttons with a literal named:

```
normal_width
```

Define the long one with a literal named:

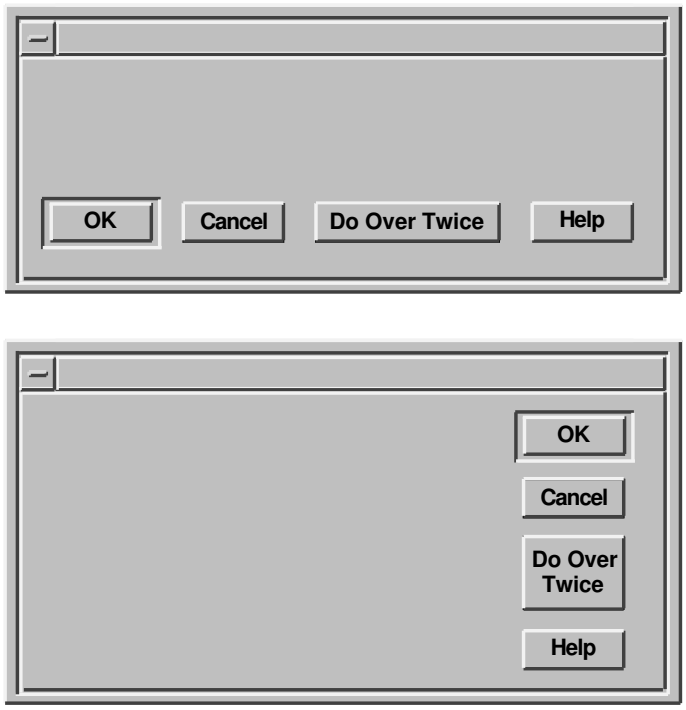
```
long_width
```

Then set their values as follows:

```
value
    normal_width = 50;
    long_width = 100; /* width of the long button*/
```

Reference these literals when defining the XmNwidth resources of the buttons. At internationalization time, if the labels in the buttons change width, change the value of the literals.

Figure 1–18 How to Handle a Push Button That Is Larger Than Others



ZK–2667A–GE

1.3.6 Help Push Button

Include a Help push button in all dialog boxes.

When users click on the Help push button, they should receive an overview of the dialog box. You might also include additional topics that provide detailed information on how to use each control within it. (Some simple message boxes may not require a Help push button.) For more information on designing online help, see Chapter 5.

If your application uses warning or error dialog boxes, provide a Help push button in the dialog box if you cannot describe the error and suggest a solution in one or two lines of text. Have the Help window describe the error or warning displayed in the dialog box and provide a suggestion for solving the problem.

1.4 Using Standard Message Dialog Boxes

Message dialog boxes provide users with information, such as error messages, warnings, and the status of the application. Some ask questions.

Like other dialog boxes, message boxes have a windows menu and a title area. Each message box has an associated icon that identifies its purpose.

When you create message dialog boxes, you should keep in mind the following design considerations:

- Provide meaningful titles.
- Use complete sentences.
- Reuse the same message box to communicate the same message.

Titles for standard message dialog boxes consist of the following:

- The name of the application followed by a colon and then a space.
- The message box operation or the type of message box.

Specifying the operation in the title is recommended because the user has more information about the context of the message box. For example, you could use either of the following titles for a message box that asks if you want to delete a mail folder, but the first one conveys more information to the user:

```
Mail: Delete Folder
```

```
Mail: Question
```

You should use complete sentences or sentence-like wording in message dialog boxes. Include the same punctuation as you would in a real sentence. Keep the message concise so that it is quick and easy to read. Do not, however, omit articles such as *the* or *a*, or conjunctions such as *and*. When you create the text, remember that it might need to be translated into other languages.

In general, you should avoid using system error codes in the message. Instead, include descriptions of the actions users can perform. Place any other information in the help text for that message box.

If you need more than one sentence in your dialog box, add a line of white space before the final sentence or question. This space helps users to focus quickly on what will happen when they press one of the buttons.

Note that you should use the term Cancel, rather than Abort, in dialog boxes.

When users perform the same action that has brought up a message box, do not display a second message box. Instead, reuse the same message box, but raise it to the top of the screen. Reusing the same box avoids clutter caused by having many message boxes on the screen.

In general, you should not display a new dialog box for every message in the application. Try to reuse one or a few of the same standard message windows by replacing the text in them. If it is important for the user to view previous messages, append new messages in a scrolled window.

When creating DECwindows applications, use one or more of the five standard message dialog boxes:

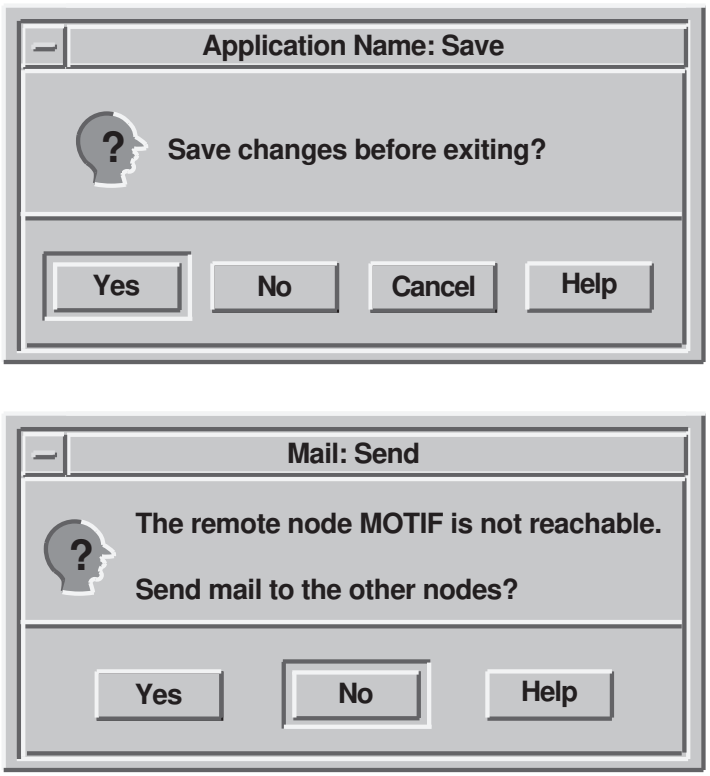
- Question (XmQuestionDialog)
- Error (XmErrorDialog)
- Information (XmInformationDialog)
- Warning (XmWarningDialog)
- Work in Progress (XmWorkingDialog)

1.4.1 Question Dialog Box

Use a question dialog box any time that you ask a question, for example, when reporting an error or giving a warning and then asking a question, as shown in Figure 1–19. Make the question message box application modal.

The question dialog box comes with a question symbol. You provide a title, a question, and properly labeled push buttons, as shown in Figure 1–19.

Figure 1–19 Examples of Question Dialog Boxes



ZK–2591A–GE

Programming Hint

To make the message box application modal, set the XmNdialogStyle resource to XmDIALOG_FULL_APPLICATION_MODAL.

Make the default the most likely choice; do not make the default push button a destructive action. Arrange the buttons in one of the following combinations:

Yes	No		
Yes	No	Help	
Yes	No	Cancel	
Yes	No	Cancel	Help

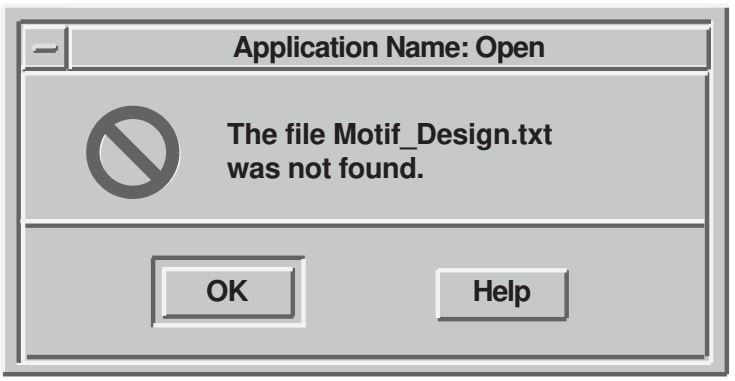
1.4.2 Error Dialog Box

Use an error message box when users do something to create the error, for example, when a user tries to write a file to a directory that does not exist. If the message box tells users they have made an error and then asks a question, use a question dialog box. If your application makes an error that is not due to user interaction, use an information dialog box. Make the error dialog box application modal.

The error dialog box comes with an error symbol. You provide a title, a statement describing the error, and properly labeled push buttons, as shown in Figure 1–20. Make the default the most likely choice; do not make the default push button a destructive action. Arrange the buttons in one of the following combinations:

OK		
OK	Help	
OK	Cancel	
OK	Cancel	Help

Figure 1–20 Error Dialog Box



ZK–2595A–GE

A Cancel button is included in the possible button combinations because Motif allows cancel buttons in error dialog boxes. Use a Cancel push button only if it does something different from the OK push button.

1.4.3 Information Dialog Box

Use an information dialog box when users need information that is not an error, or when an error occurs that is not the result of a user action. The information must be a statement. If it is a question, use a question dialog box. Make the information dialog box modeless.

The information dialog box comes with an information symbol. You provide a title, a statement, and properly labeled push buttons, as shown in Figure 1–21. Make the default the most likely choice; do not make the default push button a destructive action. Arrange the buttons in one of the following combinations:

OK

OK Help

Figure 1–21 Information Dialog Box

ZK–2596A–GE

1.4.4 Warning Dialog Box

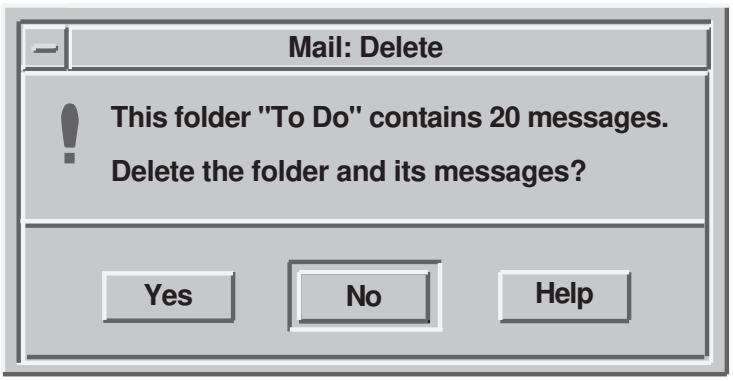
Use a warning dialog box to alert users to potential problems an action might cause. Make it application modal.

The warning dialog box comes with a warning symbol. You provide a title, a message, and properly labeled push buttons, as shown in Figure 1–22. Make the default the most likely choice; do not make the default push button a destructive action. Arrange the buttons in one of the following combinations:

Yes	No	
Yes	No	Help
OK	Cancel	
OK	Cancel	Help

Notice that Yes and No buttons are included in the possible button combinations. Motif allows questions in warning dialog boxes; in such cases you would use Yes and No push buttons. However, Digital recommends that you always use a question dialog box whenever you use a question. If you follow Digital's recommendation, you will not be using Yes or No push buttons in warning dialog boxes.

Figure 1–22 Warning Dialog Box



ZK–2670A–GE

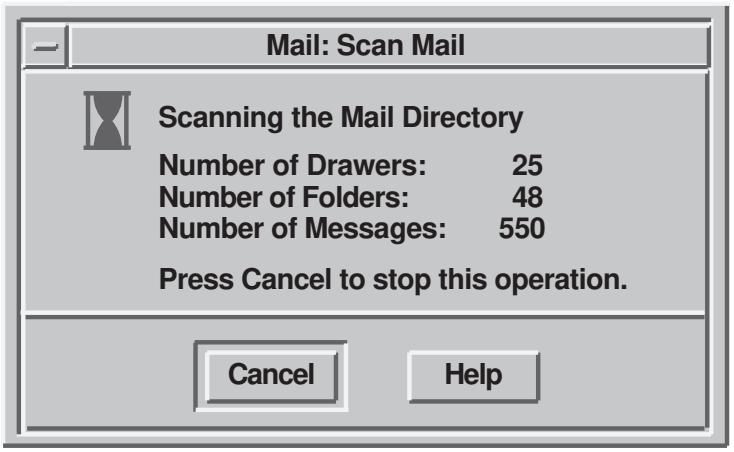
1.4.5 Working Dialog Boxes

Use a working dialog box to show work in progress and to give users an opportunity to cancel an operation. Make the working dialog box modeless. The working dialog box comes with a working symbol. You provide a title, a message, and properly labeled push buttons, as shown in Figure 1–23. Make the default the most likely choice; do not make the default push button a destructive action. Arrange the buttons in one of the following combinations:

OK Cancel

OK Cancel Help

Figure 1–23 Working Dialog Box



ZK–2597A–GE

1.5 Creating a Logical and Appealing Layout of Controls

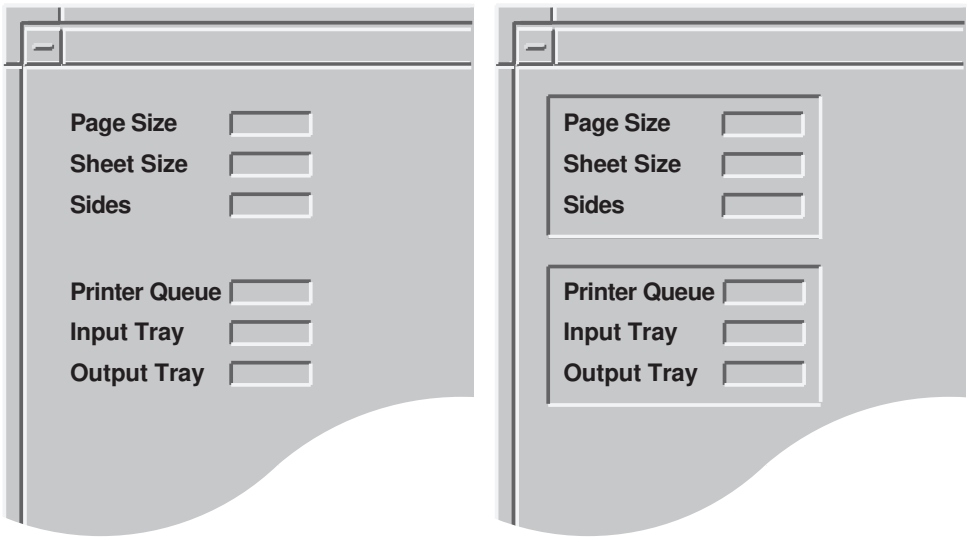
In addition to making your dialog boxes functional, it is important to organize their components in a way that is visually appealing. A good layout of a dialog box's controls will help users gain the maximum utility from the dialog boxes you create.

To help you organize controls, consider sketching each dialog box using graph paper, or an online drawing tool with a grid, or an interface design tool (IDT).

1.5.1 Group Related Choices Graphically

To give dialog boxes a perceivable structure, group related choices. Use space or bevels to create a sense of grouping. A bevel is a three-dimensional box (or frame) that you can use to surround a group of choices. For example, in Figure 1–24, the first dialog box uses space to create groupings; the second dialog box uses bevels.

Figure 1–24 Group Related Choices Using Space or Bevels



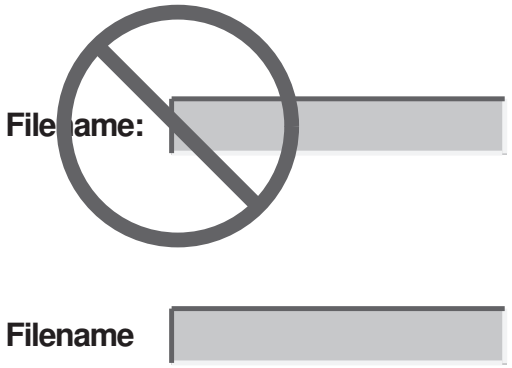
Programming Hint

To create a bevel, use an XmForm widget and set the XmNshadowType to XmSHADOW_IN. Then place the controls in the XmForm widget.

1.5.2 When to Avoid Using Punctuation in Dialog Boxes

If the elements in a dialog box are grouped and indented properly, colons or other punctuation marks between fields are not needed. For example, do not use colons between labels and input fields, as shown in Figure 1–25.

Figure 1–25 When to Avoid Punctuation in Dialog Boxes



ZK–6376A–GE

A text field with its border looks sufficiently different from a label; additional punctuation only adds clutter.

Note that punctuation is recommended when the text conveys a message. Also, colons can be used in output-only fields for header information. For example, in an editor, the current file name might appear as follows:

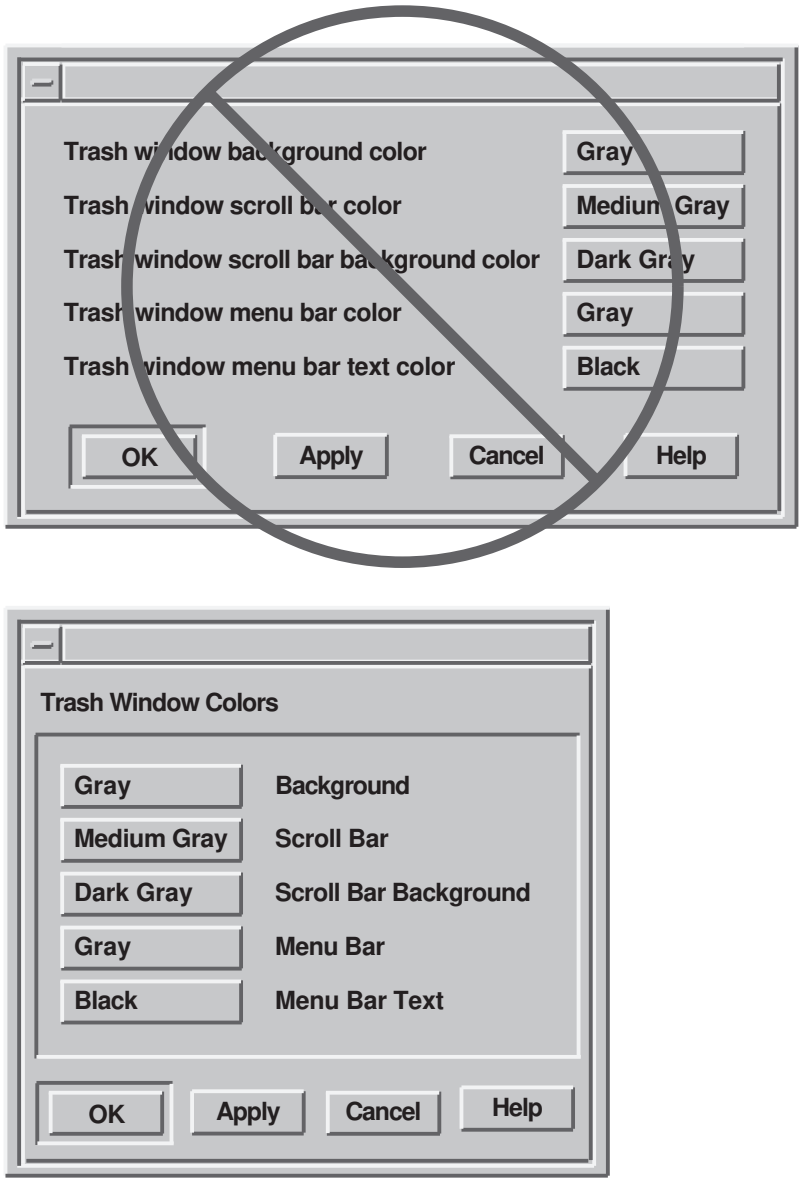
ZK–6517A–GE

1.5.3 Keep Dialog Boxes Simple

If the context or use of a dialog box is not apparent from its title, add a phrase or sentence at the top of the box to make its purpose clear. Such text can simplify the remaining contents of the box.

Figure 1–26 shows both a poor example and then an effective example of a dialog box for applying colors to the trash window.

Figure 1–26 Simplifying Dialog Boxes



ZK–6377A–GE

1.5.4 Use Uniform Spacing Within the Dialog Box

When you are placing the controls in a dialog box, follow these guidelines to make the spacing uniform:

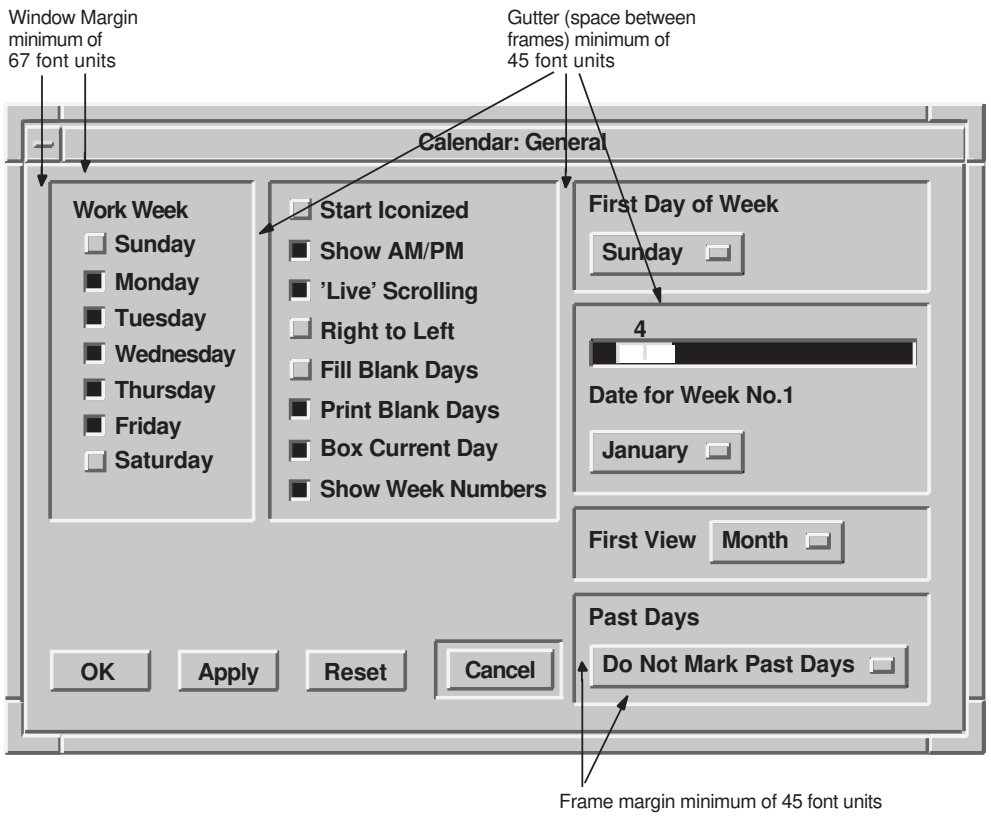
- **Make window margins uniform.**

  Make sure that each dialog box has the same top, bottom, and side margins. Do not place text or controls so that they touch any edges of the window. Use a minimum margin of 67 units of the default menu font, assuming a 100 dots-per-inch (dpi) monitor. See Figure 1–27.

  Programming Hint

  To use a 67-font-unit margin, set XmNunitType=Xm100th_font_units and specify a margin of 67.

Figure 1–27 Use Uniform Spacing



- **Make gutter sizes uniform.**

  A gutter is the inner margin between the edge of one bevel and the beginning of another. Make gutters a minimum of 45 font units of the default menu font, assuming a 100 dpi monitor. Figure 1–27 shows an example of gutter sizes.

- **Make bevel margins uniform.**

  A bevel margin is the space between the bevel and the controls or labels inside of it. Make the margin to the inside of each bevel a minimum of 45 font units of the default menu font.

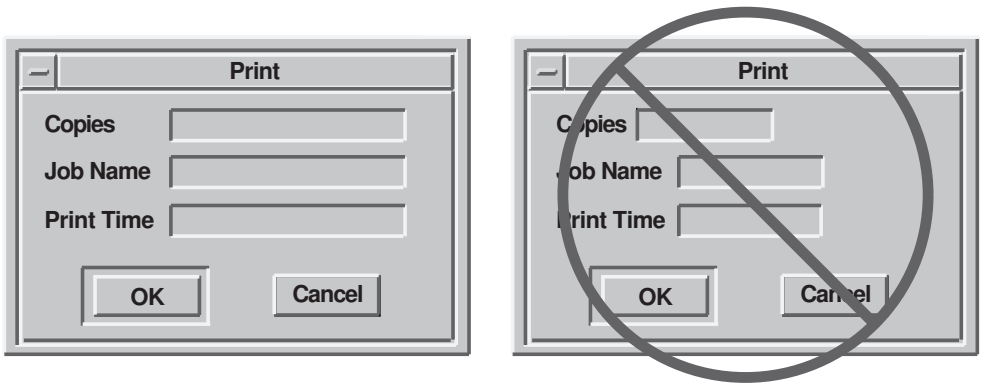
Programming Hint

Use XmNshadowThickness to make the bevel shadow a minimum of 17 font units of the default menu font.

- **Leave uniform space between labels and buttons or text-entry fields.**

  When the label comes first, followed by the button or text-entry field, adjust the amount of space between the label and the button or text-entry field so that all buttons or fields line up. Figure 1–28 contrasts an appropriate arrangement of text-entry fields with an inappropriate arrangement.

Figure 1–28 Appropriate and Inappropriate Spacing Between Labels and Fields



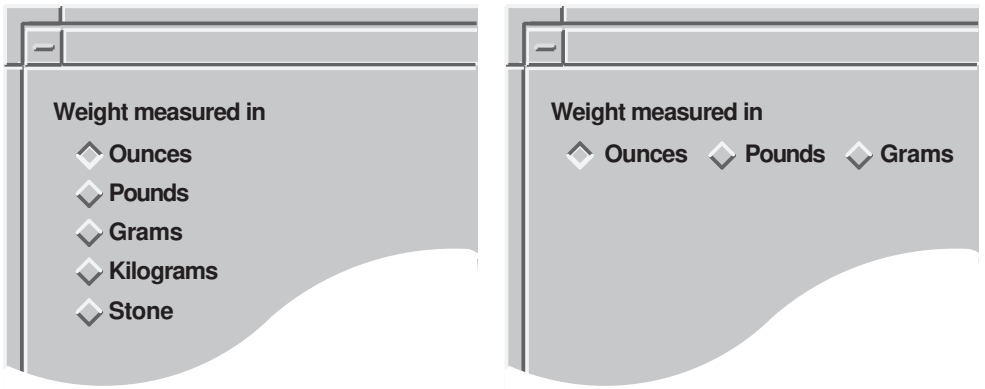
ZK–2532A–GE

Programming Hint

To make all fields line up after a label, as shown in Figure 1–28, attach your text entries to the left of the FormDialog, and specify an offset equal to the longest label, plus the space for spacing and margins.

When the button comes first followed by a label, make sure there is uniform space between the buttons and each label, as shown in Figure 1–29.

Figure 1–29 Vertical Left Justified and Horizontal Arrangements

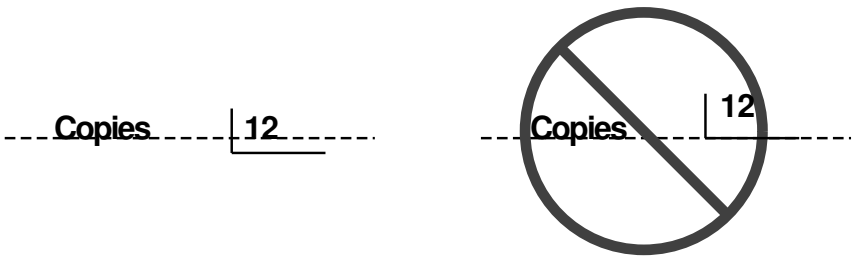


ZK–2534A–GE

1.5.5 Align Labels and Text Fields

Labels and text fields need to be aligned both vertically and horizontally. To position labels most effectively vertically, align the labels with the baseline of text in a text-entry field, not with the bottom of the entry field itself. Figure 1–30 shows an effective way to align labels, as well as one that is not effective.

Figure 1–30 Align Labels with the Baseline of Text



ZK–2514A–GE

Programming Hint

To align labels with the baseline of text, attach the top of your text widget to the top of your label, using Xm_ATTACH_OPPOSITE_ WIDGET. Then put a negative XmNtopOffset on the text widget. If, after you display the label on the screen, the baseline appears below the label, use a positive XmNtopOffset to adjust the alignment.

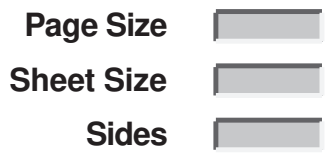
There are two recommended methods for aligning labels and text fields horizontally. One has the labels left justified, as shown in Figure 1–31.

Figure 1–31 Left Justified Alignment of Labels and Text Fields

ZK–6378A–GE

The other has the labels right justified, as shown in Figure 1–32.

Figure 1–32 Right Justified Alignment of Labels and Text Fields



ZK–6379A–GE

When labels are right justified, the right ends of the labels line up. The order of the fields is the same as the left justified method, but the implementation is different.

No matter which style you choose, make sure that you consistently use that style throughout your application and that the text fields for users to enter information are always left justified.

The following programming hints are for producing left justified and then right justified labels.

Programming Hint

For left justified labels, follow these steps:

1. Attach the left side of the label to the form using an appropriate offset.
2. Attach the left side of all the text widgets to the largest label.
3. Make sure that all your labels fit.

If you are concerned about internationalization or font customization by the user, follow these steps to modify the layout at run time:

1. Attach the left side of the labels to the form using an appropriate offset.
2. Attach the left side of each text widget to the top label.
3. At run time, traverse all the labels to find the maximum size.
4. Set the XmNwidth of the top label to be the maximum size.

Programming Hint

For right justified labels, follow these steps:

1. Start with one of these alternatives:
   - Attach the right side of the labels to a position in the dialog box, such as 30%.
   - Determine the largest label and attach it to the form using an appropriate left offset. Then attach the right side of the other labels to the largest label. Set the right offset of the other labels to zero (0) and the attachment type to OPPOSITE_WIDGET.
2. Attach the left side of the text field to the largest label.

3. Attach the right side of the text field to the form.

   Note that, if you set both the left and right attachments to the text widget, the text widget size will be overridden with some minimum size and will not be the number of columns you specified.

   To avoid this problem, perform one of these steps:

   - Explicitly set the size of the form.
   - Set the right attachment of the text field after the call to XtManageChild.

   Then, the size of the dialog box and text widgets will be determined using the column size of the text widget. Also, the text widgets will still be attached to the right side of the form and will resize with the dialog box. See Section 1.2.2 for information about how to make dialog boxes resizeable.

If you are concerned about internationalization or font customization by the user, follow these steps to modify the layout at run time:

1. Attach the left side of the labels to the form using an appropriate offset.
2. Attach the left side of each text widget to the top label.
3. At run time, traverse all the labels to find the maximum size.
4. Reset the left attachments of all the labels, except the largest one, to none. Now the largest label will determine the width of the dialog box.
5. Reset the right attachments of all the labels, except the largest one, to be attached to the largest label. Set the offset to zero (0) and the attachment type to OPPOSITE_WIDGET to align all the right sides of the labels.

1.5.6 Promote Scanning from Left to Right, Top to Bottom

Organize your controls to promote scanning from left to right, top to bottom (in left-to-right language environments). Use indentation to show groupings and relationships in a visible series. Consider the guidelines in the following sections when you organize controls.

1.5.6.1 Use the Top Left Corner

The top left corner of your dialog box is the area that users are drawn to first; use it strategically. Place one of the following controls in that corner:

- The one most used
- The most important one
- The one that would be used first in a series of controls

  For example, it might be logical to have users choose a font family before choosing whether they want a Roman or boldface typeface, so you would put the font family control in the top left corner.

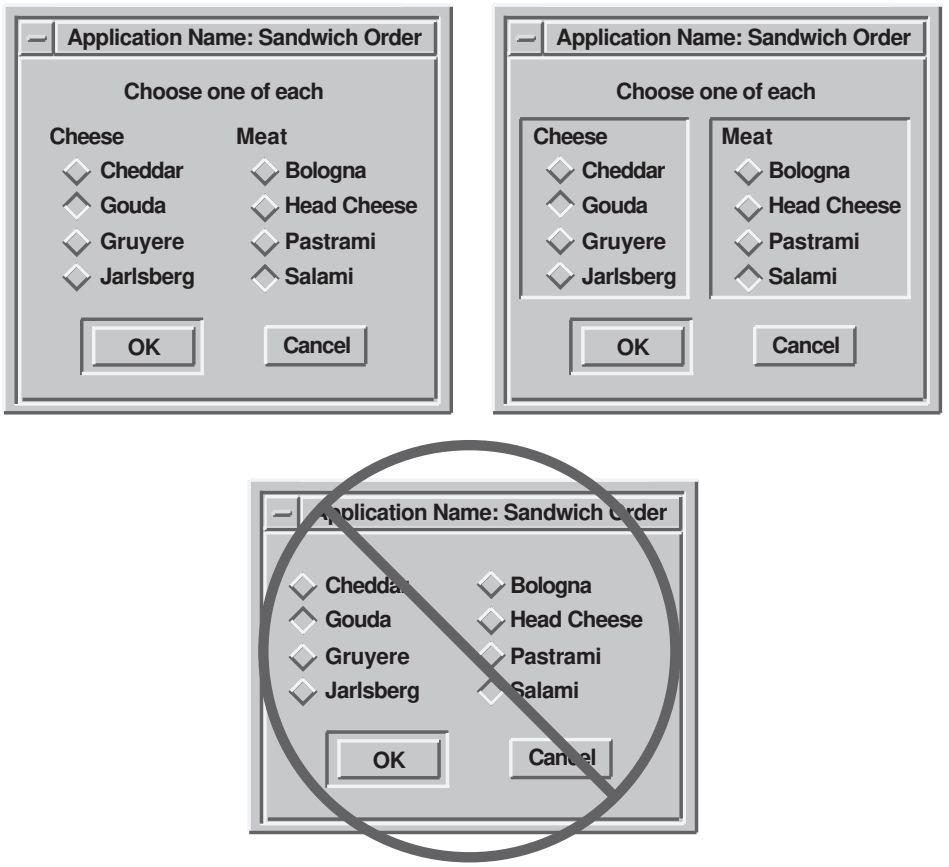
1.5.6.2 Present Related Controls in a Clearly Visible Series

When providing related controls in a series, use a left-justified vertical series or a horizontal series reading left to right, as shown in Figure 1–29. Horizontal series are acceptable for approximately four or fewer items. If the series contains more items, make it vertical. Note that a vertical series is easier to translate because foreign-language text has room to expand.

Be careful when you use two or more groups of radio buttons. Radio buttons allow mutually exclusive choices. Therefore, you want to make sure that users understand your grouping. You do not want them to be confused as to whether they can choose only one button from all of the buttons in the dialog box or whether they can choose one from each group.

You should consider indenting the radio buttons and toggle buttons so that the groupings become more obvious. Indenting adds more empty space and makes the dialog box appear less cluttered. In general, you should lay out your controls so there is enough white space between them and around the margin to make the display restful for the user's eyes, and to make it obvious where the user should focus attention. Figure 1–29 and Figure 1–33 show groupings of indented radio buttons and toggle buttons.

Figure 1–33 Appropriate and Inappropriate Labeling of Radio Buttons



ZK–2535A–GE

1.5.6.3 Clarify Structure

Clarify your structure by organizing your controls in one of the following orders:

- Alphabetical
- Frequency of choice
- Size (if choices can be measured in terms of size, such as millimeter, centimeter, meter, kilometer)

Notice that the cheeses and meats in Figure 1–33 are listed in alphabetical order. Notice also that the weight measurements in Figure 1–29 are organized for frequency of use in American culture, with the two metric units grouped together. In this particular case, you could alphabetize the choices to get the following order:

- Grams
- Kilograms
- Ounces
- Pounds
- Stone

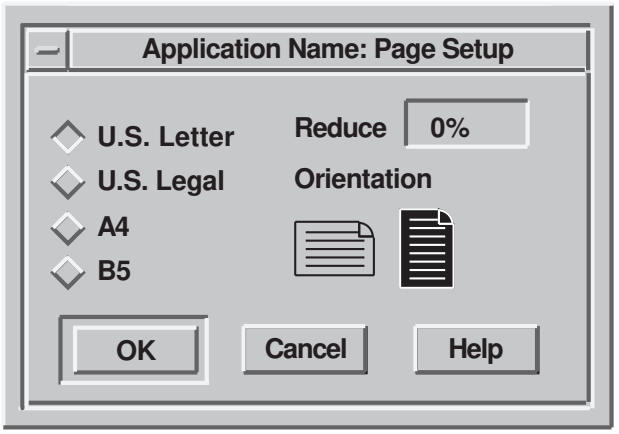
The organization still preserves the grouping of metric, English, and Imperial measurements.

1.5.6.4 Provide Appropriate Defaults

Appropriate defaults allow users to continue working by pressing the Return key (thereby activating the default push button). For example, you can have a dialog box show the most commonly chosen controls already selected so that, in most cases, users can just press Return to get those choices.

Figure 1–34 shows a Page Setup dialog box that appears with the most common choices (the defaults) already selected. Users can either click on OK or press Return to accept the defaults.

Figure 1–34 Provide Appropriate Defaults



ZK–2536A–GE

1.5.6.5 Use Option Buttons to Conserve Space

Option buttons post an option menu. (Do not confuse an option button with an Options... push button, which invokes another dialog box. Also, do not confuse it with the Options menu in the menu bar.) An option button allows you to conserve space in dialog boxes because the menu items are only displayed when users press the option button, as shown in Figure 1–35.

Using option buttons promotes progressive disclosure, avoids overloading the user with too much information at once, and contributes to a cleaner, less cluttered, interface.

You should consider using option buttons instead of radio boxes in the following circumstances:

- You have more than one radio box.
- There are many other elements in the dialog box.
- There would be many radio buttons in the radio box.

Figure 1–35 Use Option Buttons to Conserve Space

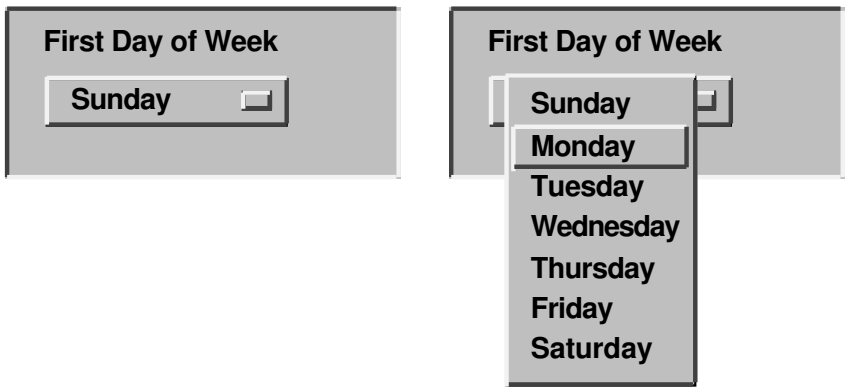


Figure 1–36 shows how you would use two option buttons to present the same choices that the user is given in Figure 1–33.

Figure 1–36 Using Two Option Buttons

If you create a set of option buttons together, either in a row or in a column, align them horizontally or vertically as appropriate to provide a pleasing layout. If the buttons are approximately the same size, you should make the option buttons the same width.

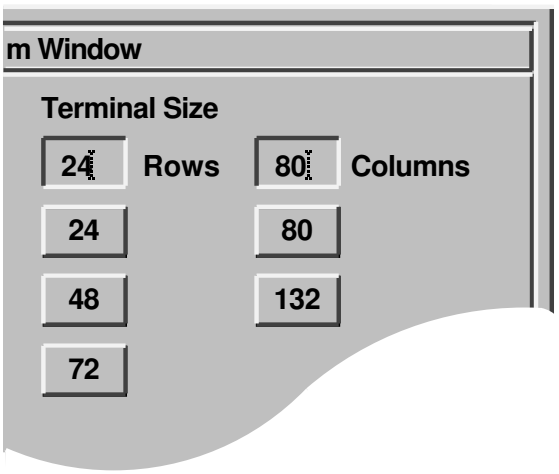
1.5.6.6 Initializing Values in Dialog Boxes

When a dialog box is displayed the second time, you should try to preserve the values for all the fields. For example, when users display a Create dialog box again, they might want to retain most of the values that were already entered. Preserving these values saves users from having to reenter all the information. In the case of a Directory dialog box, users might want to display the same elements the second time, but change the contents of one of the fields, such as the time last modified.

1.5.6.7 Provide a Variety of Selection Controls

Provide selection controls (such as radio buttons or list boxes) and text-entry fields to gather the same information, when suitable. Doing so accommodates different user preferences for entering information. For example, Figure 1–37 shows a dialog box that allows users to customize the size of their windows by either typing in the size in a text field or by selecting the appropriate push button.

Figure 1–37 Provide a Variety of Selection Controls



ZK–2531A–GE

Be careful, however, to avoid clutter by providing too much variety. Also, do not overwhelm users with too many visual features, such as flashing, bolding, and underlining. In general, avoid using flashing because it distracts the user.

1.5.7 Creating Tab Groups

A **tab group** consists of any managed or primitive widget that is organized into a traversable group or field. In other words, a tab group is a group of controls that belong together. For example, a set of radio buttons is a tab group. To move from tab group to tab group in a dialog box, users press the Tab key (hence the name tab group).

Organize the controls in your dialog boxes into distinct tab groups. The toolkit makes two distinctions with regard to navigating dialog boxes with the keyboard:

- Navigating between tab groups
- Navigating within tab groups

1.5.7.1 Navigating Between Tab Groups

Treat each of the following types of controls as its own tab group:

- List boxes
- Sashes
- Scales
- A single-line text-entry field
- A set of radio buttons
- A set of check buttons
- One option button preceded and followed by a different type of tab group

  A group of option buttons together makes one tab group.

Follow these general rules for navigating between fields:

- Allow users to navigate the dialog box in the same direction as the language of the dialog box. That is, place the location cursor in the top left of the dialog box, and when users press the Tab key, move the location cursor from left to right, top to bottom from tab group to tab group in a predictable manner.

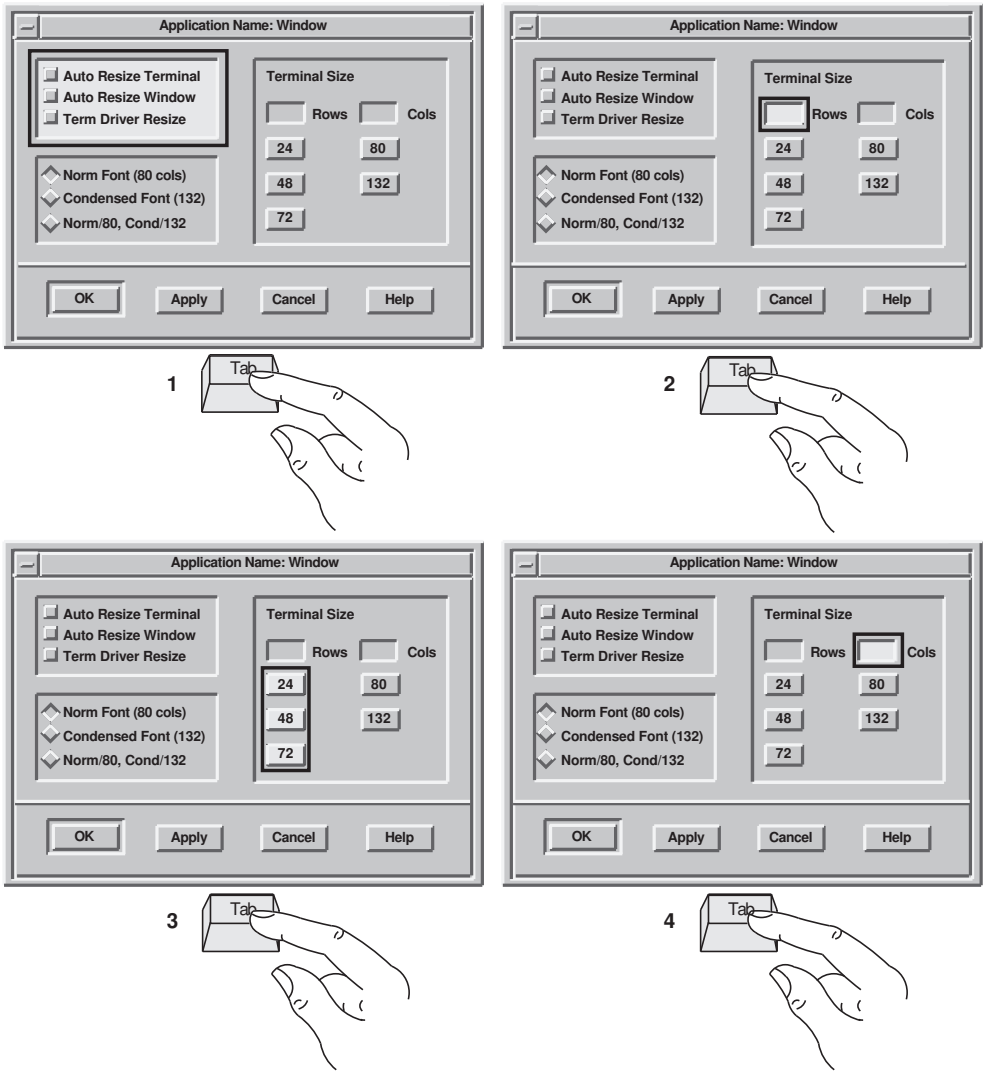
  Programming Hint

  To retain the proper order, create children in the same order in which users tab through the dialog box.

- If users leave the dialog box to give input focus to another window, do the following when users give input focus back to the dialog box:
  - If users click over a specific control in the dialog box, give that control the input focus.
  - If users click in a general area (such as the title bar or a blank space), or if users use the keyboard to assign input focus, place the location cursor on the component that last had the location cursor on it.

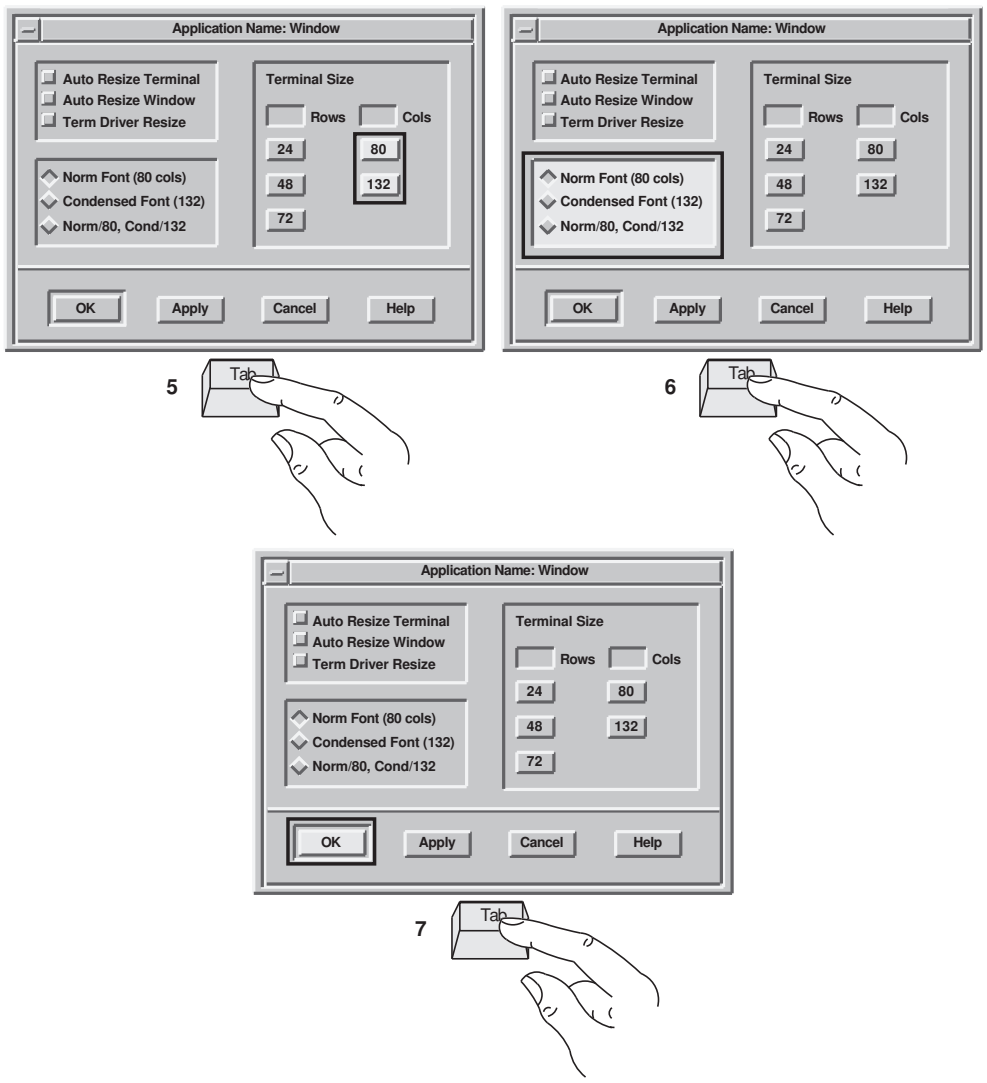
Refer to Figure 1–38 and Figure 1–39 to see how the input focus moves from tab group to tab group when users press the Tab key. The location cursor wraps from the bottom right tab group to the top left tab group when users continue to push the Tab key.

Figure 1–38 Use the Tab Key to Move Between Tab Groups



ZK–2772A–GE

Figure 1–39 Use the Tab Key to Move Between Tab Groups–continued



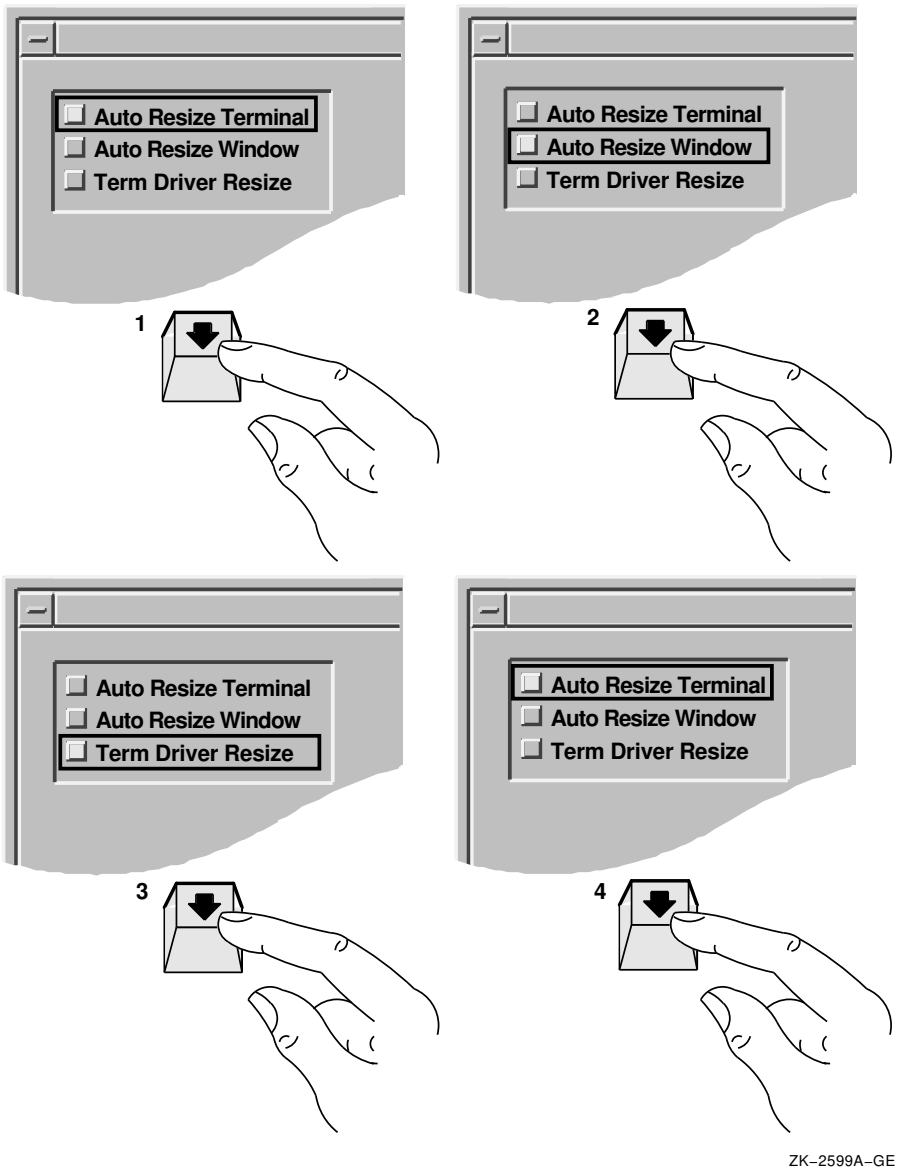
ZK–3335A–GE

1.5.7.2 Navigating Within Tab Groups

To move from component to component within a tab group, users press the arrow keys. For example, to move from one item to the next in a series of radio buttons, users press the down arrow key. However, to move from a radio button to the next tab group, users press the Tab key.

Refer to Figure 1–40 to see how the input focus moves from component to component within a tab group when users press the arrow keys. Notice that the location cursor **wraps**; that is, when users press the down arrow key to go to the last item in a series and then press the down arrow key again, the location cursor moves up to the first item in the series.

Figure 1–40 Use the Arrow Keys to Move Within Tab Groups



1.6 Presenting Text in Dialog Boxes and Other Screen Objects

When creating labels and text for dialog boxes and other screen objects such as menus and title bars, follow these guidelines:

- Avoid using all uppercase letters for text. Mixed case improves readability and style.

```
Use:    Null Pointer

Avoid:  NULL POINTER
```

- Follow the capitalization rules of the local or translated language when you use local or translated words.
- For dialog boxes, wherever possible, use sentences or sentence-like wording. Capitalize the first word of each sentence. Use sentence punctuation unless such marks could confuse literal displays, such as file names.

```
Loading file /usr/sbin/uerf.hlp

The file /usr/sbin/uerf.hlp cannot be found.
Please check the file name and directory for spelling
or other errors and try again.
```

- Capitalize each word of a screen object within a dialog box (such as labels for toggle buttons, radio buttons, or text-entry fields) except for articles and prepositions that have less than five characters and fall between two other words.

```
Levels with Periods
Number of Columns
Space Between Columns
First Line Indent
Dates Before 1993
```

- Capitalize the first letter of each word in a push button. An exception is the OK push button where both letters are uppercase.

```
Go Back
Exit
```

- Capitalize the first letter of each word in a menu name, menu item, or label, except for articles and propositions that have fewer than five characters and fall between two other words.

```
Show Room
Go to Page...
Save As...
```

- If a menu name, menu item or label uses a hyphenated word, capitalize the first letter of the first word and the first letters of all other words that are nouns or proper adjectives or that have the same importance as the first word. Use all lowercase letters for any articles, conjunctions, or prepositions that fall between two major words.

  ```
  Side-by-Side
  ```

  ```
  Cross-References
  ```

1.7 Creating Title Bars

Title bars are used in both primary and secondary windows.

For primary windows, provide labeling as follows:

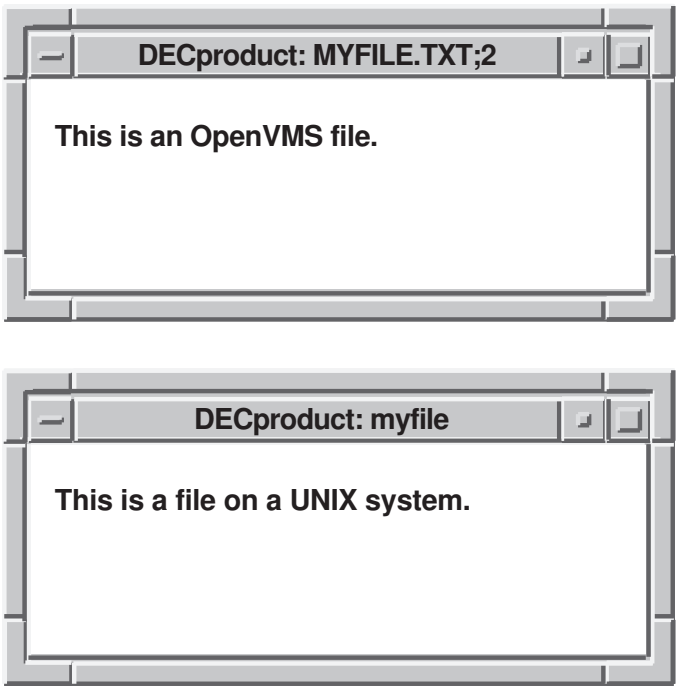
- Use the application name followed by a colon and the name of the file. Center the information.

  If the application uses a default file name or requires the user to name a file before creating it, display that name on the title bar at startup. If the application allows users to create a file before it is named, do not put a file name on the title bar initially.

- Do not put the version number of the application in the title bar.
- On UNIX and Windows NT systems, do not put the pathname of the file. On OpenVMS systems, do not include the directory where the file is located.
- On UNIX based systems, display the letters in the file name in the cases that the user typed. On OpenVMS systems, display the file name in all uppercase letters or, for ease of reading, all lowercase letters and include the file extension as well as the version number. On Windows NT systems, display the file name in either all uppercase letters or, for consistency, all lowercase letters and include the file extension.

  Figure 1–41 shows examples of file names in title bars.

Figure 1–41 Putting Information in Title Bars



ZK–2760A–GE

For secondary windows, provide labeling as follows:

- Center the title of the window in the title area. (Centering is the default.)
- If the secondary window is a dialog box, give the dialog box a title that consists of the application name followed by a colon and the command that created it.

For more information on naming dialog boxes, see Section 1.2.1.

1.8 Using the Working Dialog Box and the Wait Cursor

Use Working dialog boxes and wait cursors to give the user immediate feedback that your application is in process.

In most circumstances, if the action will take more than 15 seconds, use a Working dialog box to show work in progress. Since the length of time that an action takes is system dependent to a large extent, you might want to make the display of a Working dialog box be a customizable option.

When actions will take less than 15 seconds, let the user know the application is processing by changing the cursor to a wait cursor such as the DECwindows watch cursor. Continue to display the wait cursor until the action is complete. Use a wait cursor for the following types of actions:

- Selecting a menu item that will bring up a dialog box
- Performing a direct action
- Scrolling a window
- Expanding or collapsing items
- Selecting OK or another button in a dialog box

Set the wait cursor on all windows where the user is waiting for an application to complete an action. For example, if the application is refreshing a directory, the wait cursor should be set in the directory window as well as all other windows that are visible and pending the completion of that refresh.

Programming Hint

Use the following code example to set a wait cursor for the active window. The #ifdef is used when the application runs on an OpenVMS system. The other include statement applies to both UNIX and Windows NT systems.

```
#ifdef OpenVMS
#include <decw$cursor.h>
#else
#include <X11/decwcursor.h>
#endif
#include <DXm/DECspecific.h>

  /* Wait? */
  if ( wait )
    /* Set the wait cursor for this window. */
    waitCursor = DXmCreateCursor(widget,decw$c_wait_cursor);
  else XUndefineCursor(XtDisplay(widget), XtWindow(widget));
```

2
Designing Menus and Menu Items

This chapter provides guidelines for designing menus. It supplements the *OSF/Motif Style Guide* by discussing the following topics:

- Naming menus and menu items
- Organizing menus and menu items
- Using ellipses in menu items
- Disabling menu items
- Designing pop-up menus
- Designing tear-off menus

This chapter also discusses the OSF/Motif File, Selected, Edit, View, Options, and Help menus, and recommends items to add to these menus. In addition, you can include a Windows menu and your own application-specific menus.

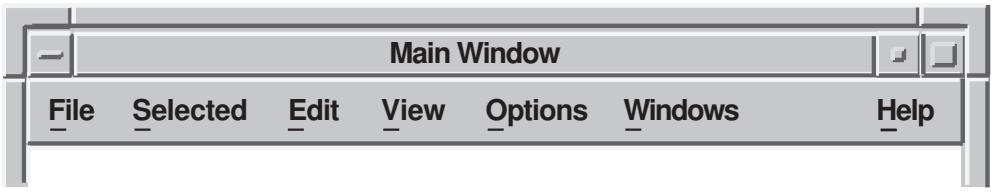
2.1 Naming Menus and Menu Items

Use the standard terms for common menus, such as File, Selected, Edit, and Options. In left to right environments, place the menus in the recommended order, with File farthest to the left, followed by Selected, and then Edit. Application-specific menus should follow the Edit menu. Place the View, Options, and Help menus to the right of any application-specific menus. The Options and Help menus should always be farthest to the right. If you include a Windows menu, place it between the Options and Help menus. See Figure 2–1 for an example.

Programming Hint

To position the Help menu at the far right of the menu bar, set XmNmenuHelpWidget to the name of the Help cascade button.

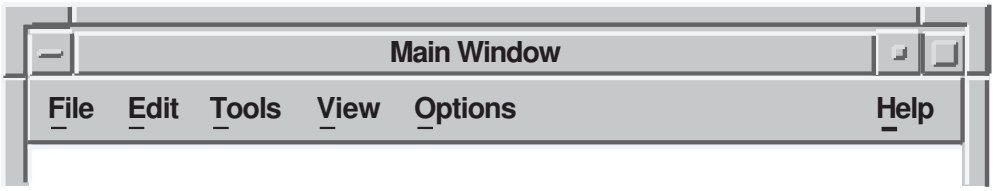
Figure 2–1 Order of Menus in the Menu Bar



ZK–2771A–GE

Additional menu bar items should be placed after Edit and before Help. When there is a View menu bar item, most additional items should be placed between Edit and View. In this way, users become accustomed to the File, Selected, and Edit menus appearing at the left and the View, Options, and Help menus appearing at the right. Figure 2–2 shows two application-specific menu bar items.

Figure 2–2 Sample Menu Bar with Application-Specific Items



ZK–6407A–GE

When you are creating your own menus and menu items, avoid giving a menu the same name as a menu item. For example, Paste is an item on the Edit menu, so do not create a Paste menu in the menu bar.

Use verbs or adjectives for your menus and menu items instead of nouns. Verbs and adjectives give users a better idea of what action a command will perform.

Follow the guidelines for text presentation (such as capitalization and punctuation) in Section 1.6.

Use terminology that will be familiar to users. For example, if you are designing a simple drawing tool, avoid terms like chamfer or bevel, but if you are designing a drawing tool for graphic artists, use these terms. Whatever terms you choose, make sure they are distinct from each other. Ask users to

review your choices of menus and menu items and to tell you if the terms are clear and distinct.

Give each menu and each menu item a mnemonic. Consider giving the most commonly used menu items accelerators.

Programming Hint

To create a mnemonic and an accelerator, use the following code example:

```
object copy_button : XmPushButton
    {
        arguments
            {
                XmNlabelString      = "Copy";
                XmNaccelerator      = "Ctrl<Key>Insert";
                XmNacceleratorText  = "Ctrl+Ins";
                XmNmnemonic         = keysym("C");
            };
    };
```

For information on capitalization of menu items, see Section 1.6.

2.2 Organizing Menus and Menu Items

The way you organize your menus and menu items depends on your application. Choose the most appropriate guidelines from the following list to help you:

- Create logical groupings of menus and menu items.

  Menu items can be composed of either text or graphics, and can span multiple lines. Within a menu, use horizontal separators (lines) to form logical groups and separate menu items. For example, on a menu for editing a document, place all graphical menu items together, and all textual menu items together, and separate the two groups with a line.

  Place together sets of related items and separate them from other menu items by a horizontal separator. Let users help you decide what constitutes a logical group.

- Place menu items within the menu in the order of most common use.

  In general, place the most commonly used commands first in the menu and the least frequently used commands last. Involve users in your design and testing so that you will know which menu items are most commonly used.

You can also arrange menus items in groups, and within those groups arrange them according to frequency of use. For example, you might place Exit and Close at the bottom of a menu (even though they are used often) to be consistent with the placement of Close in the window menu.

- Organize menu items in a perceivable order.

  For example, arrange a list in alphabetical order, or arrange the menu items from the smallest unit of measurement to the largest (for example, type size from 8 points to 24 points).

- Put related opposite menu items in the same menu.

  For example, place Show and Hide, and Undo and Redo in the same menu, or create a menu item that changes depending on the context. For example, have the Undo menu item change to Redo if users have just performed an Undo operation.

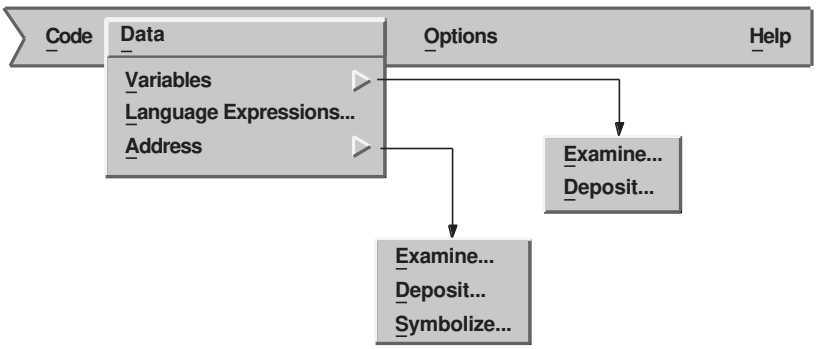
- Do not put a destructive command (such as Delete or Revert) next to other frequently chosen items.

  One of the most common user problems with menus is the off-by-one error, in which users mistakenly choose the item next to the one intended. Be aware of the result if users choose the menu item above or below the intended item. If, however, a destructive function logically goes with a frequently chosen item, make sure that the users can undo the destructive function, or make sure you provide a question dialog box that confirms that users really want to perform the destructive option.

- Use no more than three levels of menus, as a rule.

Consider providing a menu map, either on line or in hardcopy form. Some products provide one map of all menus and menu items, while others provide separate diagrams for each menu and its associated submenus. Figure 2–3 shows a map for one menu and its associated submenus.

Figure 2–3 A Menu Map



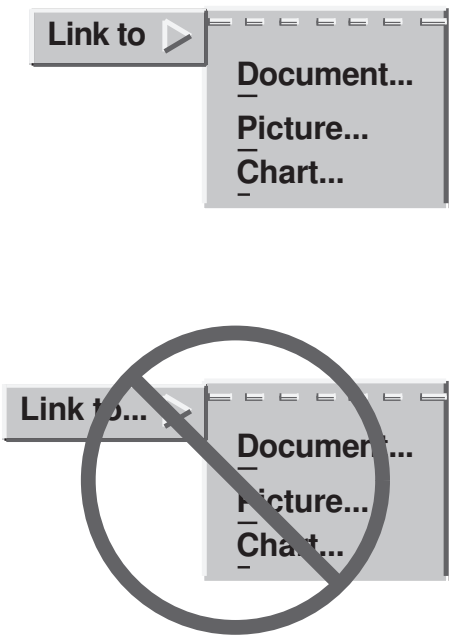
2.3 Using Ellipses in Menu Items

Use a horizontal ellipsis (...) to indicate that pushing the button invokes a dialog box or window that requires some action on the part of the user. For example, use an ellipsis with the Print menu item if it invokes a dialog box in which users must provide printer information before a file can be printed.

Do not use ellipses with the Help push button or Help menu item since Help does not require any action. (For information on designing online help, see Chapter 5.)

Do not take short cuts when providing ellipses in cascade buttons. Always use ellipses in cascade menu items. Figure 2–4 shows the appropriate way to use ellipses with cascade buttons.

Figure 2–4 Ellipses with Cascade Buttons

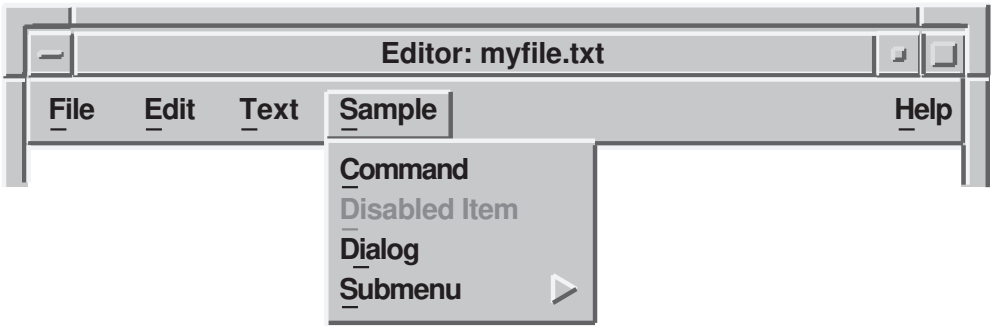


ZK–6406A–GE

2.4 Disabling Menu Items

At a given time, your application can make a menu item unavailable to users; that is, it can disable the menu item. Users cannot choose a disabled menu item. Figure 2–5 illustrates a disabled menu item.

Figure 2–5 A Menu with a Disabled Menu Item



ZK–2676A–GE

Programming Hint

To disable a menu item, call XtSetSensitive with the widget ID of the widget.

You can allow users to display a menu or submenu in which all items are disabled. With such a display, users can see what features your application provides. (This guideline does not necessarily apply to pop-up menus; for guidelines on pop-up menus, see Section 2.5.2.)

If your application does not support a standard menu item, do not display that menu item at all. For example, if your application does not support the Select All menu item, do not include Select All in a menu.

Use the following guidelines to determine whether to disable a menu item:

- Disable menu items for operations that users cannot do; that is, menu items that would cause an error or result in no action when users chose them.

  For example, disable the Cut menu item from the Edit menu if users have not selected anything to cut; disable an Undo menu item if users cannot undo the last operation.

- Do not disable menu items for operations that users can do, even if doing them does not change anything.

  For example, do not disable a Select All menu item, even if users have already selected all items. It is best not to disable Select All because doing so could cause confusion to users.

Some menu items invoke dialog boxes; your decision to disable such menu items depends on whether the dialog box that is invoked is modal or modeless:

- If the menu item invokes a modal dialog box that requires some context, such as the selection of a mail folder, you can do either of the following:
  - Disable that menu item if choosing it would cause an error or would display a dialog box with no meaningful context.
  - Enable that menu item if users can provide meaningful context within the dialog box, for example, by entering the mail folder name. In such cases, you should disable the OK and Apply push buttons until the dialog box has some context.
- If the menu item invokes a modeless dialog box, keep the menu item active, and display the associated dialog box when users click on the menu item.

  However, disable the OK and Apply push buttons on the dialog box. Because the dialog box is modeless, users have the ability to make changes, and might make a change that renders the dialog box useful. If users make such a change, enable the push buttons on the dialog box.

  For example, if a graphics editor has a menu item called Align which invokes a modeless dialog box, users should be able to click on Align and see its associated dialog box, even if they have not created or selected any objects to align. If users have not created or selected any objects to align, disable the push buttons in the dialog box. However, once they create or select a few objects, change the push buttons from disabled to enabled.

2.5 Designing Pop-up Menus

Pop-up menus provide users with shortcuts for accessing pull-down menu items.

Because pop-up menus appear at the current pointer position when MB3 is pressed, users do not need to move the mouse very far. Pop-up menus allow users to keep their attention focused on their work while choosing an action. Invoking functions exclusively from pull-down menus and dialog boxes can require extensive mouse movements, resulting in lower productivity.

2.5.1 Characteristics to Give to Pop-up Menus

Design the pop-up menus to have the following characteristics:

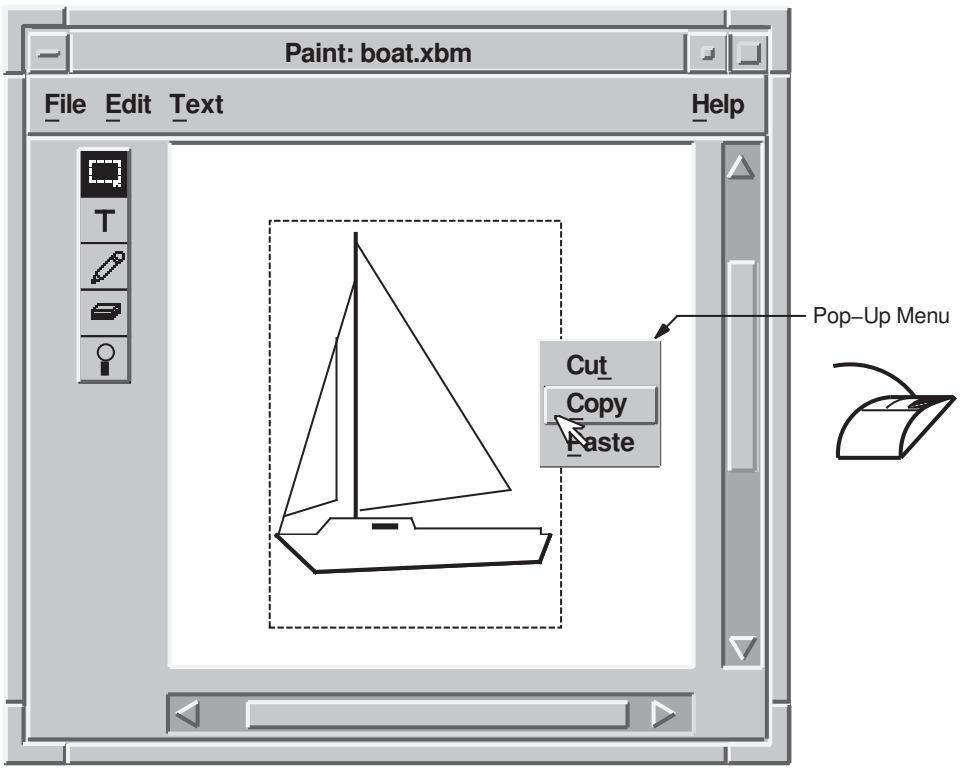
- Make each item in a pop-up menu available through another mechanism, such as a pull-down menu or a dialog box.

  Note that a pop-up menu serves only as a shortcut to menu items; it must not be the exclusive mechanism for reaching those items.

- Give pop-up menus a title. (This characteristic is optional.)

  For example, the pop-up menu for the default Motif workspace menu has the title Workspace. Figure 2–6 shows a pop-up menu without a title.

Figure 2–6 Pop-up Menu



Organize pop-up menus in one of the following manners:

- If the contents of a pop-up menu are similar to those of pull-down menus, preserve the general order that you would use in a pull-down menu. The pop-up menu order will then be consistent with the pull-down menus. For example, if the pop-up menu has some items from the File and Edit menus, place the items from the File menu first and the items from the Edit menu second, with a separator between the two groups.
- If your application's pop-up menus are very different from the pull-down menus, follow these guidelines:
  - Place menu items in descending order from the more frequently invoked to the less frequently invoked.

    To determine which items are most frequently used, involve users and analyze your application by recording the keystrokes users make and the commands they use. Consider using separators in pop-up menus.
  - Order menu items alphabetically.
  - Order menu items logically.

    The pop-up menu in Figure 2–6 is ordered logically.

Include the most frequently used functions of your applications in pop-up menus. Pop-up menus may contain a heterogeneous group of functions. Because pop-up menus act as accelerators, they require less rigid categories of items than do pull-down menus.

There are two ways to implement pop-up menus:

- Create one generic pop-up menu to display all the items you feel are appropriate.

  Then, depending on the context when users invoke the menu, disable the inappropriate items.
- Create different menus for different contexts.

  For example, create one pop-up menu for contexts in which users of a graphics editor have selected a graphical object, and a different pop-up menu when they have selected text. Even if you create separate pop-up menus, there still might be situations in which you disable certain menu items.

2.5.2 Determining the Actions of Pop-up Menus

Have items in a pop-up menu act on the current selection (if there is one), according to the following guidelines:

- Do not change the current selection when users display a pop-up menu.
- Keep the selected object selected after users use the menu so that they can perform further actions if they want.

  Of course, if users delete selected items, these items will not be present to be selected or deselected.
- Do not allow users to select multiple objects from the pop-up menu if the actions on these selections are inconsistent.

  For example, if you allow users to choose two objects to open, make sure your application opens them in two different windows. If users want to delete two objects, make sure that your application deletes both.

Make your application do one of the following two things if users have not selected any items:

- While the menu is displayed, highlight the item or objects that will be affected by the menu functions.
- Display a generic pop-up menu with menu items that pertain to the ''no-item-selected'' context. For example, in a graphics editor, a pop-up menu could have menu items for creating objects.

2.5.3 Choosing Between a Pop-up Menu or a Control Panel

Pop-up menus provide users with quick and direct access to application functions. Control panels and modeless dialog boxes that contain push buttons perform a similar role. Use pop-up menus when users are focusing on the work area, and when moving the mouse between the control panel and work area would be distracting. Use a control panel or a modeless dialog box when users want the controls to be constantly visible. You can also use tear-off menus to give the user the choice of keeping the controls visible. (See Section 2.6 for information about tear-off menus.)

2.5.4 Designing Pop-up Menus with Submenus

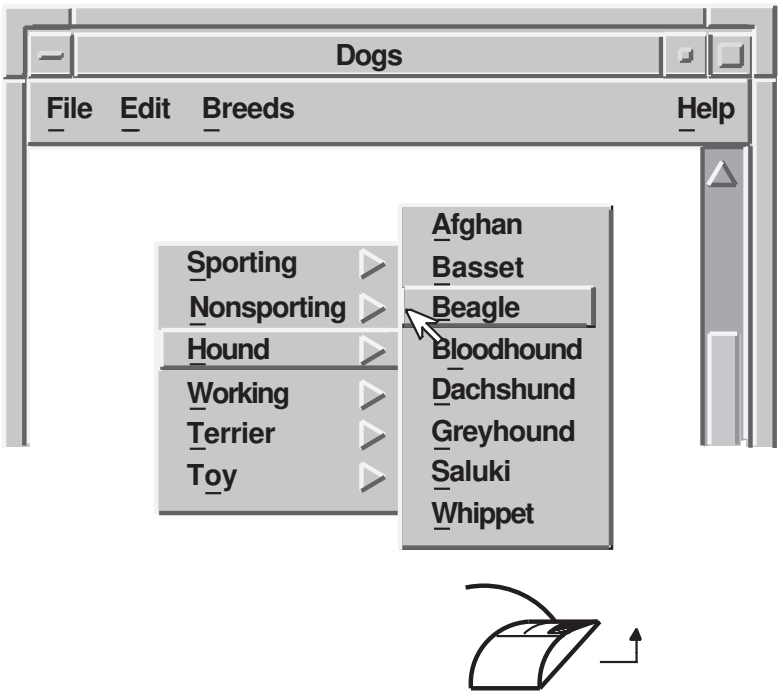
When combining pop-up menus with submenus, follow these guidelines:

- Place menu items within the submenus in logical groups.
- Do not use an extensive hierarchy of submenus because the amount of mouse movement involved defeats the purpose of the pop-up menu.

- Keep only three or fewer levels in a pop-up menu hierarchy, as a general rule.

Figure 2–7 illustrates a pop-up menu with a submenu.

Figure 2–7 Pop-up Menu with a Submenu



2.6 Designing Tear-off Menus

A tear-off menu is one that can be torn off into a secondary window. After the menu is torn off, the user can position it and use it in the same way as a dialog box.

When menus are torn off, they appear in a separate window that can be directly accessed. Tear-off menus should be used when elements in a menu can be activated many times in succession. Pull-down, option, cascade, and pop-up menus can all become tear-off menus.

For example, the View pull-down menu, which can contain different views as well as Expand and Collapse items, is a good candidate for a tear-off menu. Most Options pull-down menus, which are used infrequently, might not be appropriate for tearing off. However, Options menus for setting the font family and size for desktop publishing applications are good candidates for becoming tear-off menus.

You should avoid creating tear-off menus from pop-up menus that are context sensitive, with items changing depending on the location of the pointer.

Programming Hint

To set the tear-off attribute of a menu, set the XmNtearoffModel resource for the pull-down menu:

```
XmPulldownMenu
    {
        arguments
            {
                XmNtearoffModel = XmTEAR_OFF_ENABLED;
            };
    };
```

2.7 Items in the File Menu

The File menu provides commands that allow users to manage files. Place the File menu to the far left in the menu bar (in left to right environments), and give it a mnemonic of F . The mnemonic allows a user to press a single letter on the keyboard in order to select a menu item. Indicate the mnemonic with an underscore.

Motif recommends the following menu items for the File menu:

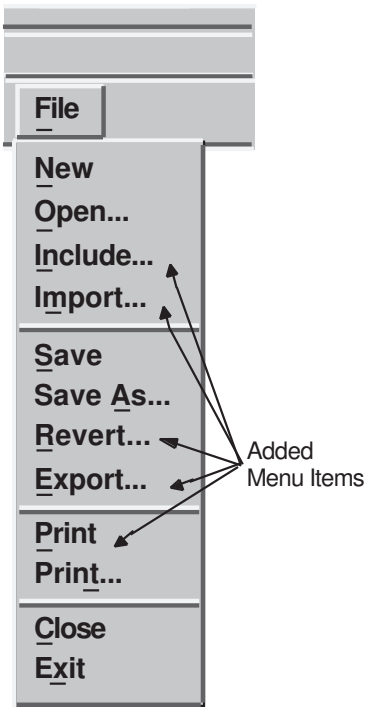
- N ew
- O pen...
- S ave
- Save A s...
- P rint
- C lose
- Ex it

You can improve the usefulness of the File menu by adding the following items:

- I nclude...
- Im port...
- R evert...
- E xport...
- Print ...

Figure 2–8 shows a File menu with Digital's additional menu items in the recommended order.

Figure 2–8 A File Menu and Its Menu Items



ZK–2551A–GE

Group the menu items according to the following:

- Initiating actions
- Saving actions
- Printing actions
- Exiting actions

The functions of the File menu items are as follows:

New

The New menu item must open a new file in the current window. It must clear existing data from the client area and update the current file name. If users specify a new file name, you can display the file name in the title area or in a status area with a full pathname. If they do not specify a file name, display the term ‘‘Untitled’’ in the title area. Digital suggests that if you give New an accelerator, make the accelerator Ctrl+N.

To accommodate users who already have a file open when they choose the New menu item, have your application display a question message box that asks if users want to save the file. Figure 2–9 shows a question message box that does this.

Open...

The Open menu item must open an existing file. When users click on this item, it must bring up a dialog box that prompts users for the name of the file. Digital suggests that if you give Open an accelerator, make the accelerator Ctrl+O.

To accommodate users who already have a file open when they choose another file to be opened, you can do either of the following:

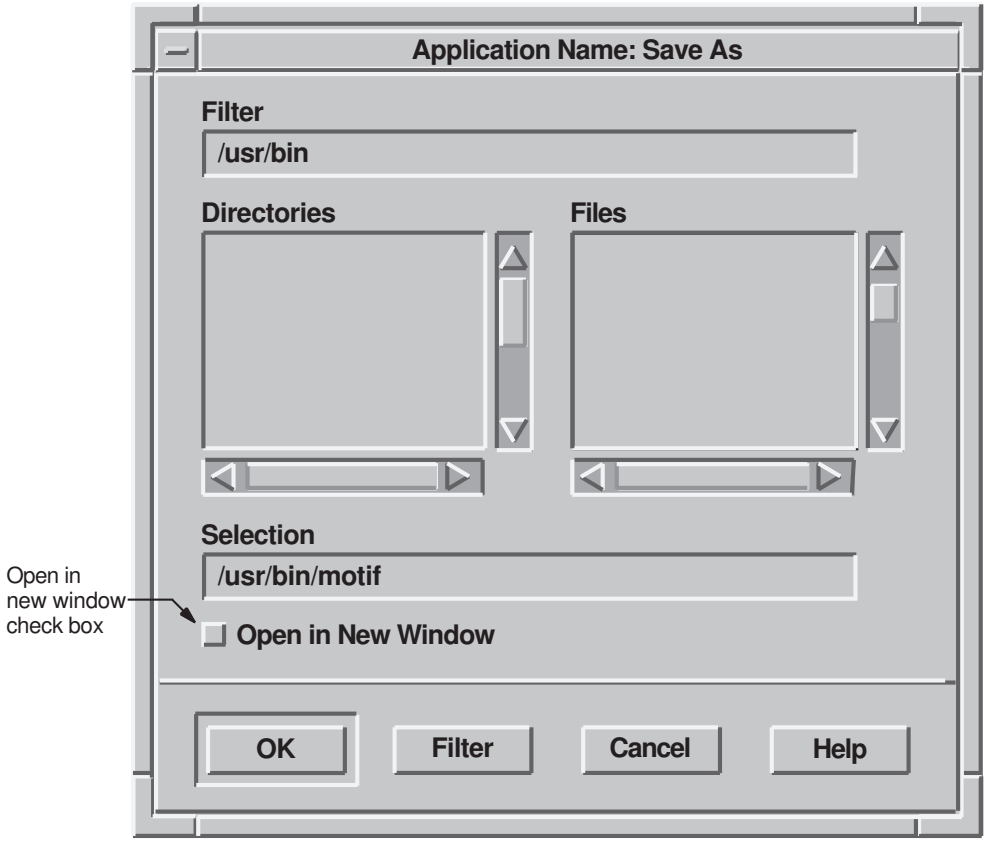
- Have your application display a question box that asks if users want to save the file before closing it and opening another. Figure 2–9 shows a ‘‘save file’’ question message box.

Figure 2–9 A “Save File” Question Message Box

ZK–2677A–GE

- Have your application display an Open File dialog box with a check button that asks if users want to open the file in a new window. Figure 2–10 shows an example of such a dialog box.

Figure 2–10 An "Open in New Window" Check Box in an Open File Dialog Box



Include...

Make the Include menu item add the contents of a specific file to the currently opened file. When users choose Include, generate a File Selection dialog box and add the included material at the current insertion point.

Import...

The Import menu item is similar to Open, except that it converts the format of the specified file to the native format of the application. Users need to supply both the name of the file to import and the current format of that file.

Save

The Save menu item must save the currently opened file without removing it from the client area. If users have not already named the currently opened file, Save must generate a Save As... dialog box that prompts users for a file name. Digital suggests that if you give Save an accelerator, make the accelerator Ctrl+S.

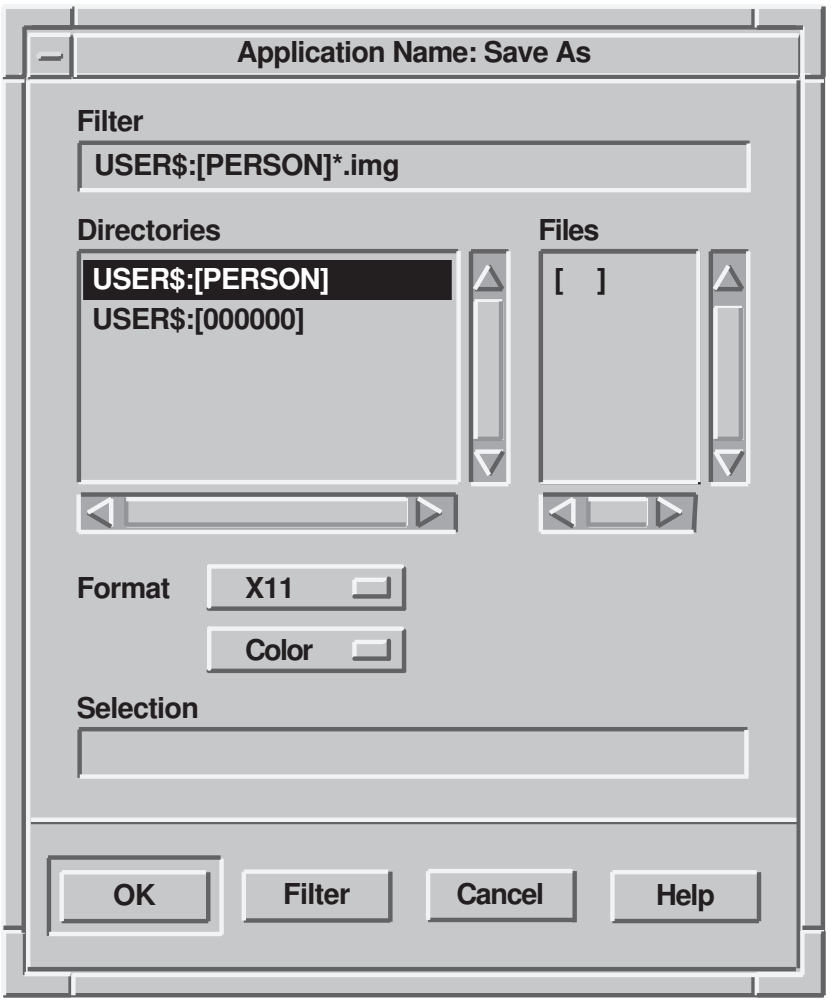
Save As...

The Save As menu item must save the current file under a new name without removing it from the client area. It must also update the name of the file in the title bar or status area. Save As must generate a dialog box that prompts users for a file name. Figure 2–11 shows a Save As dialog box that not only allows users to save a file by typing a name in the Selection field, but also allows users to save a file in different formats.

If your application runs on an operating system that does not save multiple versions of a file (for example, UNIX or Windows NT operating systems), you can have a message box displayed to warn users whenever saving a file with the name they specified will cause the application to overwrite an existing file. If your application will run on a variety of platforms, you could make the display of this warning message customizable.

When users choose Save As, they usually want to keep the newly named file in the same directory as the current file. If you use a file selection box in a Save As operation, initially set the directory to match the directory of the open file. If the user has performed a Save As operation before, leave the directory in its previous state.

Figure 2–11 A Save As Dialog Box That Allows Users to Save in Different File Formats



ZK–2679A–GE

Revert

Make the Revert menu item display the last saved version of the current file. This menu item deletes all the work done since the user last saved the file. If the operating system (for example, the OpenVMS operating system) supports multiple versions, consider allowing users to perform multiple Revert actions. Disable the Revert item if there is nothing to revert in the application.

Figure 2–12 shows a dialog box displayed by activating the Revert menu item.

Figure 2–12 A Revert Dialog Box

ZK–2680A–GE

Export
The Export menu item is similar to Save As, but causes the application to convert the file format from the native application format to a different one. Users need to specify both the name of the file to be exported and the new format that they want the application to convert the file to.

Print
The Print menu item must print the current file using the current settings of a Print dialog box without displaying the box.

Print...
Make the Print menu item generate a Print dialog box. For information on a Print dialog box, see Section 6.1. Digital suggests that if you give Print an accelerator, make the accelerator Ctrl+P.

Close
The Close menu item must close only the current primary window and its associated secondary windows; that is, the window family. You are encouraged to supply this menu item for applications with more than one primary window so that users can remove a window without going to the window menu on the border or without running the Motif window manager. Make Close perform the same action as the Close menu item from the window menu. Digital suggests that if you give Close an accelerator, make the accelerator Ctrl+C.

Closing the last primary window is the same as exiting the application. Use the following guidelines to help you decide what constitutes the last primary window:

- If your application has primary windows that are of equal importance, you should design the application so that closing the last window will exit the application. For example, if your application is a text editor where each primary window is a duplicate, arrange to have the application exit when the user closes the last window.
- If your application has primary windows whose functions differ, consider designing the application so that there is a main window from which the other windows are generated. Thus, closing the main window exits the application, while closing one of the other primary windows causes that window to be removed from the workspace.

If users can lose data by closing the last window, provide a message box that prompts users to save changes.

Exit

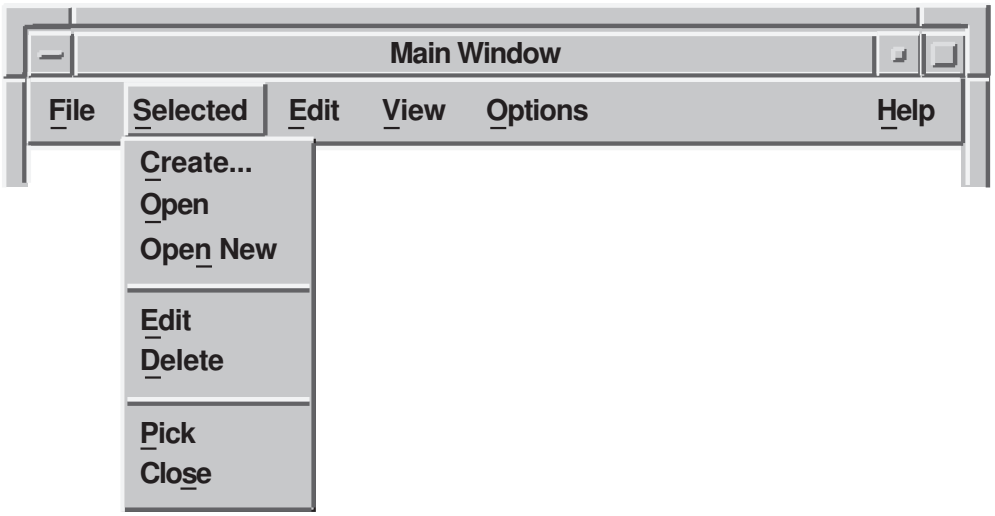
The Exit menu item must end the current session and remove all windows associated with that application. Exit is equivalent to closing the last primary window of your application. To accommodate users who want to exit, but have not saved current changes to a file, generate a message box that prompts users to save changes.

2.8 Items in the Selected Menu

The Selected menu is optional. It contains items that refer to the selected objects in the window. For example, in a mail application, the selected objects could include drawers, folders, or messages. The user would select a series of mail folders and then display the Selected menu. The menu items would apply to the selected folders.

Figure 2–13 shows a standard menu bar that contains a Selected menu with seven menu items. You can use different menu items that are appropriate for your application.

Figure 2–13 A Selected Menu and Its Menu Items



ZK–6410A–GE

Note that the File menu differs from the Selected menu. The File menu contains operations for the application as a whole, as well as file-like actions; the Selected menu items apply only to those objects that the user has just selected.

If an object can be opened in one of several views, consider using a cascade menu for the Selected menu item. For example, if you want the user to be able to open a directory in either an icon or tree view, use a cascade menu instead of a generic Open menu item.

```
Open-> Icon View
       Tree View
```

2.9 Items in the Edit Menu

The Edit menu provides commands that allow users to manipulate text or graphics within a file and between files. Place the Edit menu to the right of the File menu, or Selected menu if one is defined, in left to right environments, and give it a mnemonic of E.

Motif recommends the following items in the Edit menu:

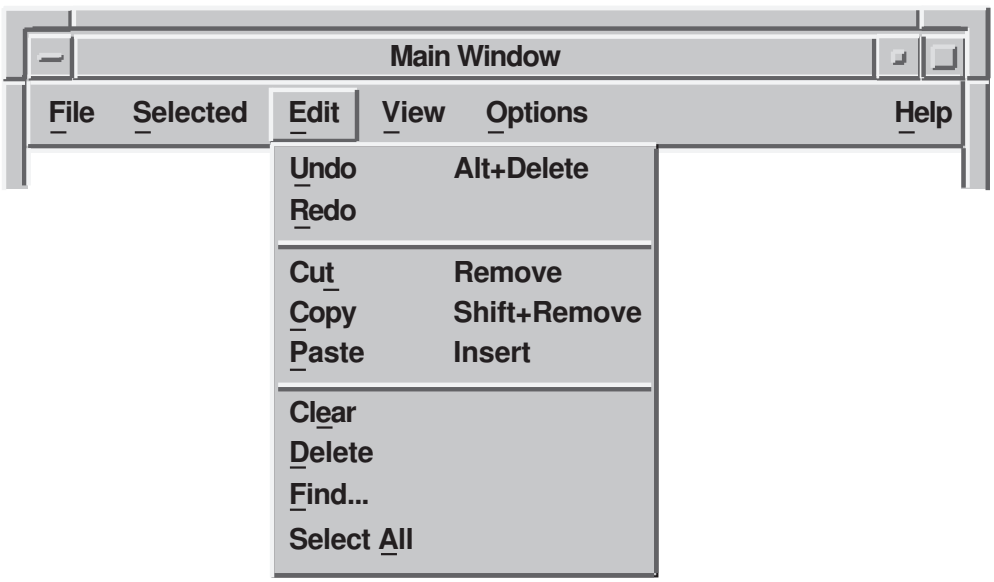
- U ndo (Alt+Delete)
- Cut (Remove)
- C opy (Shift+Remove)
- P aste (Insert Here)
- Cle ar
- D elete

You can enhance the usefulness of this menu by adding the following items:

- F ind...
- Select A ll

Figure 2–14 shows an Edit menu with the additional menu items.

Figure 2–14 An Edit Menu and Its Menu Items



ZK–2546A–GE

Group the menu items according to the following:

- Undo and Redo actions
- Systemwide clipboard actions
- Other actions

The functions of the Edit menu items are as follows:

Undo (Alt+Delete), Redo

The Undo menu item must reverse the effects of a previous operation. The Redo menu item reverses the effects of an Undo operation.

There are two forms of Undo — single-level and multilevel — as follows:

- If your application supports only single-level Undo, the Undo item appears in the menu by default. Once users perform an Undo operation, make the menu item change to Redo until users perform another action that reenables Undo.
- If your application supports multilevel Undo, include menu items for both Undo and Redo. Once users perform an Undo operation, they can either perform another Undo or choose Redo.

When an Undo operation is not possible in a given context, disable the Undo menu item. For example, when users open a file, keep the Undo menu item disabled until they make a change to the file. If your application does not support the Undo function, do not display the Undo menu item in the Edit menu.

You can make the Undo operation more specific by altering the menu item for Undo as the function changes. For example, change the label of the menu item to any of the following when appropriate:

- Undo Cut
- Undo Paste
- Undo Rotate
- Undo Typing
- Undo Fill
- Undo Font

Once a user operation is undone, have your application support redoing the operation. For example, change the label of the menu item to any of the following when appropriate:

- R edo Cut
- R edo Paste
- R edo Rotate
- R edo Typing
- R edo Fill
- R edo Font

Cut (Remove)

The Cut menu item must transfer selected information to the clipboard and delete the information from your application. Decide whether the area that was occupied by the removed data is left blank or whether the remaining data fills the space. Usually graphics applications leave the space blank, while text applications fill the space with the remaining text.

Copy (Shift+Remove)

The Copy menu item must transfer the selection to the clipboard without removing the original data from the client area. Do not allow this action to alter data in your application.

Paste (Insert Here)

The Paste menu item must copy data from the clipboard into the application. When there is more than one possible destination for the paste, use the following guidelines:

- When the focus policy is explicit, paste data to the control with the keyboard focus.
- When the focus policy is implicit (pointer), paste data to the control with the destination cursor. This guideline applies only for text widgets.
- Allow users to place the selected data by dragging an outline of the data to be pasted around the screen using MB2, and dropping it in the desired location by releasing MB2.

Decide whether the pasted data is reformatted to fit in the client area. Text applications usually reformat the pasted data to fit into the margins of the text field; graphics applications usually do not.

Do not allow Paste to affect the contents of the clipboard.

In addition to the Edit menu items, you should implement the Motif drag-and-drop functions.

Clear
The Clear menu item must delete the selection without copying it to the clipboard. The Clear menu item does not compress the remaining data to fill the space that was occupied by the cleared data. Users must be able to undo a clear operation.

Delete
The Delete menu item must remove the current selection without copying it to the clipboard. Make Delete compress the remaining data to fill the space that was occupied by the cleared data. Allow users to undo a delete operation.

Find...
The Find menu item enables users to search for an object that they specify. When they select the Find menu item, make a search dialog box appear.

Select All
Make the Select All menu item select all the data in the file, not just the data that is currently visible.

2.10 Items in the View Menu

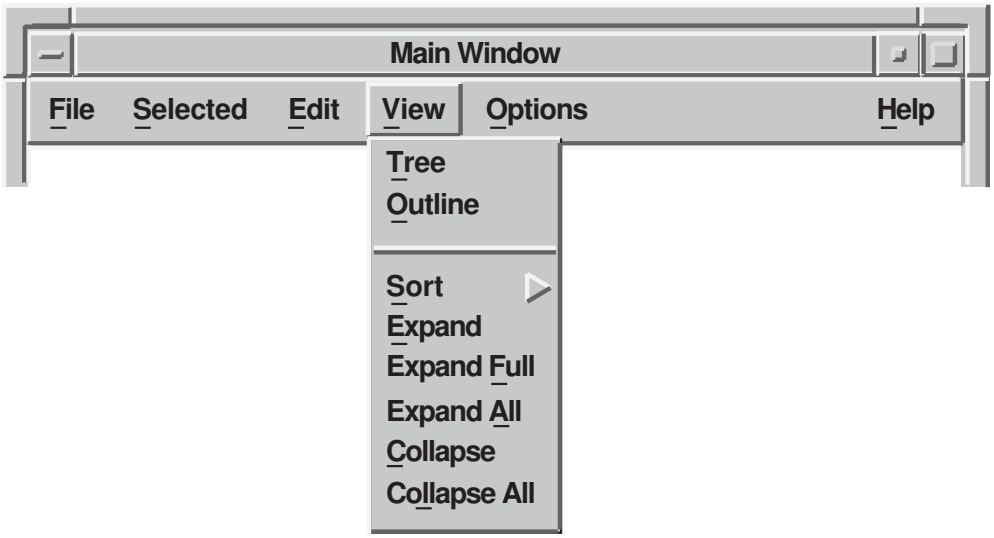
The View menu is optional. It contains items that change the user's view, but not the actual data.

For example, in Figure 2–15, the first group of items changes the entire view by displaying a tree or outline view; the second group expands or collapses the items in the view.

The Expand menu item expands the currently selected item one level deeper. Expand Full expands the selected items to their full depth. Expand All expands all the items to their full depth.

Collapse contracts the selected items. Collapse All contracts all the items.

Figure 2–15 A View Menu and Its Menu Items



ZK–6409A–GE

2.11 Items in the Options Menu

Provide commands in the Options menu for customizing your application. The menu items in the Options menu depend on your application. Give the Options menu a mnemonic of O. For more information and an example of an Options menu, see Section 3.1 and Figure 3–1.

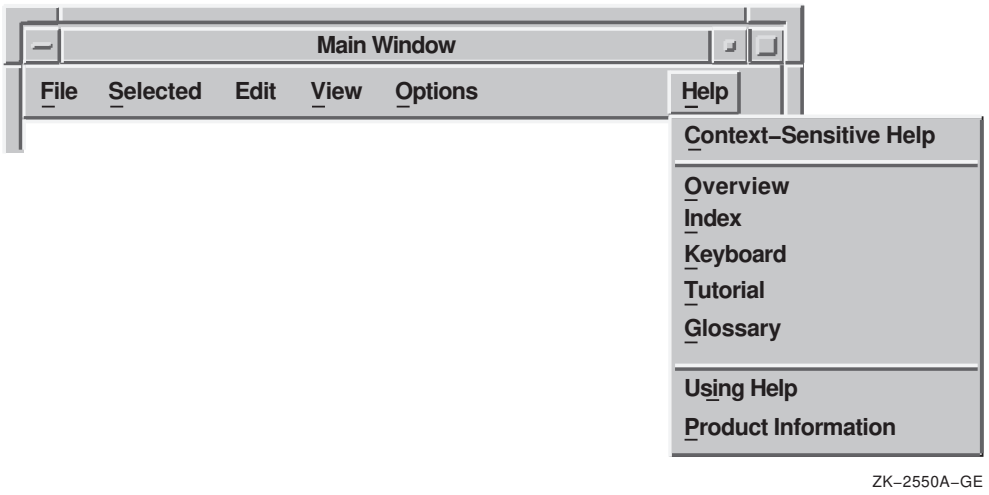
2.12 Items in the Help Menu

The Help pull-down menu can contain any of the items shown in Figure 2–16.

Note that OSF/Motif does not require your application to have all these items on a Help menu. However, if your application has one of these items, it must conform to OSF/Motif specifications.

Include only the Help menu items that your application supports.

Figure 2–16 A Help Menu and Its Menu Items



The following list shows the menu items and the results when users choose them:

Context-sensitive Help
Initiates context-sensitive help by changing the shape of the pointer to a question mark. The user then moves the pointer to the area or item on which help is needed and presses MB1. After the selection, the pointer returns to its previous shape.

Overview
Provides a brief description of the current window. The description includes information about the function of the window and the tasks that users can perform using that window.

Index
Gives users an index to all help information in the application.

Keyboard
Shows the meanings assigned to keys, especially function keys, by your application. You can use a Shortcuts menu (**Shortcuts**) instead of the Keyboard menu item to show the meanings of both key and button bindings.

Tutorial
Provides access to an online tutorial about your application.

Glossary
Provides an alphabetical list of terms used in the online help.

Using Help
Provides information on how to use the help facility.

Product Information
Provides information about your application, including the name, version, date, and copyright data.

Place any additional menu items between Index and Using Help.

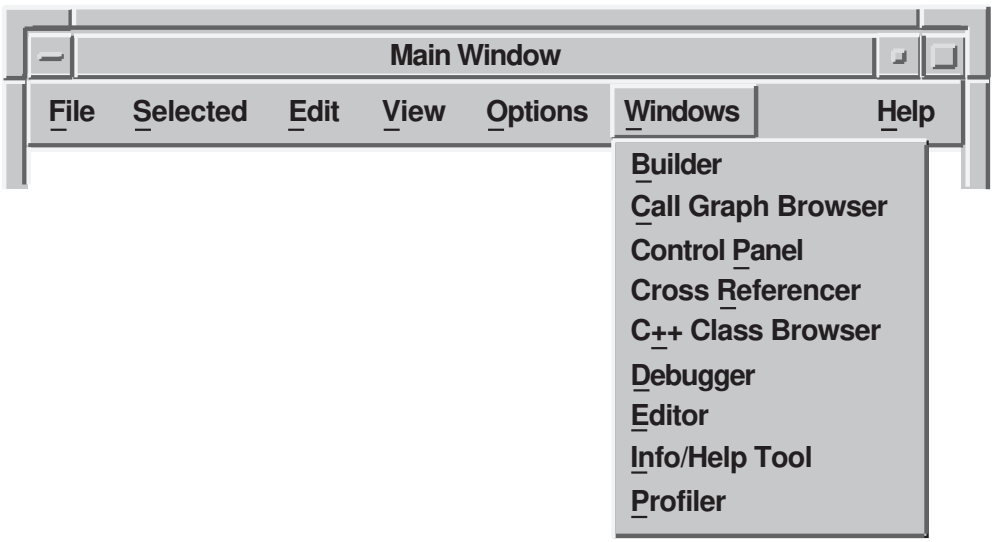
For information about designing and writing online help, see Chapter 5.

2.13 Items in the Windows Menu

The Windows menu is optional. If your application uses multiple main windows, place the titles of currently active primary windows in the Windows menu. When a user selects a title from the Windows menu, display that window. A Windows menu gives the user immediate access to any active primary windows in your application without the need to shuffle through the windows on the screen. In some instances, you might also want to place the names of currently active secondary windows in the Windows menu.

Figure 2–17 shows an example of a Windows menu.

Figure 2–17 A Windows Menu



ZK–6408A–GE

3
Customizing the Interface

Design your application so that it follows the principles of good user interface design, as explained in the *OSF/Motif Style Guide*. In addition, design your application so that users can customize important functions to meet their specific needs. Customization is discussed only in this document and not in the *OSF/Motif Style Guide*.

Allow users to customize your application by using menus and dialog boxes. Provide an Options menu in your application to help users tailor the application to their needs.

No matter what type of user interface you are creating, consider allowing users to customize the following:

- Window positions
- Colors
- Size and rendition of text

Other types of customization depend on your application. These types of customization include startup procedures, keyboard accelerators, and print and plot settings.

This chapter provides general guidelines for customizing your application, and some specific guidelines for customizing startup procedures, window positions, colors, and text.

3.1 General Guidelines for Providing Customization Features

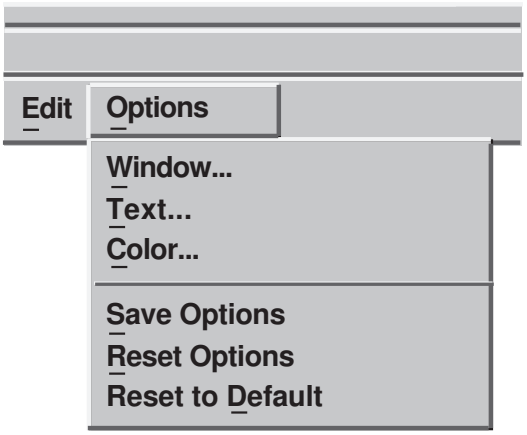
Allow users to customize your application by using the Options menu. Divide this menu into at least two sections. In the top section, place the items needed to customize specific features of the application. For example:

- Command menu items
- Dialog menu items
- Submenus containing command or dialog menu items

In the bottom section, place items for saving and restoring settings.

Figure 3–1 illustrates an example Options menu.

Figure 3–1 An Example Options Menu



ZK–2528A–GE

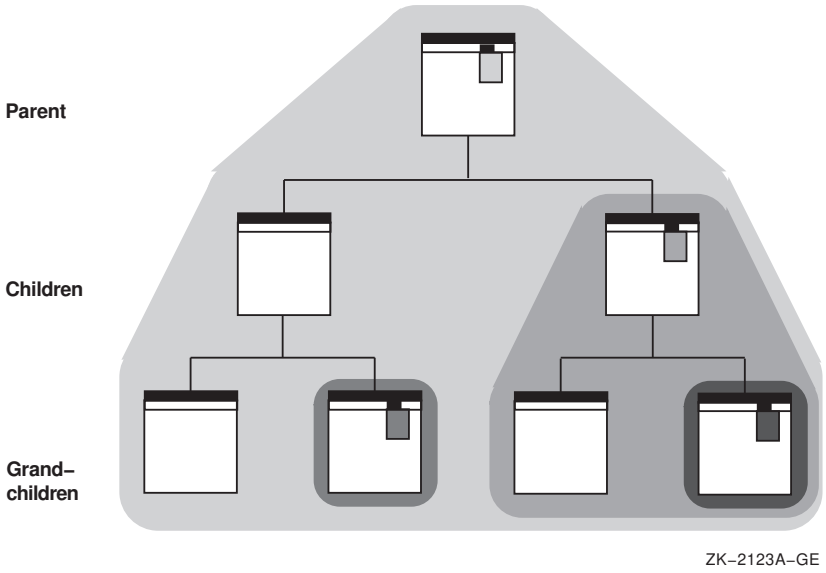
If your application has more than one window, provide an Options menu in the main window to customize settings throughout the application. Then, give each additional window an Options menu for settings specific to it.

Place the Options menu on the menu bar to the right of all other menu names except Help.

Your application can provide customization of settings in windows at any level. Make customized settings apply to the window in which the user specifies it, as well as to all its descendant windows. In addition, allow a customized setting in a child window to override the same setting specified by its parent.

Customization can be inherited from a parent window or specified in a child window. For example, a dialog box inherits its font setting from the mail application that displays it unless the font is customized at the dialog box level. In Figure 3–2, several windows provide customization for a particular setting, represented by a shaded pull-down customization menu. The range of customizing at different levels is indicated by the shaded areas. The setting represented by each menu shade is in effect throughout the range of windows within the area of the same shade.

Figure 3–2 Range of Customization



If you do not provide customization in a window, or if the user has not chosen settings, have the window inherit its settings from its parent window.

Have your applications provide customization for each setting at the highest possible level at which the setting is meaningful. For example, customize a setting at the application level if it makes sense for each of the application's tools to share that setting.

In addition, applications can provide overriding customization in lower-level windows in the following situations:

- The feature has particular significance or importance in the lower-level window.

  For example, in a mail application, users can customize the editor they use by choosing the Send... menu item from the Options menu in the main window. However, they can also change the editor in the Create window by choosing a menu item in the Change Editor menu.

- The feature will be changed frequently in the lower-level window.

  For example, in a mail application, users can customize their personal names by choosing the Send... menu item from the Options menu. However, because users may want to change their personal name

frequently depending on the content of the message, they can also change their personal names in the Create window.

Retain the results of settings until users exit the application, unless the users save them. When users exit the application, consider having a dialog box that asks them to save settings.

If users save the settings, use them the next time the application is invoked and have the settings remain in effect until users choose new settings.

3.2 Designing Options Menu Items and Dialog Boxes

Give the Options menu and associated dialog boxes all the customization features appropriate for your application. However, excessive or gratuitous customization can overload the user with too many decisions to make. It may not be necessary to include all the customization described in this chapter in your application.

Know your users' needs; provide flexible customization options for important features, but select a good standard setting and provide no customization for unimportant features. Provide customization for features that will help users feel more comfortable and in control, and that will make their tasks easier to perform. When appropriate, provide a list of common defaults rather than an open-ended text field. Use an option menu widget or a list box for the list of defaults. Such lists enable users to customize the application rapidly.

Remember that even if your application is not being translated, it may be used throughout the world by people for whom English is a second language, so you might want to allow users to customize time, date, and monetary units.

As you design your application, you may become aware of different ways to allow users to do the same thing. In such cases, consider providing settings that enable users to choose the method they prefer. For example, the Motif window manager allows users to choose between an explicit (click-to-type) and an implicit (pointer) focus policy.

You can add menu items, submenus, and menu items that lead to dialog boxes. The following sections give you a few guidelines for adding menu items and dialog boxes.

3.2.1 Adding Command Menu Items

Use menu items to allow users to customize one setting at a time. For example, use a check box in a menu such as Highlight Changes to allow users to turn highlighting on or off.

ZK–6385A–GE

If you have a menu with many items, group them within the menu by using separators, like the example Options menu shown in Figure 3–1.

For more information on designing menus, see Chapter 2.

3.2.2 Adding Dialog Boxes

Use dialog boxes to allow users to customize several things at once. Decide when to make separate dialog boxes for customizing. For example, decide whether to make a single dialog box for customizing text and color options, or whether to make a separate dialog box for each.

Here are a few guidelines for using dialog boxes:

- Find logical groupings of customizable features.

  For example, group all choices for customizing text (font, rendition, point size) in one dialog box, and all choices for customizing graphical objects (color, thickness of lines) in another.

  Involve users to tell you what they think is a logical group. You do not need a working prototype to do this. Consider giving users a generic dialog box drawn on paper and cutouts of controls that they can arrange, or having them redraw dialog boxes using a graphics editor or other prototyping tool.

- Provide feedback areas or include an Apply push button to show the effects of various settings without actually changing them.

  See the Sample Text section of Figure 3–3 for an example.

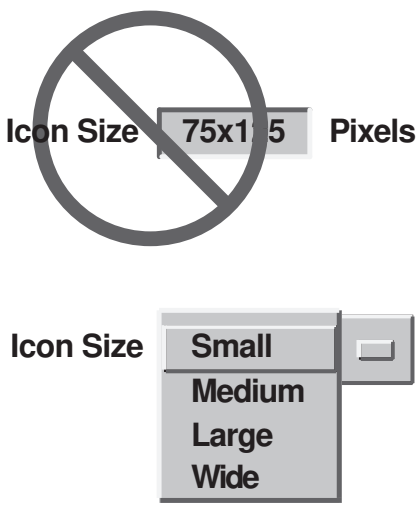
- Include a Reset button so users can reset the information in the dialog box without canceling it.

- Keep an appropriate number of tab groups in one box; avoid clutter.

  Involve users so that they can let you know when they feel you are putting too many options in one dialog box.

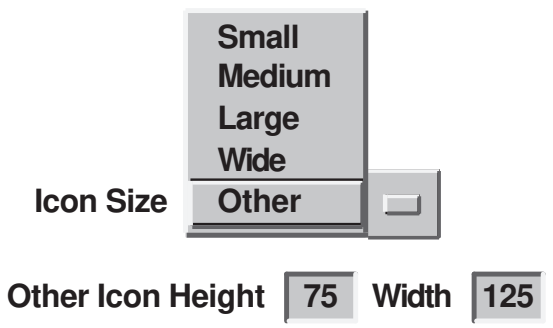
- Use an option menu widget, list box, or other widget to provide a list of the default values for options. Whenever possible, avoid using open-ended text fields or numeric scale widgets.

  The following figure illustrates an inappropriate and an appropriate way to enable users to customize the icon size.



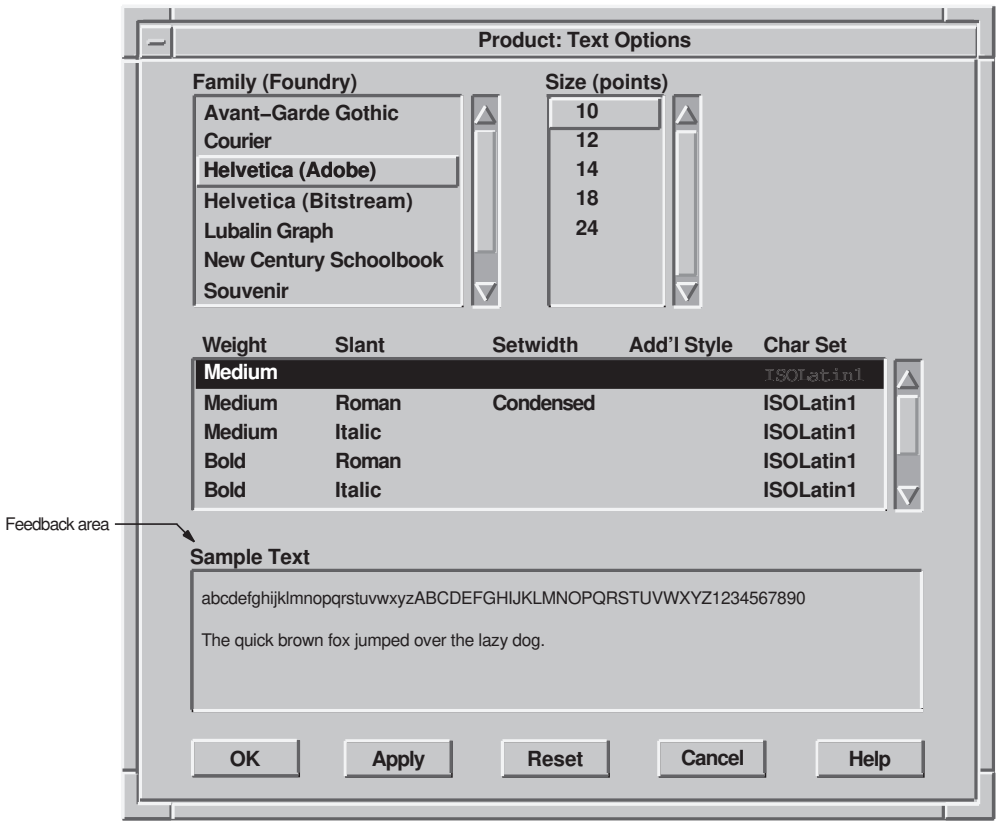
ZK–6383A–GE

When full user flexibility is needed, include an Other option. When the user selects Other, enable a text field as shown in the following figure.



ZK–6384A–GE

Figure 3–3 Feedback Area Labeled "Sample Text"



ZK–2542A–GE

3.3 Guidelines for Customizing Windows, Color, and Text

This section provides specific guidelines on customizing the following:

- Window positions
- Colors
- Size and rendition of text

3.3.1 Window Positions and Sizes

Allow users to customize and save the positions of windows so that each time they invoke the application, the windows appear in the same positions. There are several ways to do this:

- Save the positions and sizes of windows automatically when users exit the application.
- Use the Options menu item Save Options to save the positions of windows automatically, in addition to saving other options.
- Create a menu item in the Options menu called Save Window Positions.

Do not make users specify the X and Y pixel values for window positions. Users generally prefer direct manipulation. If you must provide window position dialog boxes that allow users to specify X and Y pixel values (for example, in a program where exact placement is very important), also provide a way for users to customize their window placement through direct manipulation.

3.3.2 Colors

Allow users to customize and save the colors of an application so that each time they invoke the application, the same colors appear. If you are using color to show scale or intensity, allow users to choose colors that are meaningful to them. Try to keep users from doing counterproductive things, such as making both the background and the foreground the same color.

Digital provides the ColorMix widget that you can include in your application to enable users to customize their colors. The ColorMix widget provides users with a consistent way to customize colors across applications. For information on the ColorMix widget and on using color in your application, see Section 3.3.2.

If your application does not update color changes immediately, you might want to provide feedback by displaying a message dialog box that tells users when they will actually see the color changes take place.

3.3.3 Size and Rendition of Text

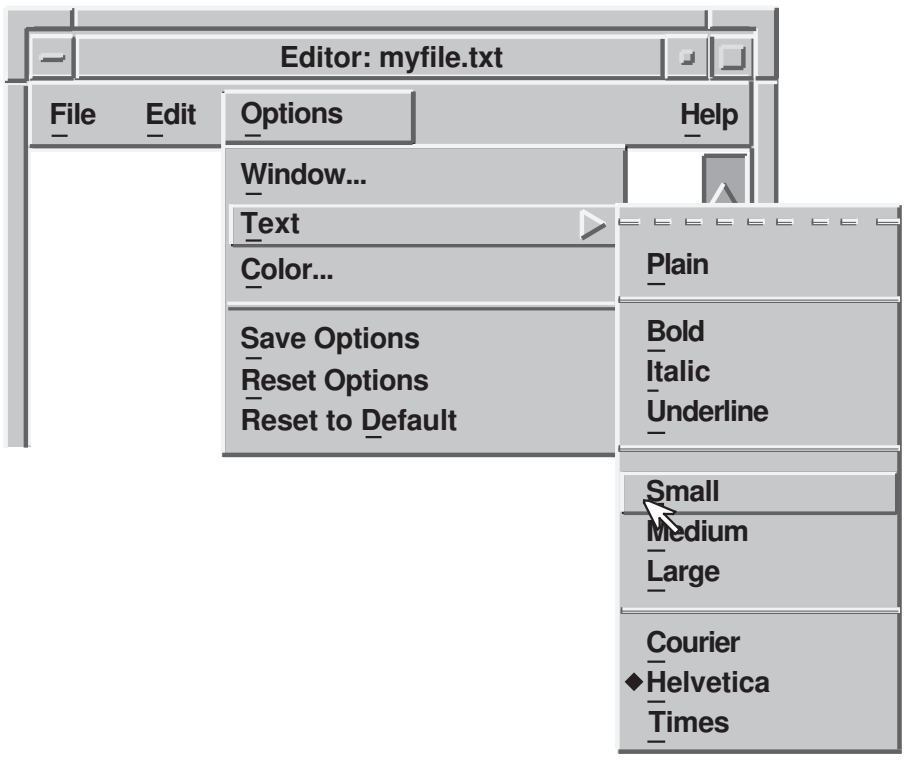
If your application uses text, allow users to customize that text by changing font family, point sizes, and rendition. Allow users to save customized text settings so that each time they invoke the application, the customized settings are invoked. If your application allows for limited customization of text, place the text customization menu items in the Options menu to be consistent with other aspects of customization. In cases where you allow extensive or sophisticated customization of text (in text processors, for example), provide a

separate menu because your application will allow users to customize text in many different ways.

The way that you implement customization of text will depend on the number of different font families, point sizes, and renditions that you provide, and on how often users will be changing text settings.

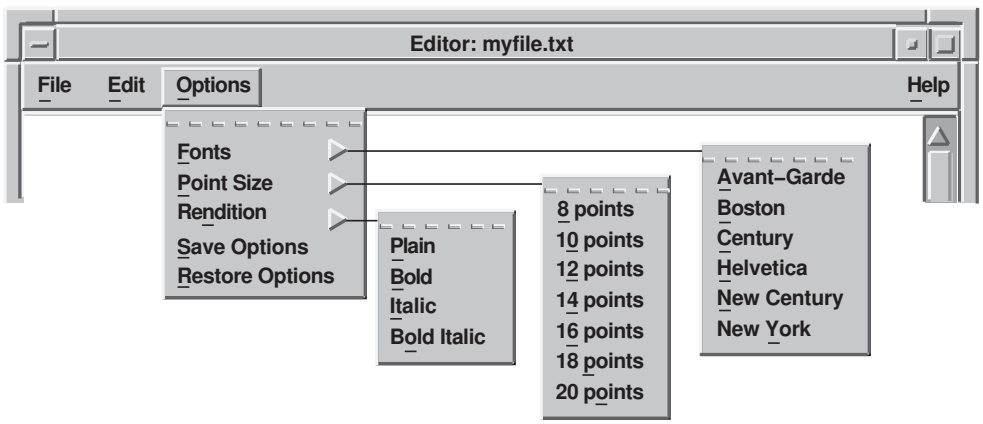
If you supply a few choices, consider having one submenu that allows users to customize the font family, point size, and rendition, as in Figure 3–4.

Figure 3–4 Using One Menu to Customize Text



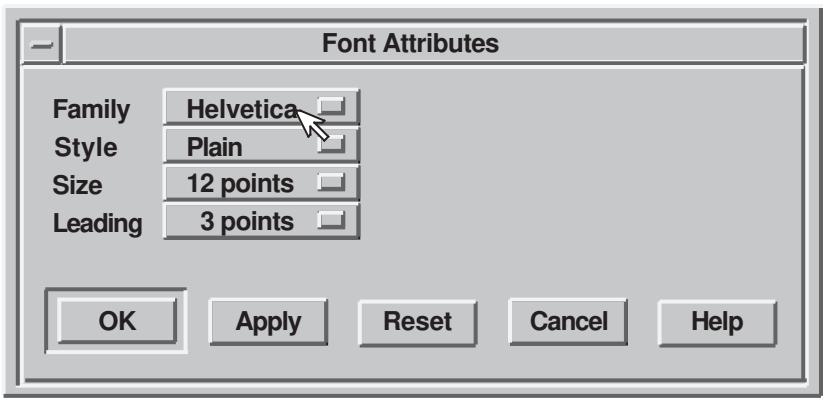
If there are more than a few choices and if users will not be customizing text often, consider having a series of submenus, as shown in Figure 3–5.

Figure 3–5 Using Several Menus to Customize Text



If users will be customizing text often, consider combining choices of font family, point size, and rendition into one dialog box to avoid making users pull down a series of menus to accomplish one task. For example, create a dialog box similar to the one in Figure 3–6.

Figure 3–6 Using a Dialog Box to Customize Text



ZK–2539A–GE

4
Using Color

This chapter discusses the following topics concerning color:

- Why use color?
- General guidelines and specific recommendations
- Digital's ColorMix widget

4.1 Why Use Color?

The primary reason for using color is to help users. Color can help users accomplish their tasks by:

- Making important things stand out

  For example, in a spreadsheet, display all negative numbers, or the cells containing these numbers, in red.

- Showing relationships among objects

  Use color to group related items. Displaying related objects in the same color can be especially useful in densely packed displays. For example, on a spreadsheet, group all income from one source in one color, and all income from another source in another color.

- Allowing users to see differences quickly and consistently

  The eye is attracted to color over every other feature on an interface (except flashing). Thus, use color for important things such as error messages because users' eyes will be attracted quickly to the color.

- Indicating keyboard focus

  For example, in a Motif environment, users can have many windows open at one time. Using a specific color border to indicate input focus helps users locate the window with keyboard focus quickly and consistently.

- Corresponding with conventions used in certain professions

  For example, if your application helps people design objects such as maps or buildings, it is important to keep color connotations consistent between the traditional colors used on paper renderings and those appearing on line. For example, mapmakers might expect to see blue water in the maps they design with a survey application.

4.2 General Guidelines and Specific Recommendations

Remember that the purpose of using color is to help users accomplish tasks. Color is not always needed and may not be available. Never give your application color-specific features that disappear in a black and white interface. For these reasons, always design the interface for black and white (monochrome) displays first. The following list shows a few reasons to design in black and white first:

- Some people are colorblind.
- Your application might run on both color and monochrome displays.
- Color printers are uncommon and expensive; many users may be printing on black and white printers.

4.2.1 Before You Select Colors

Before you select colors for your application, consider the following:

- The visual abilities and disabilities of the users
- The color preferences of the users (in terms of nationality, culture, background, previous experience, and so on)
- The physical environment in which users use the computer (including the country, company, and specific work environment)
- The areas where you want to focus the users' attention
- The assignment of colors in related applications

4.2.2 After You Select the Colors

After you have selected some colors to work with, keep the following guidelines in mind:

- Limit the number of colors.
  - Use color as a redundant cue. Make sure that users can identify an object by its shape or intensity in addition to its color. For example, stop signs in the U.S. and many parts of Europe are identified by their red color and octagonal shape.
  - Use color sparingly. Never vary the color just for variety. Try to use only four or fewer colors in one window, and try to use only seven or fewer colors in an entire application. Research has shown that users have trouble distinguishing among more than seven colors in a user interface.
  - Minimize the use of bright, saturated colors. Limit them to items that must be noticed. If you overuse these colors, the attention-getting properties will be lost.
  - Use white, black, or shades of grey for objects (like text) that users will view for a long time.
  - Do not depend on subtle changes in color to convey important information. Users may not notice the change.
- Have a purpose for each color
  - Reserve red for warnings and negative numbers (at least in European and American displays, where red connotes danger). Because it is the most noticeable color, use it with care and use an unsaturated red color. The full red color may be too jarring for users.
  - Provide a color key so that users can compare colors if you use a color code as a primary coding mechanism.
  - Use color codes only under the following circumstances:
    * When users are already familiar with a color code
    * When you must catch the users' attention
    * When users must search for categories of information
    * When related applications use a color code
  - Do not use the same color to represent two different meanings.

- Use existing conventions for color codes whenever possible, such as red for deficits and black for profits in business applications.
- Make sure that the colors you use do not conflict with color connotations in other countries, if your application will be used in those countries. Or allow users to customize their colors.

- Be careful about what colors you use together.
  - Do not use color coding for displays located in the periphery of the screen. Color receptors are located mainly in the focus point of the eye; thus, users may have difficulty perceiving colors located on the periphery.
  - Make adjacent colors differ in at least two of the three primary colors (red, green, blue), so that color-weak users will be more likely to distinguish the differences.
  - Do not require users to distinguish between green and red without other cues; green-red color confusion is the most common type of color weakness.
  - Be aware of the colors surrounding colored targets. Avoid situations where the simultaneous contrast effect could allow users to misinterpret information. For example, a yellow symbol will appear greener when surrounded by red, and redder when surrounded by green.
  - Display bright colors on dim backgrounds, or dim colors on bright backgrounds. This maximizes the contrast.
  - Do not use blue for characters, lines, or edges, unless your intent is to make the object look indistinct and fade into the background. Blue is difficult to see. If used for edges or text, these items will appear fuzzy and blurred.
  - Do not use saturated red and blue in adjacent areas; these colors induce a disturbing depth effect.
  - Do not use saturated red and green in adjacent areas. These colors can induce a disturbing spurious motion effect.

    For more information about using color, see the list of references in Appendix C.

Specific Recommendations

Here are some specific recommendations to follow concerning color in dialog boxes, menus, and pointers.

- Dialog Boxes

  Arrange and design dialog boxes for monochrome displays. There is probably no reason to use color for the objects in a dialog box, unless it is a dialog box for selecting colors, as in a drawing or painting program.

- Menus

  Design menus, like dialog boxes, for a monochrome display. The primary instance in which you need to use color in menus is when you create a menu for choosing colors. There are a few other cases when color can be helpful in menus, for example, using red letters for a menu item labeled ''Important.''

- Pointers

  Make sure that the pointer is visible over all areas of the screen. Using black with a white border is probably the safest choice, even with color displays.

4.3 Using Digital's ColorMix Widget

Use Digital's ColorMix widget to provide users with the ability to customize the colors used in your application. For example, if your application includes graphics, such as a pie chart, you can provide access to the ColorMix widget through an item in a customization menu to allow users to define the pie chart's colors.

The ColorMix widget provides users with immediate feedback, displaying each new color as it is defined. There is also a scratch pad feature that allows users to save colors that they want to use later or to compare colors.

By default, the ColorMix widget supports five color models. You can, however, customize the ColorMix widget to support other color models.

Color models are abstractions that enable users to choose specific colors. The ColorMix widget supports the following color models:

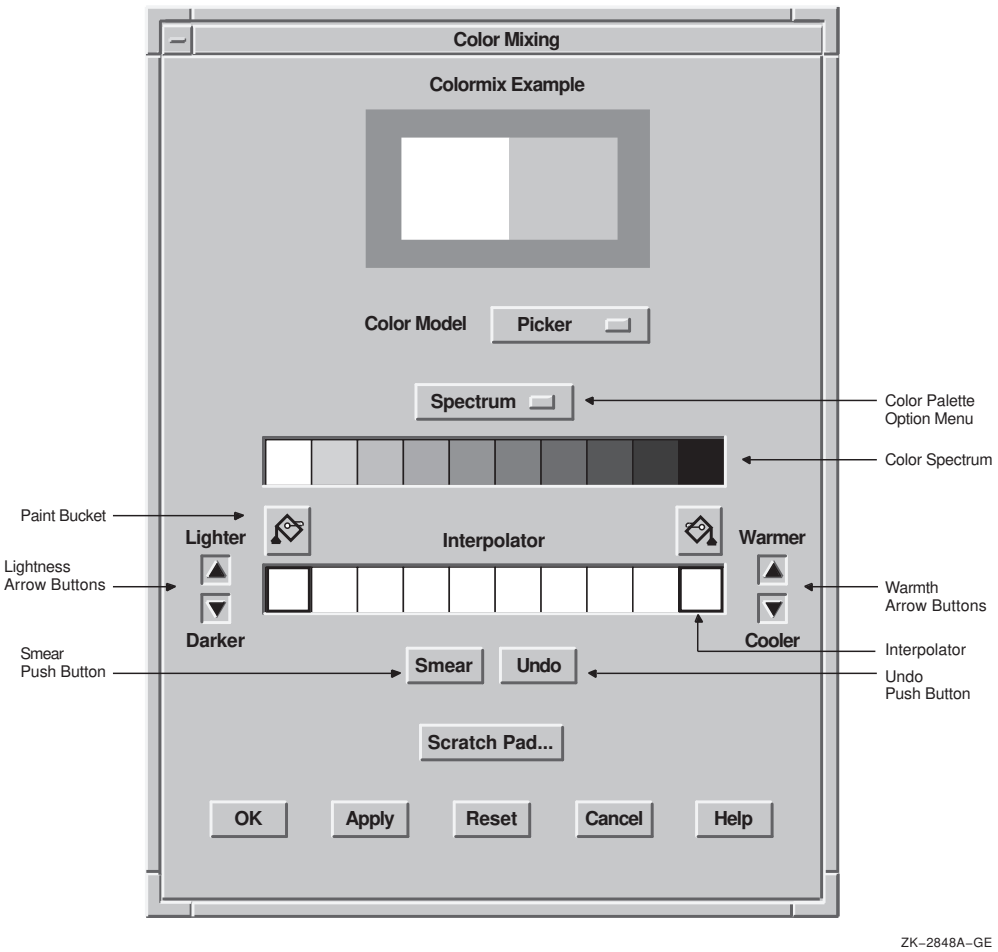
- Color Picker

  The Color Picker is the default color model for the ColorMix widget on color systems. (Note that the Color Picker model does not display on noncolor systems or on systems with too few resources.) The Color Picker allows users to pick a color from a static color spectrum (or another color model) and either use that color without changing it, or smear it into a color they

want. Users can also pick colors from the spectrum and smear them into each other to create a color they want.

Figure 4–1 shows the components in a color mixing widget with the color picker model selected.

Figure 4–1 Color Picker Model



- Hue, lightness, and saturation (HLS)

  In the HLS (Hue, Lightness, Saturation) color model, a color is specified by three characteristics: hue, lightness, and saturation. Hue is color. Lightness describes the intensity of the color; that is, the amount of the color. Saturation describes the purity of the color; that is, how much the color is diluted by white.

  HLS expresses hue as a continuous spectrum of values arranged in a circular pattern. Red appears at 0 degrees (and again at 360 degrees), magenta is at 60 degrees, blue is at 120 degrees, cyan is at 180 degrees, green is at 240 degrees, and yellow is at 300 degrees. HLS expresses the lightness of a color as a percentage between 0 and 100 percent. One hundred percent lightness creates white light; zero percent lightness creates black.

- Red, green, blue (RGB)

  The RGB color model is the default color model for the ColorMix widget on monochrome systems.

  In the RGB color model, a color is specified as a mixture of different intensities of red, green, and blue. You can specify the intensity of red, green, or blue as a value between 0 and 65,535. Zero is the lowest intensity. Black is defined as a zero-intensity value for all three colors; white is 100 percent intensity for all three colors.

- Browser

  The color browser is a window that presents the user with a list of X11 named colors. Each button in the window shows the name of an X11 color. If enough resources are available, the background is set to that color. Users can scroll through this color list. Clicking MB1 on a color in the list causes the color display subwidget to become filled with that color.

- Greyscale Mixer

  The greyscale mixer allows users to create grey shades ranging from black to white. When the greyscale mixer is selected, the current new color is converted to an appropriate shade of grey, which users can then adjust with the scale.

For more information on the ColorMix widget, see the *DECwindows Motif Guide to Application Programming* and the *DECwindows Extensions to Motif*.

5
Designing Help

Provide online help as part of your application. The methods and techniques of designing online help vary with each company; in some companies, technical communicators design the online help, while in other companies application developers design it. Regardless of who designs and implements the online help, this chapter covers the following:

- Methods of accessing help
- Guidelines for designing and writing help
- Digital tools for creating online help
  - The Help widget
  - The DECwindows Motif Help System

5.1 Methods of Accessing Help

Allow users to access online help through one or more of the following mechanisms:

- Help pull-down menu
- Help command
- Help push button
- Help key

5.1.1 Help Pull-Down Menu

Each type of help corresponds to a menu item in the Help menu. Make sure that your Help menu includes only the types of help that your application provides. For an example Help menu and an explanation of what each menu item is to do, see Figure 2–16.

5.1.2 Help Command

If your application makes use of command dialog boxes or command regions, allow users to invoke help by typing the help command at the prompt and pressing Return.

5.1.3 Help Push Button

Provide a Help push button in most dialog boxes. When users push the Help push button, have them receive an overview of the dialog box and additional topics that describe how to use each control within the dialog box.

If your application uses warning or error dialog boxes, provide a Help push button in each dialog box if the error and a solution cannot be given in one or two lines of text. Have the help that users receive describe the error or warning displayed in the dialog box, and provide a suggestion for solving the problem.

5.1.4 Help Key

When users press the Help key, have them receive context-sensitive help on the component that has the location cursor. However, if the input focus is in a dialog box and users press the Help key, have them receive an overview of help for the entire dialog box, not just the component that had input focus.

5.2 Guidelines for Designing and Writing Online Help

Because online help, especially context-sensitive help, must provide separate information about individual tasks or concepts, the information is divided into separate **Help topics**.

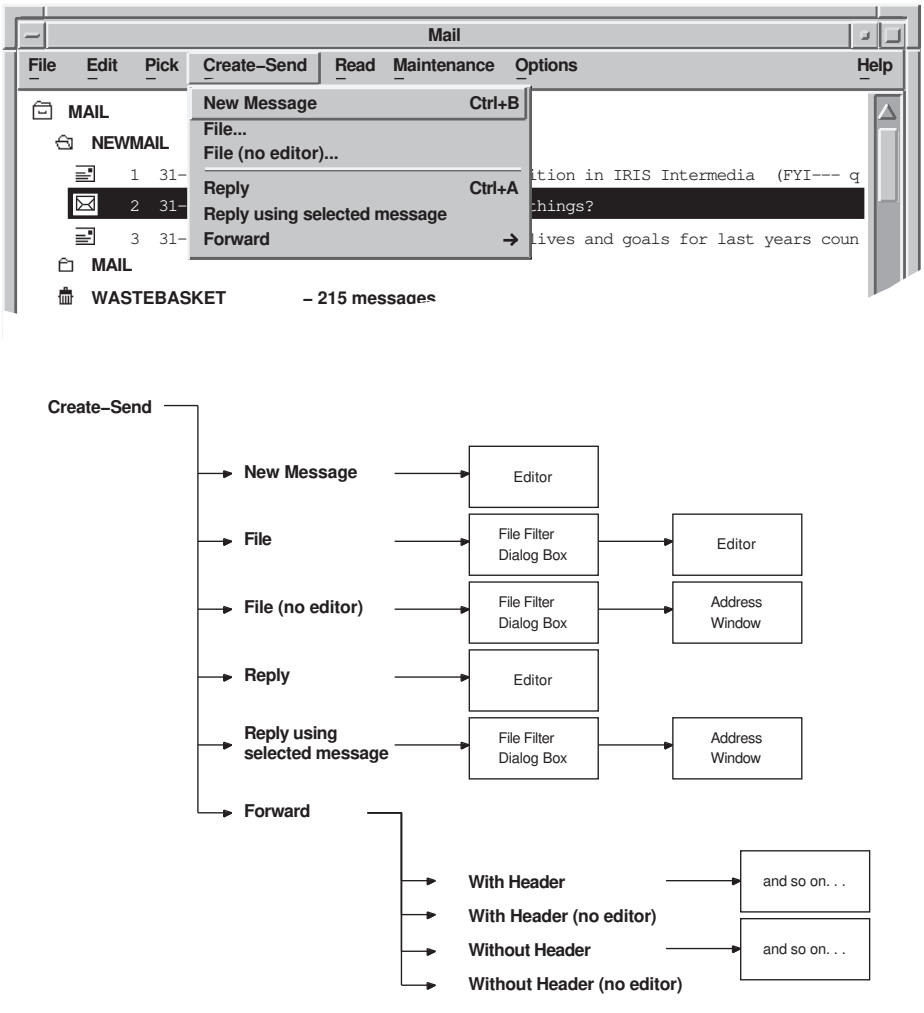
5.2.1 Planning Help Topics

Before you begin to write, plan the topics on which you will write help. Here is a list of steps to guide you:

1. Create a hierarchical list of all the application's screen objects. This includes menus, menu items, pop-up menus, dialog boxes, and controls (fields) within each dialog box.

   For example, one branch of this hierarchical tree for Mail might begin with the Create-Send menu, as shown in Figure 5–1. Each menu item is a smaller branch off the main branch. Dialog boxes and submenus are still smaller branches, and the controls within the dialog boxes are the leaves at the end of the branch. Your diagram might look similar to the one in Figure 5–1.

Figure 5–1 Create a Hierarchical Tree

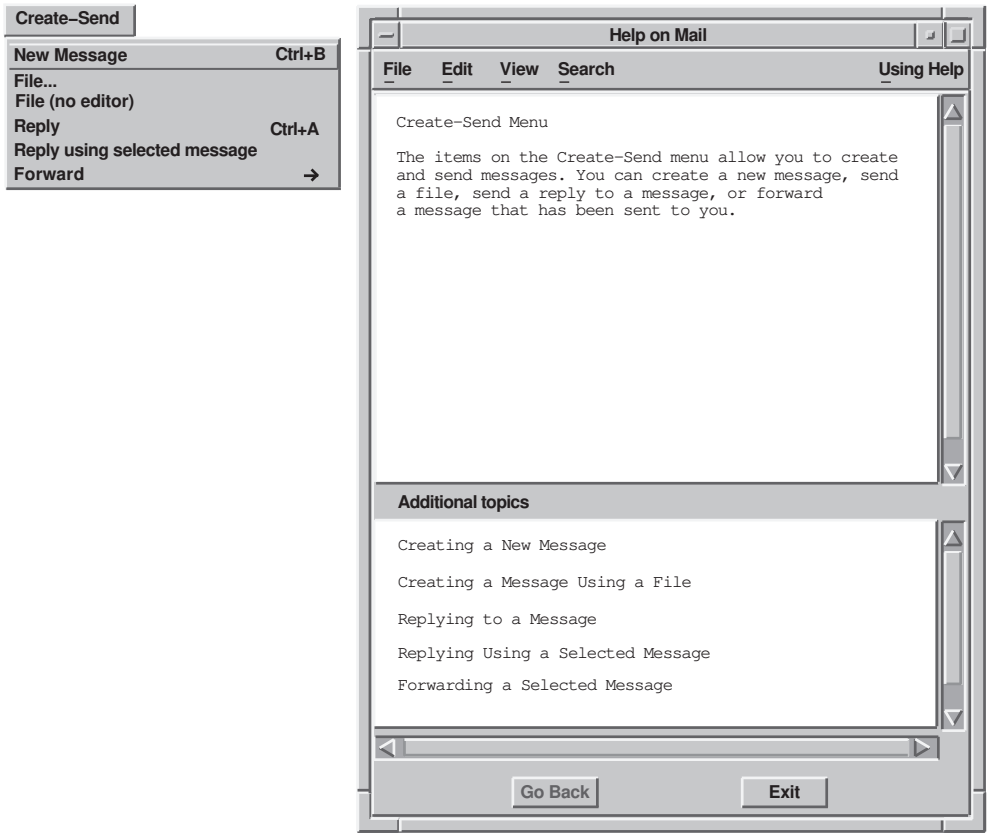


2. Once you have created your hierarchical list, create a list of tasks that users might want to accomplish and check this list with actual users. Then map the tasks back to menus, menu items or dialog boxes; that is, for each task, see if there is a clear means of accomplishing that task.

If you have trouble mapping tasks back to screen objects, it may be because you have not listed representative tasks, or because the user interface has not been designed in a task-oriented fashion. If the latter is the case, see if there are menu names or control labels that can be changed to make the interface more task oriented.

For example, Figure 5–2 shows a Help window that describes the Create-Send menu, and a list of additional topics. Notice that the additional topics correspond to the items in the Create-Send menu, and each task-oriented additional topic can be mapped back to a menu item. For example, the first topic says Creating a New Message and the associated menu item is New Message.

Figure 5–2 Create Task-Oriented Topics



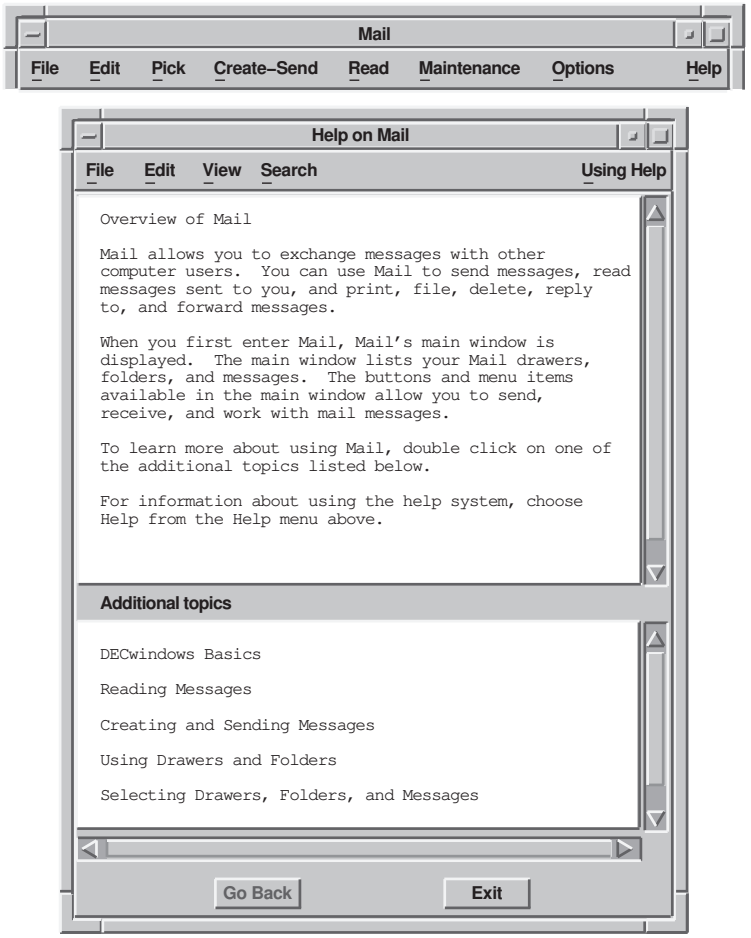
5.2.1.1 Organizing Help Topics

Once you have created a hierarchical list of the screen objects and their associated tasks, you are ready to begin organizing your help topics. You may want to use the following steps to help guide you:

1. Write an overview of the application that quickly describes what tasks users can accomplish with it. List as additional topics (or create DECwindows Motif Help System hotspots for) the general tasks that users can perform using the menus in the title bar. You may also have other additional topics.

For example, Figure 5–3 provides an overview and introduction to mail, and as additional topics it lists ‘‘Reading Messages’’ to correspond with the Read menu, and ‘‘Creating and Sending Messages’’ to correspond with the Create-Send menu.

Figure 5–3 Write an Overview for the Application



ZK–2548A–GE

2. For each menu, write a short overview of what tasks the items in this menu can help users accomplish. List as additional topics the tasks that each menu item will perform; do not just list the menu items. (Users can see a list of menu items by posting the menu. If users are in help, they need information on performing specific tasks.)

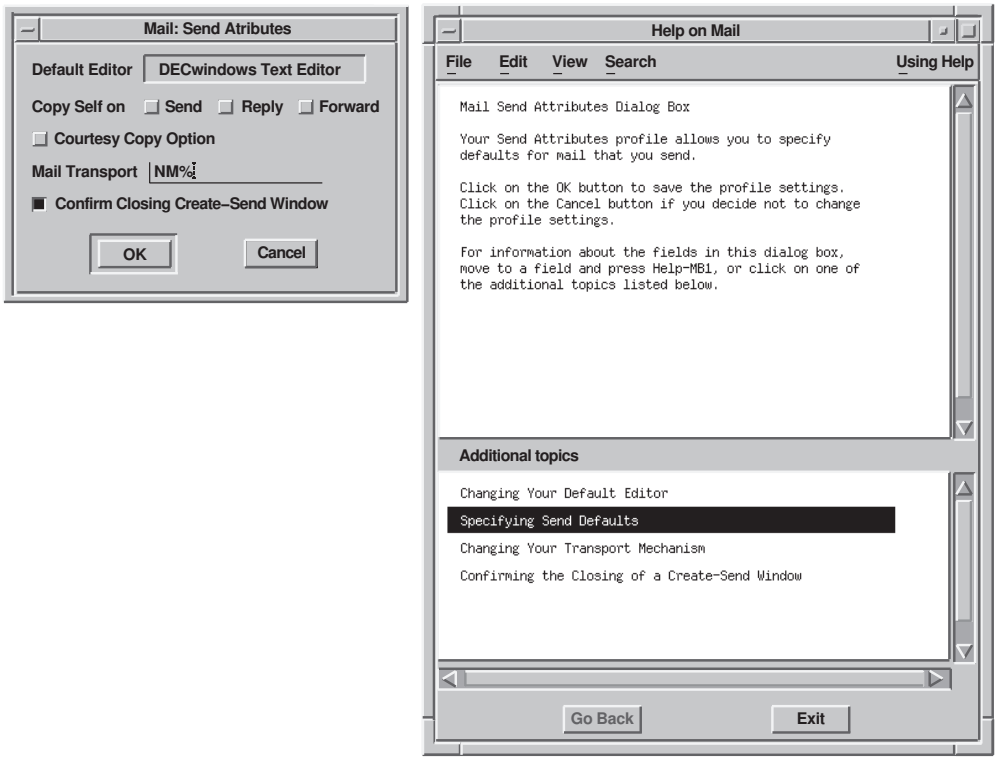
   Notice that the overview of the Create-Send menu in Figure 5–2 has additional task-oriented topics that correspond with each menu item.

   Also, consider using additional topics (or hotspots) for more than simply showing the hierarchical breakdown of an interface; use additional topics to show relationships between various parts of an interface. For example, help on the Create-Send menu item might include an additional topic on how to set a personal name or how to set an editor.

3. For each dialog box, write a short overview of the tasks that users can accomplish using that dialog box and tell users how to accomplish those tasks using the dialog box. If the dialog box is large or complicated, list as additional topics (or hotspots) the tasks that users can perform using each control in the dialog box. Then, in a subsequent Help window, provide detailed information on how to use each specific control.

   Notice that the overview of the Send attributes dialog box in Figure 5–4 lists additional topics that correspond with each control (or set of controls) in the dialog box. The additional topics are listed in a specific order; the list begins with the control in the upper left corner of the dialog box, and goes left to right, top to bottom (in left-to-right environments).

Figure 5–4 Additional Topics in a Help Window



ZK–2363A–GE

4. Decide which Help topic will be invoked when users obtain context-sensitive help using the mouse. For example, when users click on the menu bar, but not on a specific menu, will you invoke an overview of the application, an overview of the menu bar (if you have decided to write one), or help on the closest menu?

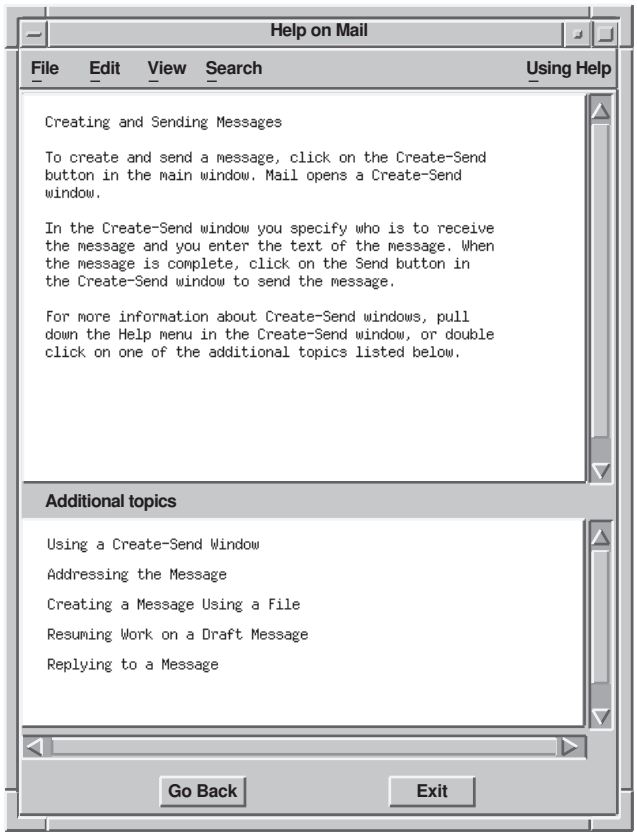
5. When users press a Help push button in a dialog box, have them receive help on the dialog box as a whole, or receive an overview of the dialog box with additional topics about each control. If users press the Help key when input focus is in the dialog box, provide help on the specific control that has the input focus.

6. Decide how to organize each task-oriented topic. Consider the following:

   - Placing task-oriented information first, followed by definitions of terms (if they are necessary), followed by conceptual information or suggestions on how to obtain additional information. Many users of online help are looking for task-oriented information first.

     For example, Figure 5–5 shows a Help topic with task-oriented information first.

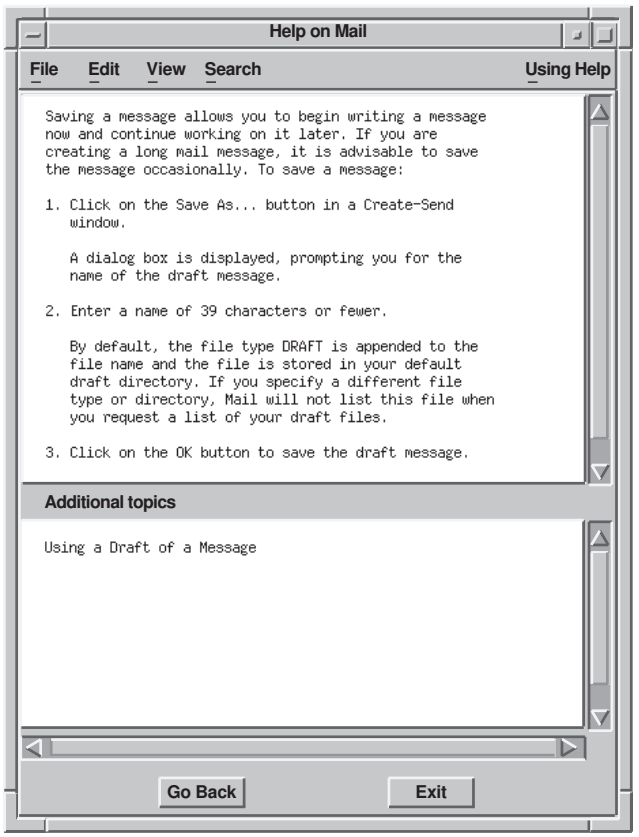
Figure 5–5 Put Task-Oriented Information First



ZK–2364A–GE

- Segmenting task-oriented information into numbered steps if the task requires more than two user actions. Figure 5–6 shows an example of a task-oriented sequential Help topic.

Figure 5–6 Number Steps



ZK–2365A–GE

7. Decide which additional topics to place in the Additional Topics list.

5.2.1.2 Writing Help Topics

Writing Help topics requires accuracy and clarity, and knowledge of users, as does writing hardcopy documentation. The screen is a different medium from the printed page, however, so writing Help topics requires a different type of writing style. Three important guidelines for writing online help are as follows:

- Be brief
- Be clear
- Be visual

Be Brief

Make help text brief so it can fit in the default size of the Help window, and to make the user's job easier. Use the following guidelines to make your Help topics concise:

- Use as few words as possible, but do not omit words such as *the* and *a* for the sake of brevity. The resulting telegraphic style can lead to misinterpretation and mistranslation.
- Avoid a chatty, overly friendly style.
- Use short sentences and short paragraphs. Figure 5–7 compares the use of brief paragraphs with a lengthy one.
- Help the user do a task.

  For example:

```
Use Print...to display the Print Attributes dialog
box to prepare and send your document to a printer.
```

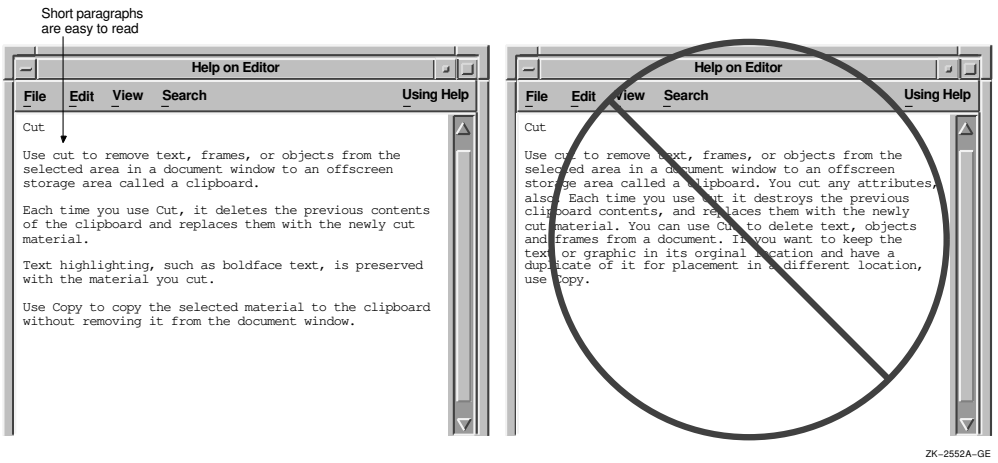
- Write imperative sentences and avoid the passive voice; for example:

```
Use Save to save your document in a file.
```

  Not:

```
The Save menu item is used for saving your document in a file.
```

Figure 5–7 Use Short Sentences and Paragraphs



Note

The art examples in this chapter reflect the Help widget rather than the DECwindows Motif Help System. However, the guidelines can be applied to either.

Be Clear

Use simple, nontechnical language. Write for the beginning user of the application, and avoid jargon, including computer jargon, product jargon, and help jargon. For example, avoid referring to the list of additional topics as a list box. Use the following guidelines to make your Help topics as clear as possible:

- Use simple language.
- Use direct instructions.
- Define all new terms whenever you use them. Define terms in lower-level topics if users can view the topic without viewing a higher-level topic.

  After defining a term, use it. Avoid synonyms.
- Spell out abbreviations, acronyms, and mnemonics and follow with the shortened form in parentheses.

- Use the same terms that are used in the user interface, where possible. For example, refer to the *DECterm window*, not the *terminal emulator window*.

Figure 5–7 shows the first and final drafts of a sample Help topic. The revised Help topic breaks the text into short paragraphs separated by blank lines and reduces the density of the text, thus making it easier to read on the screen. For optimum readability, limit topics to four or fewer short paragraphs whenever possible.

Be Visual

Try to design Help topics that are inviting to the eye and easy to read. Reading information on a screen is more difficult than reading it on a page. Screen resolution may be as much as 20 times lower than the resolution of a printed page; text density that is acceptable on a page is unacceptable on a screen.

You can create a visual format that is easy to read by following these guidelines:

- Break text into short paragraphs separated by blank lines. A high text density with little blank space is hard to read.
- Make the text have a ragged right margin; do not right-justify it. Ragged right margins help readers to keep their place in a section of text.
- Use blank lines between list items, if possible. If the list is short and fits in one Help topic without scrolling, then consider omitting the blank lines.

  Create short list items, if possible.
- Write topics that fit within the default Help window dimensions so users do not need to scroll.

  If your Help topic must exceed the number of lines that your help window displays, you can position the text to give users another cue (in addition to the scroll bar) that more information follows. For example, have the text go to the bottom of the window so that users can see the tops of letters being cut off.

For each Help topic you write, provide a list of additional topics that contain related information.

5.2.2 Remember Translation

It is important to follow the recommendations for writing style given in this chapter in order to assist in the translation of help files.

When a Help topic is translated from an English language version, the amount of text usually increases, so plan for this text expansion.

In addition, to prevent users from having to scroll the help text, translators can request that the default size of the Help window be increased to accommodate the increased amount of text.

5.3 Digital Tools for Creating Online Help

Digital provides two means for creating and displaying online help: the Help widget and the DECwindows Motif Help System.

5.3.1 Using Digital's Help Widget

Digital's Help widget is a special window that users can invoke from the Help menu, the Help command, the Help push button or the Help key. The top pane of the Help window displays the Help topic, that is, information on the object or function on which users request help.

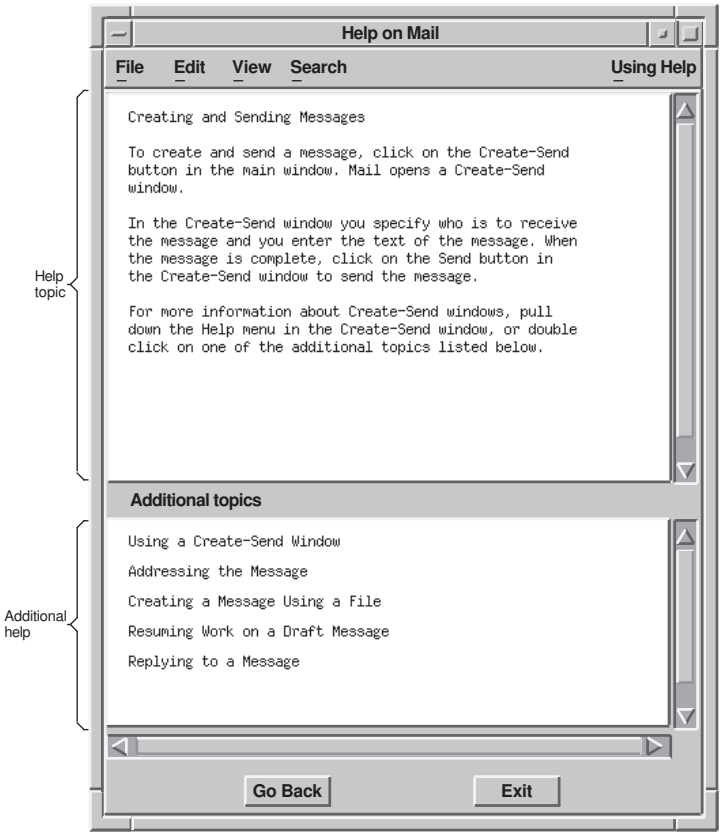
The bottom pane lists additional topics for further help. To select and display an additional topic, users position the pointer over the topic and click MB1.

A Motif Help Window created with Digital's Help widget consists of the following components:

- Title bar with the title centered and a window menu button
- Menu bar
- Help topic
- Additional topics
- Push buttons

The default size provides space for 55 monospaced characters on a line of text, 20 lines of text, and space for 5 subtopics in the Help window. Figure 5–8 shows a sample Help window.

Figure 5–8 Help Window



5.3.1.1 Title Bar

The title bar consists of the help title and the window menu button. The help title consists of the phrase ‘‘Help on’’ and the name of your application, as shown in Figure 5–8.

5.3.1.2 Menu Bar

The menu bar of the Help window contains the following menus:

- File

  The File menu contains the following menu items:

  - Save As..., which invokes a dialog box that allows users to save a copy of the current Help topic in a specified file.
  - Exit, which dismisses the current Help window.

- Edit

  The Edit menu contains the following menu items:

  - Copy, which copies a selected portion of the current topic to the clipboard.
  - Select All, which selects the entire current topic.

- View

  The View menu contains any or all of the following menu items:

  - Go To, which replaces the current topic with the selected additional topic. If no topic is selected in the Additional topics area, disable this menu item.
  - Go Back, which replaces the current topic with the previous topic. The Go Back menu item duplicates the function of the Go back button. If no previous topic exists, disable this menu item.
  - Go to Overview, which replaces the current topic with the application-specific On Window topic. Each application must have an overview topic.
  - Visit, which creates a new Help window that displays the selected additional topic. The previous window displays the current Help topic and retains the input focus. Note that input focus is not passed to the new help window. If no topic is selected in the Additional topics area, disable this menu item.
  - Go to Glossary, which provides an alphabetic list of terms associated with your application.

- Search

  The Search menu contains the following menu items:

  - History..., which invokes a modeless dialog box that lists the titles of all the Help topics viewed during the current help session. The History dialog box contains a list box with the topic titles and Go To, Visit, and Cancel push buttons.
  - Title..., which invokes a modeless dialog box that allows users to display a list of topic titles that contain a keyword or phrase.
  - Keyword..., which invokes a modeless dialog box that allows users to search for Help topics related to specific words.

- Using Help

  The Using Help menu contains an On Window menu item that invokes help on how to use the Help window.

5.3.1.3 Help Topic

A **Help topic** uses text to describe an object, concept, or function on which the user requested help. Have your application display the application-specific On Window topic when users do not request help on a specific topic.

5.3.1.4 Additional Topics

The Additional topics portion of the Help window allows users to get further help on the current Help topic or topics related to the current topic. Involve users in testing to determine how many and what type of additional topics are best.

To select and display an additional topic, users position the pointer over the topic, click MB1, and then choose the Go To menu item from the View menu. Users can also double click on the topic.

5.3.1.5 Push Buttons

The Help window includes two push buttons:

- Go Back, which displays the previous Help topic.
- Exit, which dismisses the current help display.

For more information on using the Help widget, see the *DECwindows Motif Guide to Application Programming* and the *DECwindows Extensions to Motif*.

5.3.2 Using the DECwindows Motif Help System

The DECwindows Motif Help System uses Bookreader to display the online help file. Digital provides three routines for allowing users to access online help files in Bookreader format. These routines are documented in the *DECwindows Motif Guide to Application Programming*. Users can invoke the DECwindows Motif Help System from the Help menu, the Help command, the Help push button or the Help key.

The DECwindows Motif Help System includes the following features:

- Proportional fonts, which are more legible than fixed fonts
- Graphics
- Formatted tables
- Hotspots (A way to move around in the help text by clicking on parts of the text)

For more information on Bookreader, see the *DECwindows Motif Guide to Application Programming*.

You can use any of the following tools to create DECwindows Motif Help System files:

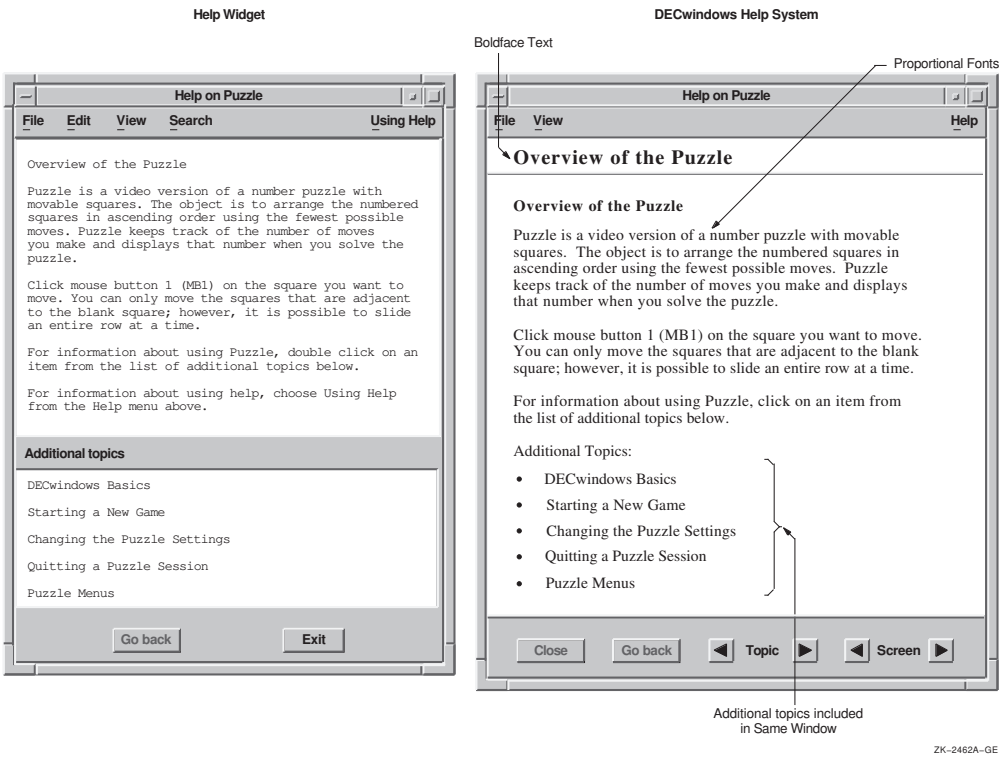
- DECwrite
- VAX DOCUMENT (OpenVMS for VAX only)

Table 5–1 compares the features of the Help Widget to the features of the DECwindows Motif Help System, and Figure 5–9 shows some of the differences in the Help windows.

Table 5–1 Comparison of Help Widget and DECwindows Motif Help System Features

Feature	Help Widget	DECwindows Motif Help System
Text	Small, monospaced font	Multiple fonts, including italics and boldface and multiple point sizes
Performance	Adequate for large help libraries	Better for large help libraries
Graphics	No support	FSE binary file format for graphics
Navigation	Additional topics list	Hotspots
Tables	No support	Supported

Figure 5–9 Comparison of Help Widget and DECwindows Motif Help System Windows



6
Using Digital-Specific Widgets

This chapter describes design and style considerations for the additional widgets that Digital provides with its offering of Motif, or that Digital sells as layered products. The additional widgets that Digital provides with its offering of Motif are the following:

- ColorMix

  This widget is described in Chapter 4.

- Help

  This widget is described in Chapter 5.

- Print

  This widget is described in this chapter.

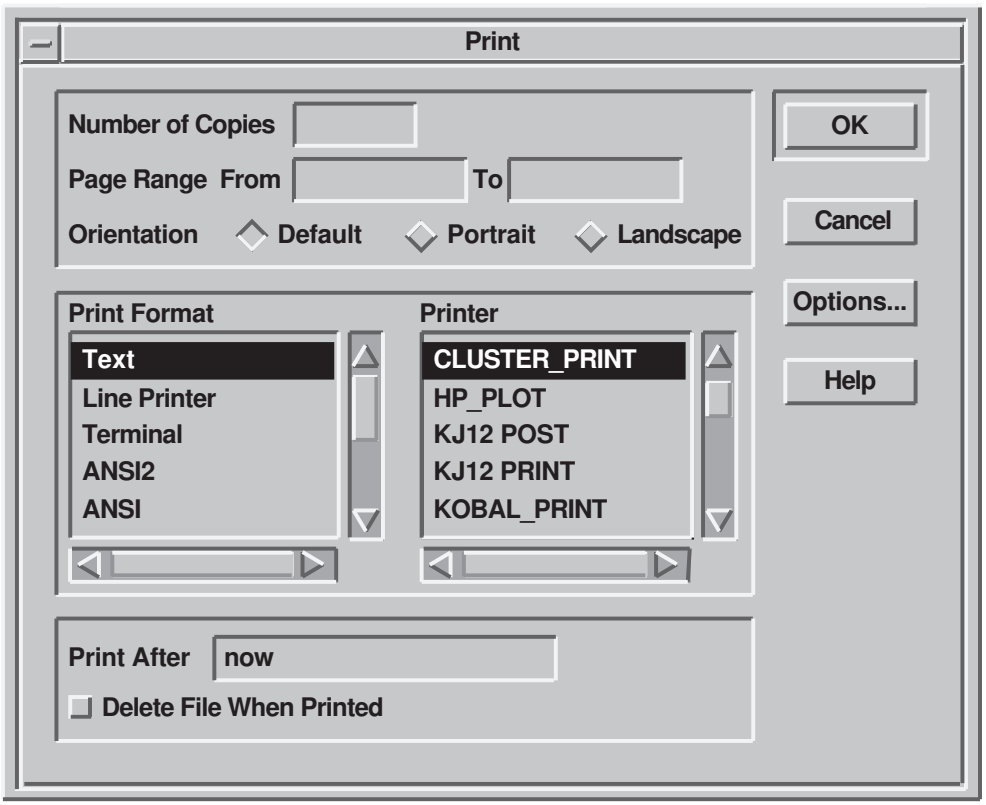
- Structured Visual Navigation (SVN)

  This widget is described in this chapter.

6.1 Using Digital's Print Widget

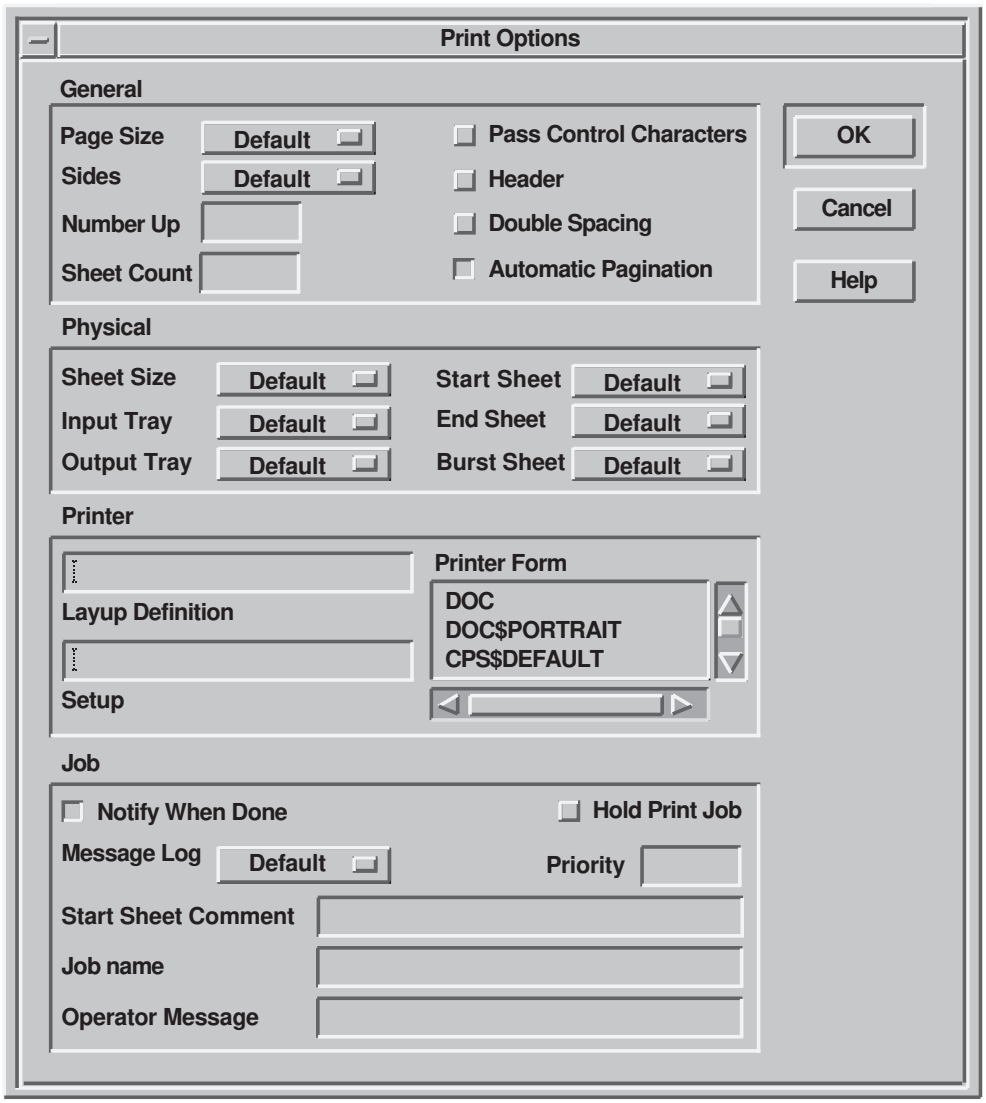
You can use Digital's Print widget in any application that allows users to print files on OpenVMS, UNIX, or Windows NT systems. It consists of a primary dialog box (as shown in Figure 6–1) and a secondary dialog box that is invoked by pressing the Options... push button. The secondary dialog box varies depending upon the operating system; Figure 6–2 shows the secondary dialog box on OpenVMS systems.

Figure 6–1 Print Widget's Primary Dialog Box



ZK–2515A–GE

Figure 6–2 Print Widget's Secondary Dialog Box on OpenVMS Systems



ZK–2527A–GE

In the primary Print dialog box, users can change any of the following settings:

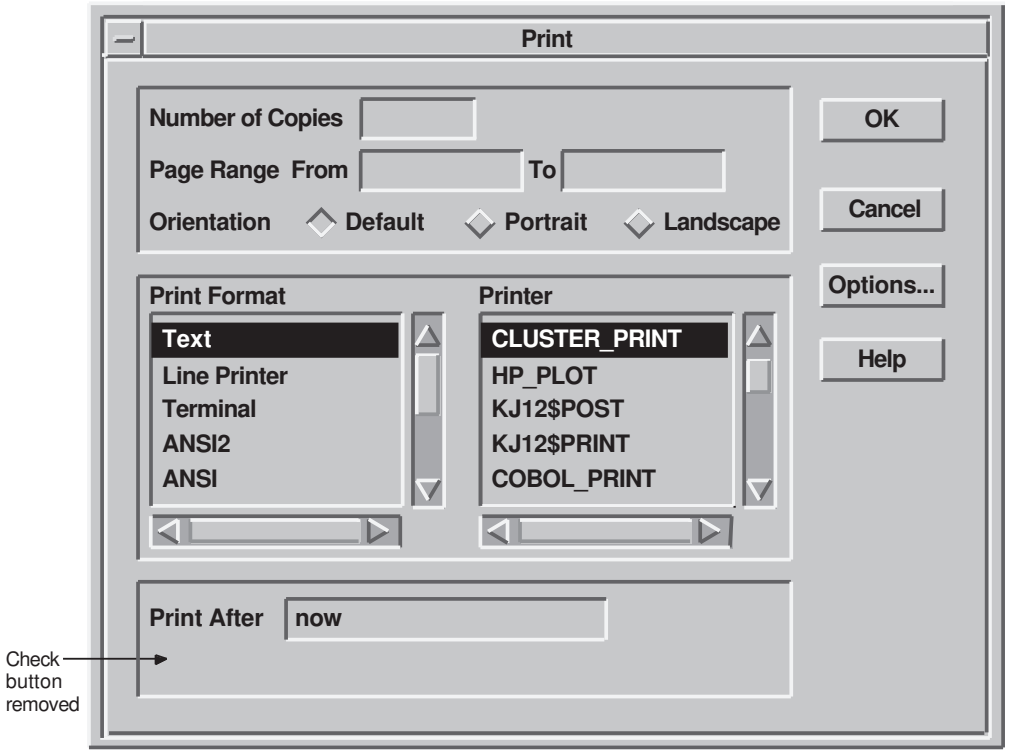
- Number of copies
- Page range
- Orientation
- Print format
- Printer
- Time after which the file prints
- Whether to delete the file after it is printed

The secondary dialog box varies greatly depending on the operating system on which it is being used.

Customizing the Print Widget

You can customize the Print widget dialog boxes to meet the printing needs of your application. For example, in some applications the Delete File When Printed check button is not appropriate. The Print widget allows you to erase such tab groups easily so that they do not appear in the dialog box, as shown in Figure 6–3.

Figure 6–3 Erasing Tab Groups Users Do Not Need



ZK–2759A–GE

Because the Print widget does not provide a mechanism for allowing users to save their print settings, provide a way for them to do so. For example, have the Save Options menu item from the Options menu automatically save print settings, or add a Save Print Options menu item to the Options menu.

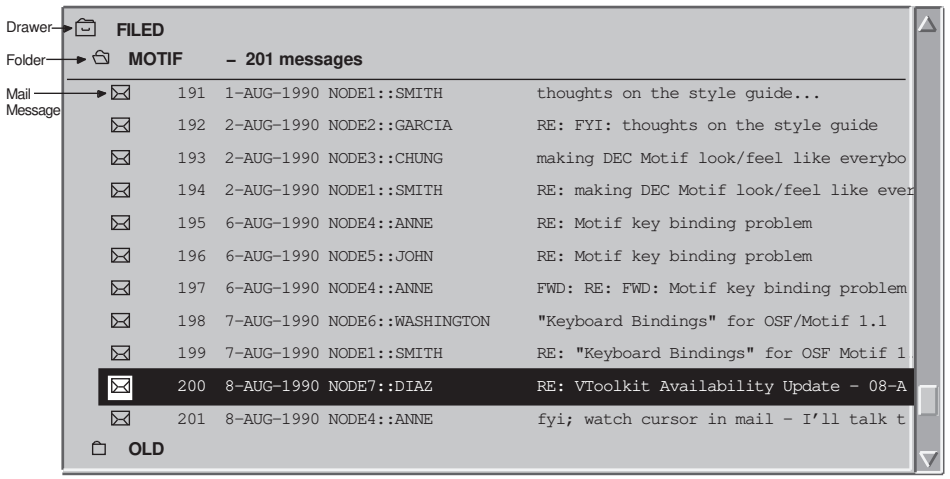
For more information about the Print widget, see the *DECwindows Motif Guide to Application Programming* and the *DECwindows Extensions to Motif*.

6.2 Using the Structured Visual Navigation Widget

You can use the Structured Visual Navigation (SVN) widget to create hierarchies of information in an application; it allows users to access and restructure information in a hierarchical way. You can think of SVN as being able to display levels of information. Users can have it display only the top level of information, or they can expand that level to display the information under the top level.

For example, in OpenVMS DECwindows Mail, users can create drawers that contain folders, and folders that contain messages. In displaying this hierarchy, they can ‘‘open’’ a drawer to display all the folders within it, and ‘‘open’’ a folder to display all the mail messages in the folder, as shown in Figure 6–4.

Figure 6–4 Using the Outline Format to Show SVN Hierarchy



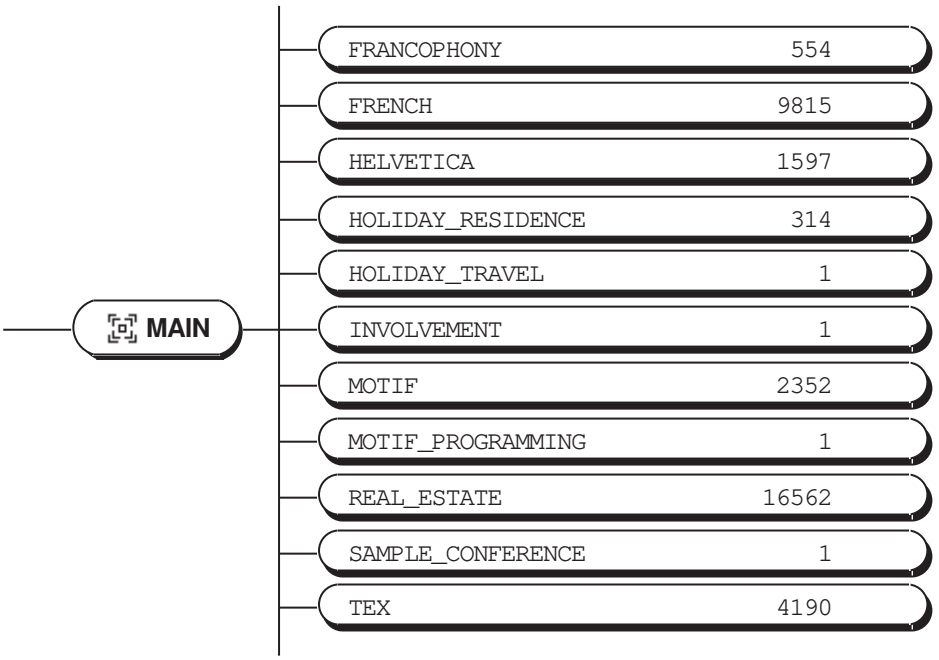
ZK–2517A–GE

You can use SVN to allow users to display hierarchical information in three different formats, or modes:

- Outline format, shown in Figure 6–4
- Tree format, shown in Figure 6–5
- Column format, shown in Figure 6–6.

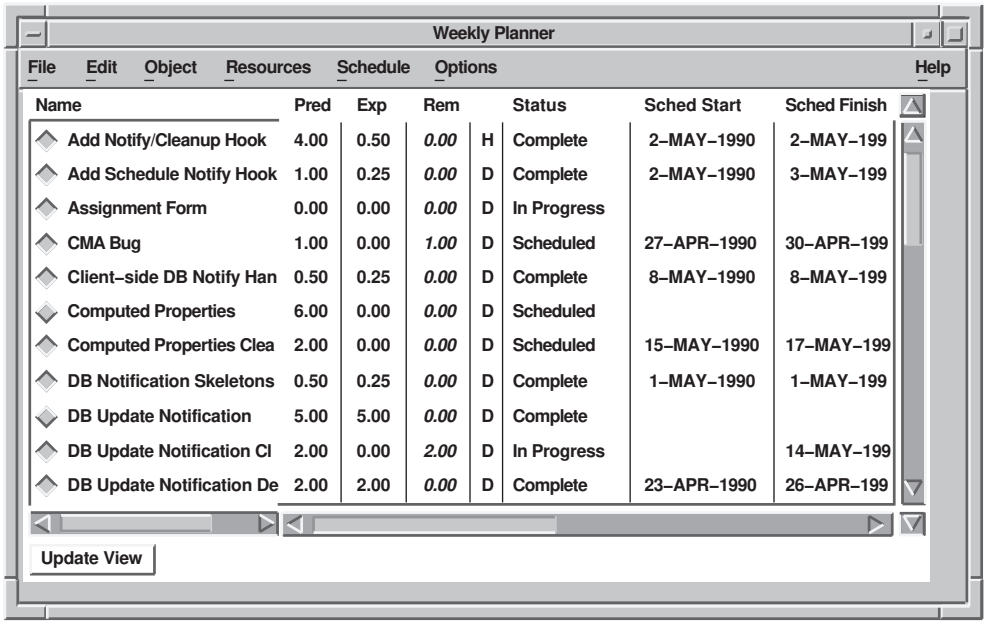
  The difference between the outline and the column format is that in column format, a window pane separates a set of components from the rest of the display. Users can scroll horizontally on each side, independently of the other side. However, there is only one vertical scroll bar.

Figure 6–5 SVN Tree Format



ZK–2519A–GE

Figure 6–6 SVN Column Format



ZK–2518A–GE

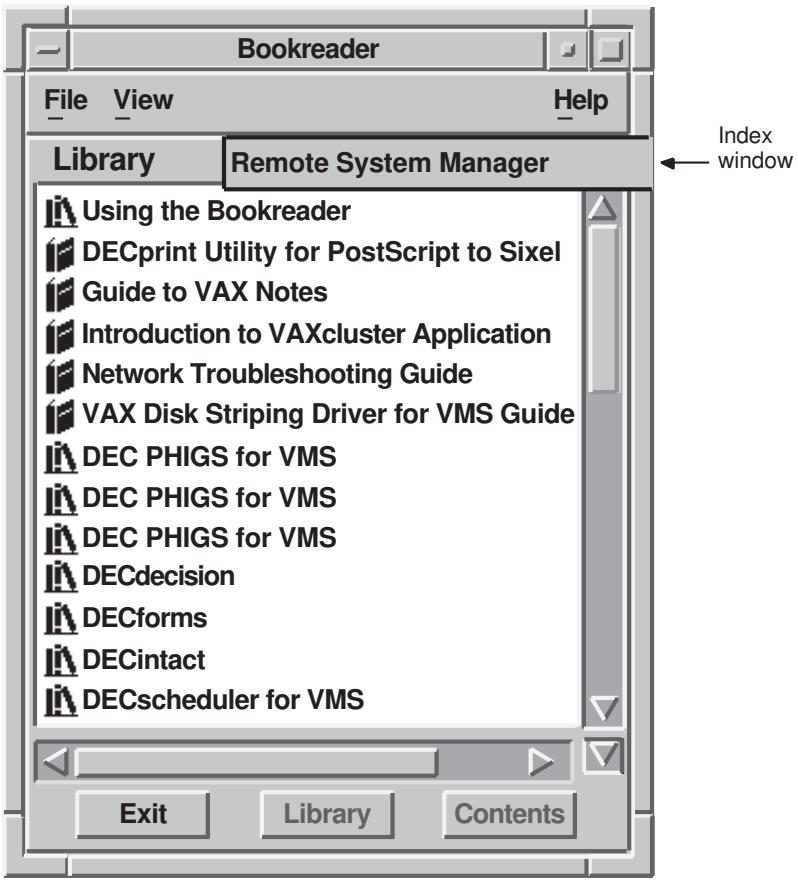
In addition to the three formats, SVN has other features that you can use to create an interface that helps users navigate through hierarchical information. For example, SVN allows you to create outer arrows on a scroll bar (shown in Figure 6–6), and an index window (shown in Figure 6–7).

Outer arrows perform operations that are a magnitude greater than the stepper (inner) arrows. Clicking on the upper stepper arrow moves the display up one unit. Clicking on the upper outer arrow moves the display to the top of the level of hierarchy that users are currently seeing. For example, if a screen displays messages 20 through 40 in a folder that contains 200 mail messages, clicking on the upper outer arrow would cause messages 1 to 20 to be displayed; clicking on the lower outer arrow would cause messages 180 to 200 to be displayed.

The index window is a special window, attached to the scroll bar that offers a guide to the material to be displayed in the window when users release the mouse button. In Figure 6–7, you can see an index window displaying book titles from a Bookreader library. In addition, SVN provides **live scrolling**, sometimes called smooth scrolling. When you enable live scrolling, the display changes dynamically when you move the slider; without live scrolling, when

you move the slider on the scroll bar, the display does not move until after you move the slider.

Figure 6–7 SVN Index Window



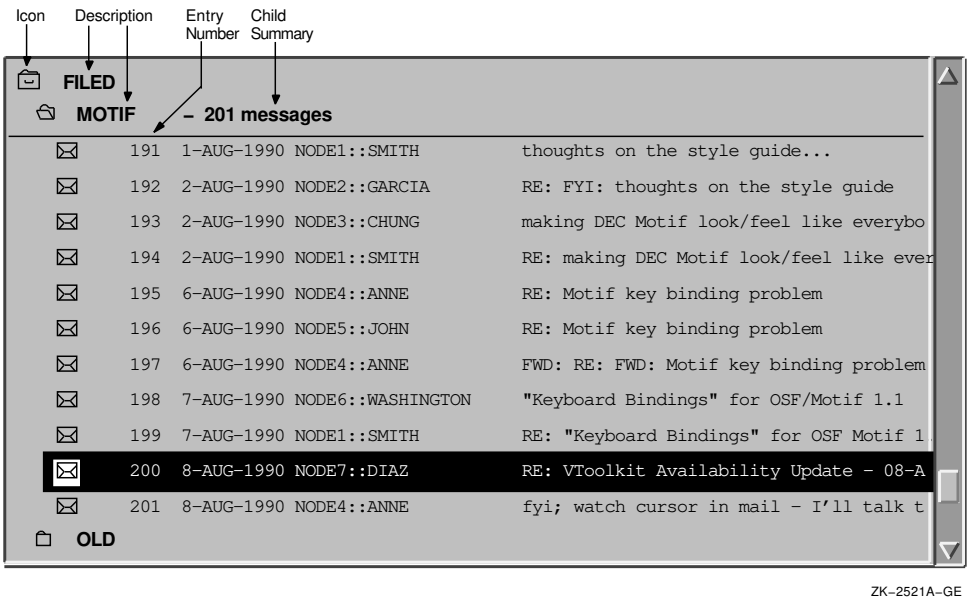
6.2.1 Determining the Components of an Entry

Each SVN entry, or line, in your hierarchy may display as many as 30 components of information. Each component is one of three types:

- Text
- Pixmap (for example, an icon)
- Widget (for example, a push button)

Figure 6–8 illustrates SVN entries using pixmaps (icons) and text.

Figure 6–8 Components in an SVN Entry



ZK–2521A–GE

The highlighted entry (near the bottom of the illustration) has four components:

- A pixmap (An envelope icon that depicts a mail message)
- Text (A text string showing the mail message number)
- Text (Another text string, which is a short line of text that describes whom the mail message is from)
- Text (Yet another text string, which shows the subject line of a mail message)

6.2.2 Designing the Appearance of Your Hierarchy

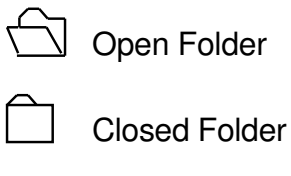
The appearance of the hierarchy greatly influences users' ability to grasp the information it contains. A hierarchy that displays entries in an organized manner is more likely to be understood than one that does not. The following elements help to create a well organized hierarchy:

- The states of icons
- The entry level spacing, that is, the way the entries are indented to show a lower level of hierarchy
- The number and type of fonts used
- The type of selection line presented

6.2.2.1 Showing Icon States

The icon must be able to represent either the expanded or collapsed state of the entry. An expanded entry displays the entries in the level below it. A collapsed entry does not. For example, the folder icon in OpenVMS DECwindows Mail can show a closed folder for an unexpanded entry, or an open folder for an expanded entry, as shown in Figure 6–9.

Figure 6–9 Expanded and Collapsed SVN Icon States



6.2.2.2 Aligning Entries

Align the children of an entry along a common line. This gives the hierarchy an even, ordered appearance and enables users to distinguish one level of hierarchy from another. For example, the hierarchy of mail messages to folders to drawers is clear because mail messages are indented under folders, and folders are indented under drawers, as shown in Figure 6–4.

6.2.2.3 Using Fonts Within a Hierarchy

The size and type of font style you use throughout your SVN hierarchy will determine how quickly and efficiently users perceive information. Although this choice is highly subjective, there are a few guidelines to follow when you design the appearance of the hierarchy.

- Limit the Number of Fonts

  Limit the number of different fonts and font sizes contained within a hierarchy. Use one font and one font size throughout the entire hierarchy, while altering the style of the font on different levels within the hierarchy.

  For example, use 14 point Helvetica with bold letters for the parent entries on one level of the hierarchy and 14 point Helvetica with plain letters for their children. This enables users to distinguish between the parent and child entries.

  Displays that contain many different fonts, font styles, and font sizes are confusing to users because they constantly draw their eyes from one element to another.

- Choose a Simple Font

  Choose a font that is easy to read on the screen. A commonly used style like Helvetica is a good choice because the characters are simple and clear. Be sure the characters are large enough to be seen. Characters displayed on a screen need to be larger than those in print to be clear. A good character size is 14 points. Avoid using characters that are smaller than 12 points.

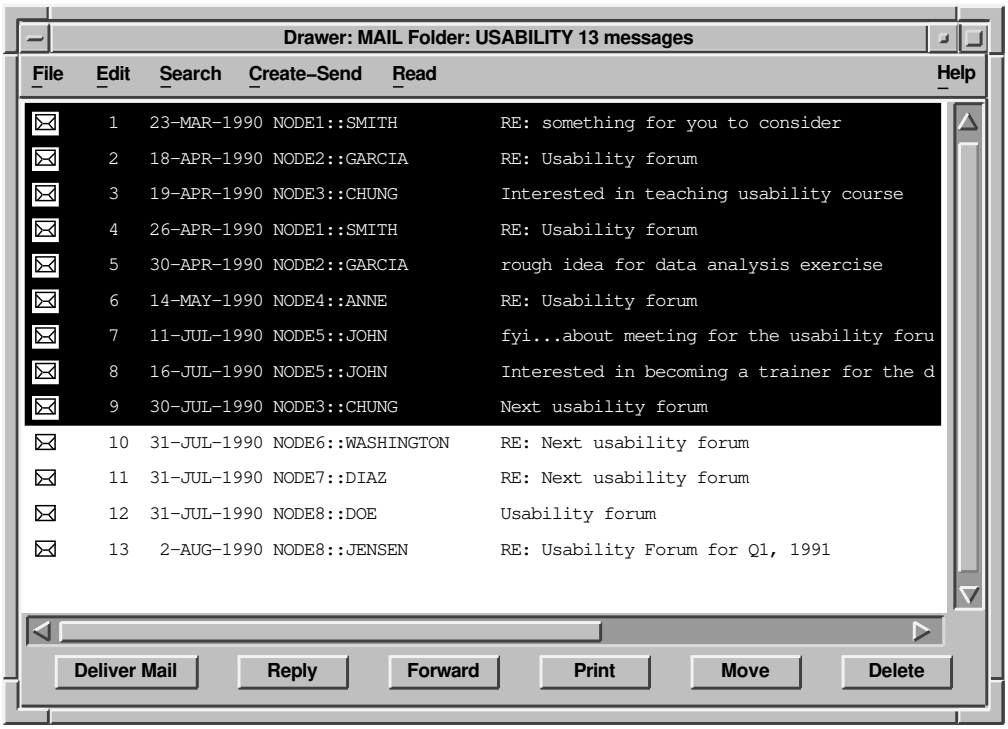
- Using Additional Fonts and Font Sizes

  Some applications may require you to use several fonts or sizes of fonts to present your material clearly. If this is the case, organize your entries carefully, using the same type of font and font size for all entries on the same level of hierarchy. Preliminary tests indicate users prefer a gradient scale of font sizes, with the larger sizes at the top of the hierarchy and the smaller ones at the bottom.

6.2.2.4 Choosing Selection Modes

There are two selection modes in SVN: full and limited selection. In full selection line mode, when users select any component in an entry line, the entire entry is selected; that is, every component of the entry is automatically selected. An example of full selection is shown in Figure 6–10.

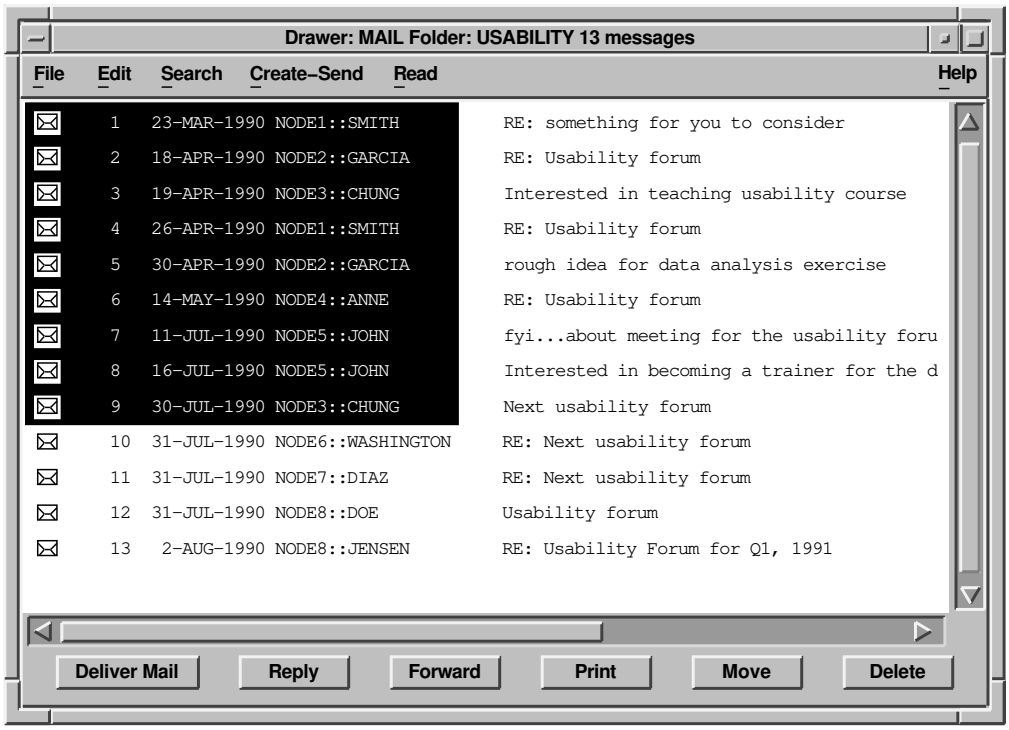
Figure 6–10 SVN Selection with a Full Selection Line



ZK–2523A–GE

In limited selection line mode, users can select individual components within the entry, as shown in Figure 6–11. SVN does not support both selection modes within one application, so choose the selection mode that best suits the users' needs.

Figure 6–11 SVN Selection with a Limited Selection Line

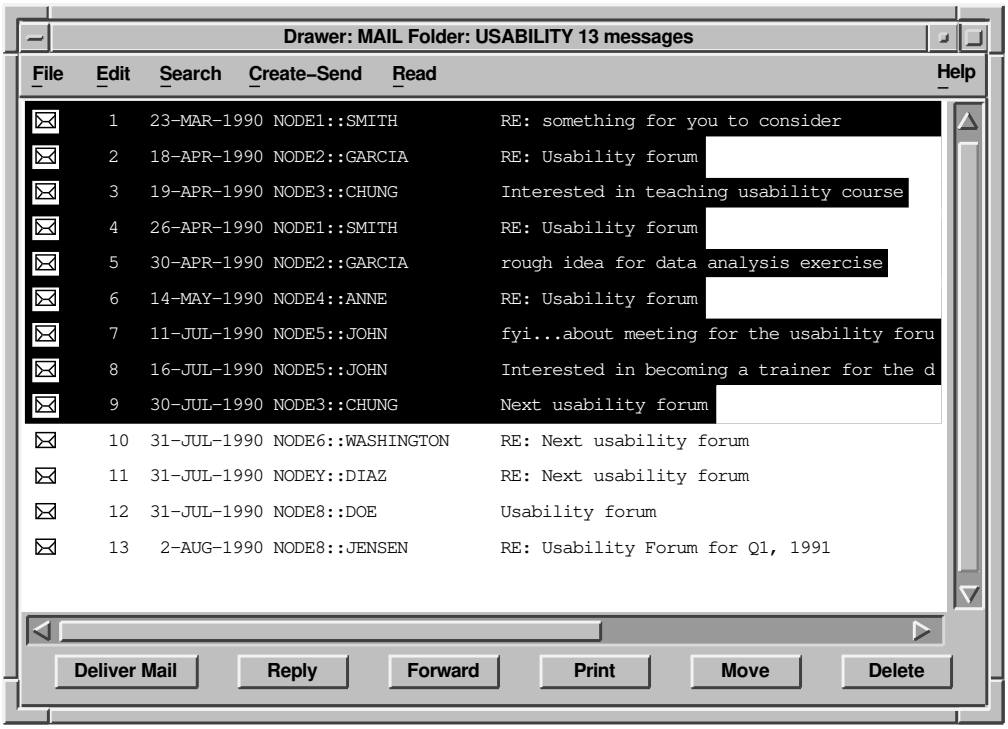


ZK–3338A–GE

6.2.2.5 Choosing Selection Line Length

Within each selection mode are two selection line lengths that you can choose: fixed and variable. With a fixed-length selection line, the entire length of the entry appears selected, as shown in Figure 6–10. However, with a variable-length selection line, only the part of the component field that displays user information appears selected, as shown in Figure 6–12.

Figure 6–12 SVN with a Variable-Length Selection Line



ZK–2524A–GE

6.2.3 Providing Items in the View Menu

When you use SVN in any window except a dialog box, allow users to quickly expand and collapse multiple entries by providing the following items in the View menu:

- E xpand

  Expands the currently selected item one level deeper.

- Expand F ull

  Expands the selected items to their full depth.

- Expand All

  Expands all items to their full depth.

- C ollapse

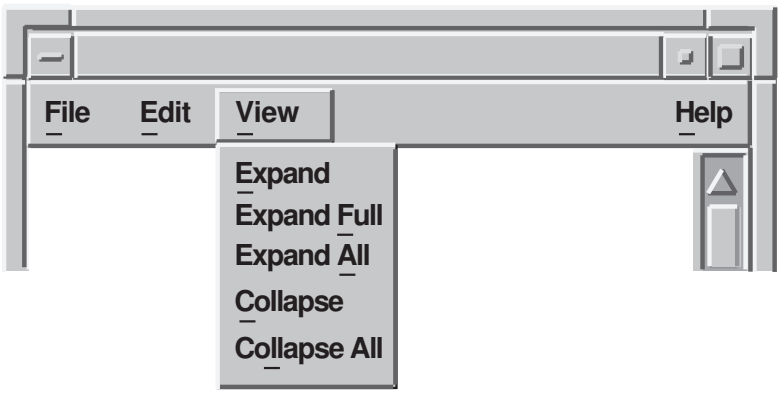
  Contracts the selected items.

- Col lapse All

  Contracts all items.

Figure 6–13 shows an example arrangement of these items in the View menu.

Figure 6–13 SVN Menu Items in the View Menu



ZK–2525A–GE

For more information on creating interfaces with SVN, see the *DECwindows Motif Guide to Application Programming* and the *DECwindows Extensions to Motif*.

A

Keyboard and Mouse Bindings for DECwindows Motif

This appendix provides a table of the keyboard bindings for DECwindows Motif, a table of the mouse bindings for Digital's three-button mouse, and a table of the standard accelerators (keyboard bindings for menu items) for the Window menu and the Edit menu.

Menus are assumed to be vertical; menu bars, horizontal.

Examples are intended to illustrate the most common uses of the keys. The examples are not exhaustive.

Motif uses a plus sign (+) to show key combinations (for example, Alt+F5). Digital traditionally uses a slash (/) for key combinations. This document uses plus signs throughout these tables.

A.1 Keyboard Bindings

The bindings described in the tables of this appendix pertain to Digital LK401 keyboards and Digital PC-style keyboards (including the LK443 and LK444 models). DECwindows applications can also use the Digital LK201 keyboard, which is almost identical to the LK401 model. Table A–1 lists the differences between the LK201 and the LK401 keyboard.

Table A–1 Differences Between LK201 and LK401 Keyboards

LK201	LK401
Compose Character Key	Two Alt keys and two Compose Character keys, one of each on each side of the space bar
Prev Screen	Prev
Next Screen	Next

OSF/Motif supports bindings for a number of other keyboards. For information about bindings for these other keyboards, see the *OSF/Motif Style Guide*.

The general concepts of the keyboard bindings are as follows:

- Motif keyboard binding names that begin with M are modifiers, for example, MShift. A modifier is a key that, when pressed with another key, changes the meaning of the other key.
- Motif keyboard binding names that begin with K are keys, for example, KTab.

Table A–2 lists the modifiers for the DECwindows Motif bindings.

Table A–2 Modifiers for DECwindows Motif Keyboard Bindings

Motif Name	Keyboard Equivalent	Usage Examples
MShift	Shift	A modifier key. When used with KUp, KDown, KLeft, KRight, KBeginLine, KEndLine, KBeginData, and KEndData will perform the navigation operation and extend the selection.
MCtrl	Ctrl	A modifier key. Keyboard bindings that use the MCtrl modifier (the Ctrl key) are usually higher-order actions than the same key binding without the MCtrl modifier. For example, to go the beginning of a line, you press Alt+ ←, while to go to the beginning of the file, you press Ctrl+Alt+←.
MAlt	Alt	A modifier key.

Table A–3 lists the general keyboard bindings for DECwindows Motif.

Table A–3 General Bindings

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KEnter	Return	Return	Inserts a carriage return in a multiline text-entry field. Also used in activation. See KActivate.
KHelp	Help	F1	Invokes context-sensitive help for the control that contains the location cursor.
KCancel	F11	Escape	Cancels mouse-drag operations such as moving a window, resizing a window, or using a menu. Cancels menus. Cancels dialog boxes.
KTab	Tab	Tab	Inserts a tab character in a multiline text-entry field. Also used in field navigation. See KNextField, KPrevField.
KBackTab	Shift+Tab	Shift+Tab	Used in field navigation. See KPrevField.
KSpace	Space bar	Space bar	Inserts a blank space in text-entry fields. Also used in selection. See KSelect.
KBackSpace	<X	Backspace	In text, if any characters are selected deletes them. Otherwise, deletes the character preceding the cursor. Also used in undo. See KUndo.

(continued on next page)

Table A–3 (Cont.) General Bindings

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KUp	↑	↑	Moves up in menus, multiline text controls, list boxes, and other scrollable controls. Moves up between controls in a tab group.
KDown	↓	↓	Moves down in menus, multiline text controls, list boxes, and other scrollable controls. Moves down between controls in a tab group.
KLeft	←	←	Moves left in text controls and other scrollable controls. Moves left between controls in a tab group. Also used in menu navigation.
KRight	→	→	Moves right in text controls and other scrollable controls. Moves right between controls in a tab group. Also used in menu navigation. In pull-down menus, acts as KPrevMenu. See KNextMenu.
KPageUp	Prev	PageUp	Scrolls up in a multiline text field, list box, or other scrollable control.
KPageDown	Next	PageDown	Scrolls down in a multiline text field, list box, or other scrollable control.

(continued on next page)

Table A–3 (Cont.) General Bindings

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KDelete	Remove	Delete	In text, if any characters are selected, deletes them. Otherwise, deletes the character following the cursor. In other controls, deletes the selected items.

Table A–4 lists the keyboard bindings for general navigation in DECwindows Motif.

Table A–4 Bindings for General Navigation

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KBeginLine	Alt+ ←	Home	Moves to beginning of line in a text field. In a list box or other scrollable control, moves to first item.
KEndLine	Alt+ →	End	Moves to end of line in a text field. In a list box or other scrollable control, moves to last item.
KBeginData	Ctrl+Alt+←	Ctrl+Home	Moves to beginning of a text field, list box, or other scrollable control.
KEndData	Ctrl+Alt+→	Ctrl+End	Moves to end of a text field, list box, or other scrollable control.
KPageLeft	Ctrl+Prev	Ctrl+PageUp	Scrolls left in a text control, list box, or other scrollable control.
KPageRight	Ctrl+Next	Ctrl+PageDown	Scrolls right in a text control, list box, or other scrollable control.

(continued on next page)

Table A–4 (Cont.) Bindings for General Navigation

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KNextField	Tab Ctrl+Tab	Tab Ctrl+Tab	Moves location cursor to the next basic control or tab group in a dialog box. Moves the location cursor from pane to sash to next pane in a paned window.
KPrevField	Shift+Tab	Shift+Tab	Moves location cursor to the previous basic control or tab group in a dialog box. Moves the location cursor from pane to sash to previous pane in a paned window (reverse order of Tab key).

Table A–5 lists the keyboard bindings for navigating menus in DECwindows Motif.

Table A–5 Bindings for Menu Navigation

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KMenuBar	F10	F10	Moves location cursor to the menu bar. If the location cursor is already in the menu bar or in a pull-down menu, moves it back to the client area.
KMenu	Shift+F10	Shift+F10	Displays a pop-up menu. If the location cursor is already in a pull-down menu, cancels the menu.

Table A–6 lists the keyboard bindings for navigating text in DECwindows Motif.

Table A–6 Bindings for Text Navigation

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KPrevWord	Ctrl+ ←	Ctrl+←	Moves to the previous word in a text field.
KNextWord	Ctrl+ →	Ctrl+→	Moves to the next word in a text field.
KPrevPara	Ctrl+ ↑	Ctrl+↑	Moves to the previous paragraph in a multiline text field. An optional function.
KNextPara	Ctrl+ ↓	Ctrl+↓	Moves to the next paragraph in a multiline text field. An optional function.
KEraseEndLine	Ctrl+Remove	Ctrl+Delete	Erases to the end of the line. An optional function.

Table A–7 lists the keyboard bindings for navigating windows in DECwindows Motif.

Table A–7 Bindings for Window Navigation

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KWindowMenu	Shift+F11	Shift+F11	Posts the window menu.
KNextWindow	Alt+Tab	Alt+Tab	Moves to the next window that is not part of the current window family.
KPrevWindow	Shift+Alt+Tab	Shift+Alt+Tab	Moves to the previous window that is not part of the current window family.
KNextFamilyWindow	Alt+F6	Alt+F6	Moves to next window in window family.
KPrevFamilyWindow	Shift+Alt+F6	Shift+Alt+F6	Moves to previous window in window family.

Table A–8 lists the keyboard bindings for activating screen objects in DECwindows Motif.

Table A–8 Bindings for Activation and Editing

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KActivate	Return Ctrl+Return Enter	Enter	Terminates a move or resize of a window. In menus (pull-down, pop-up, and option), activates (chooses) a menu item. In dialog boxes, when location cursor is not in the push-button field or in an option menu, activates the default push button. In dialog boxes, when location cursor is in the push-button field, activates the push button containing the location cursor. When the location cursor is in a multiline text-entry field, Return will insert a carriage return. Ctrl+Return must be used to activate the default push button.
KUndo	Alt+	Alt+Backspace	An optional function. Undoes the previous operation.

(continued on next page)

Table A–8 (Cont.) Bindings for Activation and Editing

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KSelect	Select Space bar	Space bar	Activates push buttons, check buttons (toggles them on or off), and option menus. Activates (turns on) a radio button that is off. If all the buttons in a set of radio buttons can be off, turns off a radio button that is on. If one of the buttons in a set of radio buttons must always be on, does not turn radio buttons off. Can be used to select an item in a list box. KSelect followed by a shifted navigation operation initiates range selection; for text-entry fields, Ctrl+Space bar must be used.
KExtend	Shift+Ctrl+Space bar Shift+Select	Shift+Ctrl+Space bar	In Add Mode, a user can type KSelect, navigate to a different location, and then type KExtend to select all of the items in the extended range.
KReselect	Ctrl+Shift+Space bar	Ctrl+Shift+Space bar	An optional function. Restores the previous selection.
KSelectAll	Ctrl+Slash (/)	Ctral+Slash (/)	Selects all of the elements in a control (for example, a text-entry field, list box, graphics canvas).

(continued on next page)

Table A–8 (Cont.) Bindings for Activation and Editing

Motif Name	LK401 Equivalent	PC-style Equivalent	Use
KDeselectAll	Ctrl+Backslash (\)	Ctrl_Backslash (\)	Deselects all of the elements in a control (for example, a text-entry field, list box, graphics canvas).
KCut	Remove Shift+Delete	Shift+Delete	Cuts selected object to the clipboard.
KCopy	Ctrl+Insert Here	Ctrl+Insert	Copies selected object to the clipboard.
KPaste	Shift+Insert Here	Shift+Insert	Pastes from the clipboard.
KAddMode	Shift+F8	Shift+F8	Toggles between Normal Mode and Add Mode. Normal Mode is the default state, and allows only range selection from the keyboard. Add Mode allows keyboard users to make discontiguous selections.
KRestore	Ctrl+Shift+Insert Here	Ctrl+Shift+Insert	Restores text to value prior to editing.

A.2 Mouse Bindings

The bindings described in Table A–9 pertain to Digital's three-button mouse. Motif also supports bindings for a one-button mouse and a two-button mouse.

Table A–9 Mouse Bindings for DECwindows Motif (Three-Button Mouse)

Motif Name	Mouse Binding	Use
BSelect	MB1	Used to set the location cursor, and for selection and activation.
BDrag	MB2	Used for moving, copying, and dragging objects.
BCustom	MB3	Used for additional, application-defined interactions. Applications are encouraged to use MB3 for pop-up menus. See BMenu.
BMenu	MB3	Used for pop-up menus.
BExtend	Shift+MB1	Extends a selected range.
BToggle	Ctrl+MB1	Toggles between items in the range.
BPrimaryPaste	MB2	Cuts or copies the current selection to the pointer. Operation depends on object being cut or copied and upon the destination.
BPrimaryCopy	Ctrl+MB2	Copies the selection to the pointer.
BPrimaryCut	Alt+MB2	Cuts (moves) the selection to the pointer.
BQuickPaste	MB2	Cuts or copies the secondary selection to the destination cursor.
BQuickCopy	Ctrl+Shift+MB2	Copies a secondary selection to the destination cursor.
BQuickCut	Alt+Shift+MB2	Cuts a secondary selection to the destination cursor.

A.3 Standard Accelerators

Accelerators are keyboard bindings that have equivalent functions in menus. You can find all of the standard accelerators in either the Window menu or the Edit menu.

Table A–10 lists the OSF/Motif standard accelerators for the Window Menu. These accelerators are also listed in the *OSF/Motif Style Guide*.

Table A–10 Motif Accelerators for the Window Menu

Window Menu Function	Accelerator
Restore	Alt+F5
Move	Alt+F7
Size	Alt+F8
Minimize	Alt+F9
Maximize	Alt+F10
Lower	Alt+F3
Close	Alt+F4

Table A–11 lists the accelerators that Digital suggests you use with the File menu. The accelerators for the Edit menu are defined by Motif while the accelerators for the File menu are suggestions by Digital.

Table A–11 Suggested Accelerators for the File Menu

File Menu Function	Accelerator
New	Ctrl+N
Open	Ctrl+O
Save	Ctrl+S
Print...	Ctrl+P
Close	Ctrl+C

Table A–12 lists the OSF/Motif standard accelerators for the Edit menu. These accelerators are also listed in the *OSF/Motif Style Guide*.

Table A–12 Motif Accelerators for the Edit Menu

Edit Menu Function	LK401 Accelerator	PC-style Accelerator
Undo	Alt+<X	Alt+Backspace
Cut	Shift+Remove	Shift+Delete
Copy	Ctrl+Insert Here	Ctrl+Insert
Paste	Shift+Insert Here	Shift+Insert

B

Common Motif Terms Translated into European Languages

This appendix contains common OSF/Motif terms in English with corresponding translations into ten European languages. These terms apply to screen objects (such as the standard menus) and to terms that writers and translators can use when creating user information. By using the translated terms in these tables for labeling screen objects in all Motif applications, and for referring to them in all documentation, you contribute to the consistency of all Motif user interfaces.

The tables include the following languages:

- Danish
- Dutch
- Finnish
- French
- German
- Italian
- Norwegian
- Portuguese
- Spanish
- Swedish

B.1 Danish and Dutch Terms

English	Danish	Dutch
Accelerator	Accelerator	Toetscombinatie
Acknowledge	Læse	Gezien
Activate	Aktivere	Activeren
Active	Aktiv	Actief
Add	Tilføj	Voeg toe
Alt key	Alt tasten	Toets [Alt]
Apply	Anvend	Voer uit
Arrow keys	Piltaster	Pijltoetsen
Background	Baggrund	Achtergrond
Block cursor	Blokmarkør	Blokcursor
Border	Ydre ramme	Rand
Button	Knap	knop
Cancel	Annullér	knop [Annuleer]
Capture	Skrive i fil	Bewaren
Check button	Til/fra knap	Aan/uit-knop
Choose	Vælge	Kiezen
Clear	Fjern	Verwijder
Click	Klikke	Klikken
Clipboard	Mellemlager	Prikbord
Close	Lukke	Sluit
Compose sequence	Tastsekvens	Reeks van toetsaanslagen
Continue	Fortsæt	Verder
Control panel	Kontrolpanel	Keuzepaneel
Copy	Kopiere	Kopieer
Cursor	Markør	Cursor
Customize	Tilpasse	Instellingen
Cut	Slette	Knip
Deactivate	Deaktivere	Deactiveer
Dialog box	Dialogboks	Kader
Dimmed	(Skrevet med) svag skrift	Grijs

English	Danish	Dutch
Dismiss	Afbryd	Sluit kader
Display	Vise	Tonen
Double click	Dobbeltklikke	Tweemaal klikken
Drag	Trække	Verschuiven
Edit	Redigere	Bewerk
Enable	Slået til	Inschakelen
End session	Afslutte session	Beëindig sessie
Enter	Indlæs	Toets [Enter]
Exit	Slut	Einde
Expand	Ekspandere	Expanderen
Extract	Overføre til fil	Bewaar
Fields	Felter	Velden
File	Fil	Bestand
Filter	Filter	Bestandsfilter
Find	Søg	Zoek
Foreground	Forgrund	Voorgrond
Go Back	Tilbage	Ga terug
Go To	Gå til	Ga naar
Help	Hjælp	Help
Highlight	Fremhæve	Benadrukken
Hold	Holde	Vast
Hotspot	Reference	Klikkader
Icon	Ikon	Pictogram
Include	Inkludere	Neem op
Index	Stikordsregister	Index
Initial state	Starte som:	Start op als
Input focus	aktivt vindue	actief
Interface	Brugergrænseflade	Interface
Login screen	Indlogningsbillede	Login-scherm
Lower	Flytte bagest	Onderste
Main window	Hovedvindue	Hoofdvenster

English	Danish	Dutch
Maximize	Maksimere	Maximaliseren
Maximize button	Knappen Maksimér	Vensterknop
Menu	Menu	Menu
Menu bar	Menulinie	Menubalk
Menu item	Menuvalg	Optie
Merge	Samordne	Voeg samen
Minimize	Minimere	Minimaliseren
Minimize button	Knappen Minimér	Pictogramknop
Mnemonic	Mnemonico	Afkorting van optie
Mouse	Mus	Muis
Mouse button	Museknap	Muisknop
Move	Flytte	Verplaats
New	Ny	Nieuw
On Context	Kontekst	Context
On Keys	Taster	Toetsen
On Window	Vindue	Venster
Open	Åbne	Open
Options	valgmenu	Opties
Pane	Vinduesramme	omschrijven
Password	Kodeord	Wachtwoord
Paste	Indsætte	Plak
Pause	Pause	Onderbreken
Point	Pege	Aanwijzen
Pointer	Pegemarkør	Aanwijzer
Pop-up menu	Pop-op menu	Contextmenu
Post	Fastholde	Inactief maken
Press	Trykke	Drukken
Previous	Forrige	Vorige
Prompt	Klartegn	Vraag
Pull-down menu	Rullegardinsmenu	(Rol)menu
Quit	Annullér	Stop

English	Danish	Dutch
Release	Frigive	Vrijgeven
Replace	Erstatte	Vervang
Reset	Reset	Beginwaarde
Resize	Ændre størrelse på	Venstergrootte aanpassen
Resize border	Standardstørrelse	Pas kadergrootte aan
Restore	Genoprette	Wijzigingen ongedaan maken
Return	Retur	Toets [Return]
Sash	Vindueshåndtag	Blokje voor venstergrootte
Save	Arkivere	Bewaar
Save as	Arkivere som	Bewaar als
Scroll	Rulle	Scrollen
Scroll bar	Rullebane	Scroll-balk
Search	Søg	Zoek
Select	Vælge	Selecteren
Send	Afsende	Verzenden
Session	Session	Sessie
Shift click	Skift-klikke	Omschrijven
Show	Vise	Tonen
Shrink	Formindske	Verkleinen
Shuffle down	Flytte bagest	Venster onderop
Shuffle up	Flytte forrest	Venster bovenop
Size	Størrelse	Grootte
Slider	Rullefelt	Schuif
Start	Starte	Beginnen
Stepping arrow	Rullepile	Scroll-pijl
Stop	Stop	Stop
Tab key	Tabulator	Toets [Tab]
Text entry field	Feltet Tekst	Tekstinvoerveld
Title bar	Titellinien	Titelbalk
Toggle button	Skifteknap	Aan/uit-knop
Topic	Emne	Onderwerp

English	Danish	Dutch
Trough	Rulleområde	Sleuf van scroll-balk
Undo	Fortryde	Maak ongedaan
Update	Opdatér	Werk bij
View	Se på	Raadpfleeg/Overzicht
Window	Vindue	Venster
Work area	Arbejdsområde	Werkgebied
Workspace	Arbejdsområde	Werkgebied

B.2 Finnish and French Terms

English	Finnish	French
Accelerator	Pikavalinta	Raccourci
Acknowledge	Vahvistaa	Reçu
Activate	Käynnistää	Activer
Active	Käynnissä	Active
Add	Lisää	Ajouter
Alt key	Alt-näppäin	Touche Alt
Apply	Toteuta	Appliquer
Arrow keys	Nuolinäppäimet	Touches de déplacement
Background	Tausta	Arrière-plan
Block cursor	Kohdistimena nelikulmio	Curseur Bloc
Border	Reunat vai Ulkokehys	Bordure
Button	Painike	Bouton
Cancel	Peruuta	Annuler
Capture	Kopioida	Capturer
Check button	Vaihtopainike	Bouton de sélection
Choose	Valita	Sélectionner
Clear	Poista alue	Effacer
Click	Painaltaa	Cliquer
Clipboard	Leikesäilö	Presse-papiers
Close	Sulje	Fermer
Compose sequence	Merkkiyhdelmänäppäilyt	Séquence de composition
Continue	Jatka	Continuer
Control panel	Ohjauspaneeli	Panneau de contrôle
Copy	Kopioi	Copier
Cursor	Kohdistin	Curseur
Customize	Muuta asetuksia	Personnaliser
Cut	Leikkaa	Couper
Deactivate	Passivoida	Désactiver
Dialog box	Valintaikkuna	Boîte de dialogue
Dimmed	Himmeä	Grisé

English	Finnish	French
Dismiss	Poista näytöstä	Abandonner
Display	Näyttö	Afficher
Double click	Kaksoispainallus	Cliquer deux fois
Drag	Kuljettaa	Traîner
Edit	Muokkaa (muokata)	Éditer
Enable	Käyttöön	Activée
End session	Lopeta istunto	Quitter
Enter	Enter	Valider
Exit	Poistu	Sortir
Expand	Laajenna	Agrandir
Extract	Talleta	Créer fichier
Fields	Kentät	Champs
File	Tiedosto	Fichier
Filter	Näytä	Filtre
Find	Etsi	Rechercher
Foreground	Edusta	Premier plan
Go Back	Palaa	Revenir
Go To	Siirry	Ouvrir
Help	Opastus	Aide
Highlight	Korosta	Mettre en valeur
Hold	Pidä	Figer
Hotspot	Pikaviite	Renvoi
Icon	Kuvake	Icône
Include	Liitä	Inclure
Index	Hakemisto	Index
Initial state	Aloitustila	État au démarrage
Input focus	Käytössä oleva (ikkuna)	Verbaliser
Interface	Liittymä	Interface
Login screen	Sisäänkirjoittautumisnäyttö	Écran d'accueil
Lower	Alle	Inférieur
Main window	Pääikkuna	Fenêtre principale

English	Finnish	French
Maximize	Koko näytöksi	Agrandissement
Maximize button	Kasvattamispainike	Bouton d'agrandissement
Menu	Valikko	Menu de fenêtre
Menu bar	Valikkorivi	Barre de menus
Menu item	Valikon vaihtoehto	Option de menu
Merge	Yhdistä	Fusionner
Minimize	Kuvakkeeksi	Réduction
Minimize button	Kuvakepainike	Bouton de réduction
Mnemonic	Toimintolyhenne	Caractère mnémonique
Mouse	Hiiri	Souris
Mouse button	Hiiren painike	Bouton de la souris
Move	Siirtää	Réplacement
New	Uusi	Créer
On Context	Tilannekohtainen opastus	Contextuelle
On Keys	Näppäimet	Touches de fonction
On Window	Yleistä	Généralités
Open	Avaa	Ouvrir
Options	Vaihtoehdot	Options
Pane	Ruutu	Carreau
Password	Salasana	Mot de passe
Paste	Lisää	Coller
Pause	Tauko	Mettre en pause
Point	Osoittaa	Pointer
Pointer	Osoitin	Pointeur
Pop-up menu	Pikavalikko	Menu instantané
Post	Pitää Näkyvissä	Figer
Press	Painaa	Appuyer
Previous	Edellinen	Précédente
Prompt	Kehote	Invite
Pull-down menu	(Valikkorivin) valikko	Menu déroulant
Quit	Lopeta	Quitter

English	Finnish	French
Release	Vapauttaa	Relâcher
Replace	Korvaa	Remplacer
Reset	Palauta	Réinitialisation
Resize	Muuttaa ikkunan kokoa	Changer la taille
Resize border	Ulkokehyksen koon muutto	Bordure Taille
Restore	Palauta	Restauration
Return	Palata	Retour
Sash	Siirtopiste	Meneau
Save	Talleta	Enregistrer
Save as	Talleta ja nimeä	Enregistrer sous
Scroll	Vierittää	Défiler
Scroll bar	Vieritysjana	Barre de défilement
Search	Etsi	Rechercher
Select	Valita	Sélectionner
Send	Lähetä	Envoyer
Session	Istunto	Session
Shift click	Painaltaa Vaihto/P1	Appuyer sur MAJ./min. et cliquer
Show	Näyttää	Montrer
Shrink	Muuttaa ikkuna kuvakkeeksi	Réduire
Shuffle down	Siirrä alle	Placer en bas de pile
Shuffle up	Siirrä päälle	Placer en haut de pile
Size	Koko	Dimension
Slider	Paikannin	Coulisse
Start	Käynnistä	Lancer
Stepping arrow	Suuntanuoli	Flèche de défilement
Stop	Pysäytä	Arrêter
Tab key	Tab-näppäin	Touche de tabulation
Text entry field	Tekstikenttä	Champ de saisie
Title bar	Otsikkorivi	Barre de titre
Toggle button	Vaihtopainike	Inverseur
Topic	Aihe	Sujet

English	Finnish	French
Trough	Vako	Rainure
Undo	Peruuta	Défaire
Update	Päivitä	Mettre à jour
View	Näytä	Visualiser
Window	Ikkuna	Fenêtre
Work area	Työskentelyalue	Zone de travail
Workspace	Työalue	Zone de travail

B.3 German and Italian Terms

English	German	Italian
Accelerator	Abkürzungsmöglichkeit	Acceleratore
Acknowledge	Bestätigen	Confermare
Activate	Aktivieren	Attivare
Active	Aktiv	Attivo
Add	Hinzufügen	Aggiungi
Alt key	Kombitaste	Tasto composizione
Apply	Anwenden	Applica
Arrow keys	Pfeiltasten	Tasti freccia
Background	Hintergrund	Base
Block cursor	Schreibmarke: Block	Tipo cursore: blocco
Border	Umrandung	Bordi
Button	Knopf	Pulsante
Cancel	Abbrechen	Annulla
Capture	Abfotografieren	Selezionare
Check button	Umschaltknopf	Pulsante
Choose	Auswählen	Scegliere
Clear	Löschen	Elimina
Click	Klicken	Premere XXX su
Clipboard	Zwischenspeicher	Area di lavoro
Close	Schließen	Chiudi
Compose sequence	Tastenkombination	Sequenza di composizione
Continue	Weiter	Continua
Control panel	Funktionspalette	Pannello di controllo
Copy	Kopieren	Copia
Cursor	Schreibmarke	Cursore
Customize	Anpassen	Preferenze
Cut	Ausschneiden	Taglia
Deactivate	Deaktivieren	Disattivare
Dialog box	Dialogbox	Finestra di dialogo
Dimmed	Schwach dargestellt	Vuoto

English	German	Italian
Dismiss	Abbrechen	Chiudi
Display	Anzeigen	Visualizzare
Double click	Zweimal hintereinander klicken	Premere due volte
Drag	Ziehen	Fare scorrere
Edit	Bearbeiten	Elaborazione
Enable	Ein	Abilitato
End session	Beenden	Chiudi sessione
Enter	Eingabe	Invio
Exit	Beenden	Uscita
Expand	Zum Fenster vergrößern	Espandi
Extract	Als Datei ablegen	Estrai
Fields	Felder	Campi
File	Steuerung	File
Filter	Filter	Filtro
Find	Suchen	Ricerca
Foreground	Vordergrund	Primo piano
Go Back	Zurück	Indietro
Go To	Zur Anwendung	Avanti
Help	Hilfe	Guida
Highlight	Hervorhebung	Evidenzia
Hold	Anhalten	Pausa
Hotspot	Querverweis	Riferimento
Icon	Sinnbild	Icona
Include	Einfügen	Includi
Index	Register	Indice analitico
Initial state	Beim Starten	Stato iniziale
Input focus	Aktivieren	Input focus
Interface	Schnittstelle	Interfaccia
Login screen	Anmeldefenster	Schermo di collegamento
Lower	In Hintergrund stellen	Inferiore
Main window	Hauptfenster	Finestra principale

English	German	Italian
Maximize	Vergrößern	Ingrandire
Maximize button	Vergrößerungsknopf	Ingrandisci pulsante
Menu	Menü	Menu
Menu bar	Menüleiste	Barra di menu
Menu item	Menüfunktion	Elemento del menu
Merge	Mischen	Unire
Minimize	Verkleinern	Ridurre
Minimize button	Verkleinerungsknopf	Riduci pulsante
Mnemonic	Kürzel	Mnemonico
Mouse	Maus	Mouse
Mouse button	Maustaste	Pulsante del mouse
Move	Verschieben	Muovere
New	Erstellen	Nuovo
On Context	Kontext	Guida oggetto
On Keys	Tasten	Guida tasti
On Window	Überblick	Guida finestra
Open	Öffnen	Apri
Options	Anpassen	Opzioni
Pane	Fensterteil	Pannello
Password	Kennwort	Parola chiave
Paste	Einfügen	Metti
Pause	Unterbrechen	Mettere in pausa
Point	Zeigen	Puntare
Pointer	Zeiger	Puntatore
Pop-up menu	Einblendmenü	Menu instantaneo
Post	Angezeigt halten	Fissare
Press	Drücken	Premere
Previous	Vorherige	Precedente
Prompt	Eingabeaufforderung	Messagdio di richiesta
Pull-down menu	Menü	Menu a tendina
Quit	Abbrechen	Abbandona

English	German	Italian
Release	Loslassen	Rilasciare
Replace	Ersetzen	Sostituisci
Reset	Zurücksetzen	Reimpostare
Resize	Größe ändern	Ridimensionare
Resize border	Bereich zur Größenänderung	Ridimensiona i bordi
Restore	Neu beginnen	Ripristinare
Return	Return	Ritorno
Sash	Trennlinie	Telaio
Save	Speichern	Salva
Save as	Speichern als	Salva come
Scroll	Abrollen	Scorrere
Scroll bar	Rolleiste	Barra di scorrimento
Search	Suchen	Search
Select	Selektieren	Selezionare
Send	Versenden	Inviare
Session	Sitzung	Sessione
Shift click	Umschalttaste gedrückt halten und MT1 klicken	Premere maiuscolo
Show	Anzeigen	Visualizzare
Shrink	Zum Sinnbild verkleinern	Ridurre ad icona
Shuffle down	In Hintergrund stellen	Indietro
Shuffle up	In Vordergrund stellen	Avanti
Size	Größe	Dimensione
Slider	Schieber	Cursore di scorrimento
Start	Starten	Iniziare
Stepping arrow	Pfeil	Freccia di scorrimento
Stop	Anhalten	Arresto
Tab key	Tabulatortaste	Tasto Tab
Text entry field	Texteingabefeld	Campo di inserimento del testo
Title bar	Titelleiste	Barra del titolo
Toggle button	Umschaltknopf	Pulsante di selezione
Topic	Thema	Argomento

English	German	Italian
Trough	Rollbereich	Slitta di scorrimento
Undo	Rückgängig	Annulla operazione
Update	Aktualisieren	Aggiorna
View	Anzeigen	Vista
Window	Fenster	Finestra
Work area	Arbeitsbereich	Area di lavoro
Workspace	Arbeitsbereich	Spazio di lavoro

B.4 Norwegian and Portuguese Terms

English	Norwegian	Portuguese
Accelerator	Snarveitast	Acelerador
Acknowledge	Bekrefte	Tomar Conhecimento
Activate	Aktivere	Activar
Active	Aktiv	Activa
Add	Tilføy	Incluir
Alt key	Alt-tast	Tecla Alt
Apply	Bruk	Aplicar
Arrow keys	Piltaster	Teclas de Direcção
Background	Bakgrunn	Segundo Plano
Block cursor	Blokkmarkør	Cursor Rectangular
Border	Ytre ramme	Moldura
Button	Knapp	Botão
Cancel	Avbryt	Cancelar
Capture	Skrive	Capturar
Check button	Av/på-knapp	Botão de Verificação
Choose	Velge	Escolher
Clear	Fjern	Limpar
Click	Klikke	Clicar
Clipboard	Utklippstavle	Painel
Close	Lukke	Fecho
Compose sequence	Tastsekvens	Sequência de Composição
Continue	Fortsette	Continuar
Control panel	Styrepanel	Painel de Controle
Copy	Kopiere	Copiar
Cursor	Markør	Cursor
Customize	Tilpasse	Adaptar
Cut	Ta ut	Cortar
Deactivate	Deaktivere	Desactivar
Dialog box	Dialogrute	Modelo de Diálogo
Dimmed	Matt	Esbatido

English	Norwegian	Portuguese
Dismiss	Avbryt	Dispensar
Display	Vise	Visualizar
Double click	Dobbeltklikk	Clique Duplo
Drag	Dra	Arrastar
Edit	Redigere	Editar
Enable	På	Activar
End session	Avslutte økt	Terminar a Sessão
Enter	Enter	Enter
Exit	Avslutte	Saída
Expand	Utvide	Expandir
Extract	Lage fil	Extrair
Fields	Felt	Campos
File	Arkiv	Ficheiro
Filter	Utvalg	Filtro
Find	Finne	Pesquisar
Foreground	Forgrunn	Primeiro Plano
Go Back	Gå tilbake	Recuar
Go To	Gå til	Ir Para
Help	Hjelp	Auxílio
Highlight	Utheve	Realce
Hold	Holde	Reter
Hotspot	Referanse	Ponto Importante
Icon	Symbol	Ícone
Include	Hente	Incluir
Index	Stikkordregister	Índice
Initial state	Starte som	Estado Inicial
Input focus	Aktivt vindu	Ponto de Entrada de Dados
Interface	Grensesnitt	Interface
Login screen	Innloggingsbilde	Écran de Acesso
Lower	Flytte bakerst	Inferior
Main window	Hovedvindu	Janela Principal

English	Norwegian	Portuguese
Maximize	Stor	Maximizar
Maximize button	Stor-knappen	Botão Maximizar
Menu	Meny	Menu
Menu bar	Menylinje	Barra dos Menus
Menu item	Menyvalg	Item do Menu
Merge	Samordne	Fundir
Minimize	Krympe	Minimizar
Minimize button	Krympeknapp	Botão Minimizar
Mnemonic	Kommandobokstav	Mnemónica
Mouse	Mus	Rato
Mouse button	Museknapp	Botão do Rato
Move	Flytte	Mover
New	Nytt	Novo
On Context	Om konkret punkt	No Contexto
On Keys	Om tastene	Sobre as Teclas
On Window	Om vinduet	Sobre a Janela
Open	Åpne	Abrir
Options	Tilpasse	Opções
Pane	Vindusdel	Painel
Password	Passord	Senha
Paste	Sette inn	Repor
Pause	Pause	Pôr em Pausa
Point	Peke	Apontar
Pointer	Peker	Ponteiro
Pop-up menu	Sprettoppmeny	Menu Rápido
Post	Feste	Fixar
Press	Trykk på	Premir
Previous	Forrige	Anterior
Prompt	Ledetekst	Indicador
Pull-down menu	Rullegardinmeny	Menu Descendente
Quit	Avbryt	Abandonar

English	Norwegian	Portuguese
Release	Frigjøre	Largar
Replace	Erstatte	Substituir
Reset	Tilbakestill	Repor
Resize	Endre størrelse på	Redimensionar
Resize border	Justere størrelse på ytre ramme	Moldura Redimensionável
Restore	Gjenopprette	Repor
Return	Retur	Return
Sash	Vindushåndtak	Linha Divisória
Save	Lagre	Guardar
Save as	Lagre som	Guardar Como
Scroll	Rulle	Percorrer
Scroll bar	Rullebane	Barra Cursora
Search	Søke	Pesquisar
Select	Velge	Seleccionar
Send	Sende	Enviar
Session	Ut	Sessão
Shift click	Skift-klikke	Clicar com Shift
Show	Vise	Mostrar
Shrink	Krympe	Reduzir
Shuffle down	Flytte bakerst	Pôr Atrás
Shuffle up	Flytte forrest	Pôr à Frente
Size	Størrelse	Dimensão
Slider	Rullemerke	Régua Deslizante
Start	Starte	Iniciar
Stepping arrow	Rullepil	Seta Cursora
Stop	Stopp	Parar
Tab key	Tab-tasten	Tecla de Tabulação
Text entry field	Skrivefelt	Campo de Entrada de Texto
Title bar	Tittellinjen	Barra de Títulos
Toggle button	Av/på-knapp	Botão de Comutação
Topic	Emne	Tópico

English	Norwegian	Portuguese
Trough	Rullefelt	Calha
Undo	Angre	Anular
Update	Oppdatér	Actualizar
View	Se på	Visualização
Window	Vindu	Janela
Work area	Arbeidsområde	Área de Trabalho
Workspace	Arbeidsområde	Espaço de Trabalho

B.5 Spanish and Swedish Terms

English	Spanish	Swedish
Accelerator	Acelerador	Tangentkommando
Acknowledge	Informar	Bekräfta
Activate	Activar	Aktivera
Active	Activo	Aktiv
Add	Añadir	Lägg till
Alt key	Tecla Alt	Alt-tangent
Apply	Aplicar	Tillämpa
Arrow keys	Teclas de flecha	Piltangenter
Background	Fondo	Bakgrund
Block cursor	Cursor en bloque	Blockmarkör
Border	Borde	Ram
Button	Botón	Knapp
Cancel	Cancelar	Avbryt
Capture	Capturar	Spara
Check button	Botón de comprobación	Av/på-knapp
Choose	Elegir	Välja
Clear	Suprimir	Rensa
Click	Marcar	Klicka
Clipboard	Area de recorte	Urklipp
Close	Cerrar	Stänga
Compose sequence	Secuencia de composición	Tangentsekvens
Continue	Continuar	Fortsätt
Control panel	Panel de control	Kontrollpanel
Copy	Copiar	Kopiera
Cursor	Cursor	Markör
Customize	Personalizar	Anpassa
Cut	Cortar	Klippa
Deactivate	Desactivar	Avaktivera
Dialog box	Área de diálogo	Dialogruta
Dimmed	Difuminado	Skuggad

English	Spanish	Swedish
Dismiss	Rechazar	Avbryt
Display	Visualizar	Visa
Double click	Remarcar	Dubbelklicka
Drag	Arrastrar	Dra
Edit	Editar	Redigera
Enable	Activa	På
End session	Terminar sesión	Avsluta session
Enter	Validar	Enter
Exit	Salir	Avsluta
Expand	Ampliar	Förstora
Extract	Extraer	Kopiera till
Fields	Campos	Fält
File	Fichero	Fil
Filter	Filtro	Filter
Find	Localizar	Sök
Foreground	Primer plano	Förgrund
Go Back	Retroceder	Gå tillbaka
Go To	Ir a	Gå till
Help	Ayuda	Hjälp
Highlight	Realzar	Markera
Hold	Bloquear	Hålla
Hot spot	Referencia	Snabbreferens
Icon	Icono	Symbol
Include	Incluir	Infoga
Index	Índice	Sakregister
Initial state	Estado inicial	Startläge
Input focus	Marca de entrada	Aktivt fönster
Interface	Interfaz	Gränssnitt
Login screen	Pantalla de acceso	Inloggningsbild
Lower	Posponer	Undre
Main window	Ventana principal	Huvudfönster

English	Spanish	Swedish
Maximize	Maximizar	Förstora
Maximize button	Botón "Maximizar"	Förstoringsknapp
Menu	Menú	Meny
Menu bar	Barra de menús	Menyrad
Menu item	Opción de menú	Menyalternativ
Merge	Fundir	Sammanfoga
Minimize	Minimizar	Krympa
Minimize button	Botón "Minimizar"	Krympknappen
Mnemonic	Mnemotecnia	Menykod
Mouse	Ratón	Mus
Mouse button	Botón de ratón	Musknapp
Move	Mover	Flytta
New	Nuevo	Ny
On Context	Sobre el contexto	Om konkret punkt
On Keys	Sobre teclas	Om tangenter
On Window	Sobre la ventana	Om fönster
Open	Abrir	Öppna
Options	Opciones	Tillval
Pane	Cristal	Fönsterdel
Password	Clave de acceso	Lösenord
Paste	Pegar	Klistra
Pause	Hacer una pausa	Göra en paus
Point	Apuntar (a)	Peka
Pointer	Puntero	Pekare
Pop-up menu	Menú contextual	Snabbmeny
Post	Fijar	Fästa
Press	Pulsar	Trycka
Previous	Anterior	Föregående
Prompt	Indicador	Ledtecken
Pull-down menu	Menú desplegable	Rullgardinsmeny
Quit	abandonar	Gå ur

English	Spanish	Swedish
Release	Soltar	Frigöra
Replace	Sustituir	Ersätt
Reset	Restablecer	Återställa
Resize	Cambiar de tamaño	Omforma
Resize border	Cambiar tamaño de borde	Ramstorlek
Restore	Restaurar	Återställa
Return	Retorno	Retur
Sash	Separador de ventanas dobles	Fönsterhake
Save	Salvar	Spara
Save as	Salvar como	Spara som
Scroll	Desplazarse	Rulla
Scroll bar	Barra de desplazamiento	Rullningslist
Search	Buscar	Söka
Select	Seleccionar	Välja
Send	Enviar	Sända
Session	Sesión	Session
Shift click	Cambiar y marcar	Skiftklicka
Show	Mostrar	Visa
Shrink	Reducir	Krympa
Shuffle down	Relegar	Lägg fönster underst
Shuffle up	Anteponer	Lägg fönster överst
Size	Tamaño	Storlek
Slider	Guía	Rullningsruta
Start	Arrancar	Starta
Stepping arrow	Flecha de deslizamiento	Piltangent
Stop	Parar	Stopp
Tab key	Tecla tabulación	Tabtangent
Text entry field	Campo para introducción de texto	Textfält
Title bar	Barra de título	Titelrad
Toggle button	Botón de conmutación	Av/på-knapp
Topic	Tema	Ämne

English	Spanish	Swedish
Trough	Pasillo de desplazamiento	Rullningsfält
Undo	Cancelar función	Ångra
Update	Actualizar	Uppdatera
View	Ver	Visa
Window	Ventana	Fönster
Work area	Área de trabajo	Arbetsarea
Workspace	Espacio de trabajo	Arbetsarea

C

Sources of Further Information

This appendix lists books and periodicals which may provide you with additional information concerning user interface design.

C.1 Books

- Baecker, R., and W. Buxton, eds. *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Morgan Kaufmann, 1987.
- Barbacetto, Gianni. *Design Interface*. Milano: Arcadia Edizioni, 1987.
- Berryman, G., *Notes on Graphic Design and Visual Communication*, William Kaufmann, 1984.
- Bertin, J., *Semiology of Graphics*, University of Wisconsin Press, 1983.
- Bolt, R.A., *The Human Interface: Where People and Computers Meet*, Van Nostrand Reinhold, 1984.
- Caplan, R. *By Design*. New York: McGraw-Hill, 1982.
- Card, S.K., et.al., *The Psychology of Human-Computer Interaction*, Lawrence Earlbaum Associates, 1983.
- Carroll, J.M., *Interfacing Thought: Cognitive Aspects of Human Computer Interaction*, Lawrence Earlbaum Associates, 1987.
- Coombs, M.J., and J.L. Alty, *Computing Skills and the User Interface*, Academic Press, 1981.
- Dondis, D.A., *A Primer of Visual Literacy*. Cambridge, MA: The MIT Press, 1973.
- Dreyfuss, H., *Symbol Sourcebook, an Authoritative Guide to International Graphic Symbols*, Van Nostrand Reinhold, 1984.
- Ehn, P. *Work-Oriented Design of Computer Artifacts*. Stockholm: Arbetslivscentrum, 1988.

- Green, T., and E. Edmonds, *The Ergonomics of the User Interface. Behaviour and Information Technology Special Issue Series*, vol. 3, no. 2, Taylor and Francis, 1984.
- Guedj, R.A., et. al., *Methodology of Interaction*, North-Holland, 1980.
- Heckel, P., *Elements of Friendly Software Design*, Warner Books, 1984.
- Helander, M. *The Handbook of Human-Computer Interaction*, North-Holland, 1988.
- Hurlbutt, A. *The Grid*. New York: van Nostrand Reinhold, 1978.
- Itten, J., *The Elements of Color*, F. Birren, ed.,Van Nostrand Reinhold, 1970.
- McConnell, V., *Building the End User Interface*, Addison-Wesley, 1983.
- Nickerson, R., *Using Computers: Human Factors in Information Systems*, MIT Press, 1986.
- Norman, D.A., and S. Draper, eds., *User Centered System Design*, Lawrence Earlbaum Associates, 1986.
- Pfaff, G., ed., *User Interface Management Systems*, Springer-Verlag, 1985.
- Rubenstein, R., and H. Hersh, *The Human Factor*, Digital Press, 1984.
- Schneiderman, B., ed., *Data Bases: Improving Usability and Effectiveness*, Academic Press, 1978.
- Schneiderman, B., *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop, 1980.
- Schneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley, 1987.
- Smith, H.T., and T.R.G. Green, eds., *Human Interaction with Computers*, Academic Press, 1980.
- Tufte, E.R., *The Visual Display of Quantitative Information*, Graphics Press, 1983.
- Tufte, E.R., *Envisioning Information*, Graphics Press, 1990.
- Vassilou, Y., *Human Factors and Interactive Computer Systems*, Ablex, 1982.
- Whitney, P, and C. Kent, eds., *Design in the Information Environment*, Knopf, 1985.

- Wildbur, P. *Information Graphics*. New York: Van Nostrand Reinhold, 1989.
- Zuboff, S. *In the Age of the Smart Machine*, Basics Books, 1988.

C.2 Periodicals

- *acm Transations on Information Systems*, published by the Association for Computer Machinery
- *Cognitive Science*, Journal of the Cognitive Science Society
- *Ergonomics* published by the Ergonomics Research Society
- *Human-Computer Interaction*, published by Lawrence Erlbaum Associates
- "Human Factors in Computing Systems," an annual issue of the *SIGCHI Bulletin*, the journal of the Association of Computing Machinery's Special Interest Group on Computers and Human Interaction (SIGCHI)
- *Human Factors: The Journal of the Human Factors Society*
- *Human Factors Review*, published by the Human Factors Society
- *IEEE Transactions on Systems, Man, and Cybernetics*
- *International Journal of Man-Machine Studies*
- *Proceedings of the Annual Conference of Cognitive Science Society*
- *Technical Communication*, a publication of the Society for Technical Communication

Index

A

Accelerator
  creating for menu items, 2–3
Additional topics in Help window, 5–17
  displaying, 5–17
  selecting, 5–17
Adjectives as menu names and menu items, 2–2
Aligning entries in SVN widget, 6–11
Aligning labels, 1–45
Aligning labels with the baseline of text, 1–45
Alt key, A–2
Alt+Backspace, A–9, A–14
Alt+Delete, A–9, A–14
Alt+F6, A–8
Alt+Left Arrow, A–5
Alt+MB2, A–12
Alt+Right Arrow, A–5
Alt+Shift+F6, A–8
Alt+Shift+MB2, A–12
Alt+Shift+Tab, A–8
Alt+Tab, A–8
Applications
  printing files created in, 6–1
Appropriate and inappropriate titles in dialog boxes, 1–15
Appropriate defaults, 1–51
Arrow keys to move within tab groups, 1–59
Arrows
  outer, in SVN widget, 6–8

B

Backspace key, A–3
Baseline of text, alignment with label, 1–45
Basic choices in nested dialog boxes, 1–9
BCustom, A–12
BDrag, A–12
Bevel
  definition of, 1–38
  margins of, 1–43
BExtend, A–12
Bindings
  keyboard, A–1 to A–14
BMenu, A–12
Boldfacing, 1–54
BPrimaryCopy, A–12
BPrimaryCut, A–12
BPrimaryPaste, A–12
BQuickCopy, A–12
BQuickCut, A–12
BQuickPaste, A–12
Brevity in Help topics, 5–11
Brightness of color, 4–3
Browser color model
  definition, 4–7
Browser values
  definition, 4–7
BSelect, A–12
BToggle, A–12

C

Cancel button, 1–22
Capitalization in dialog boxes, 1–61 to 1–62
Capitalization in menus, 1–61 to 1–62
Cascade menu
  as a tear-off menu, 2–12
Caution pointer
  definition of, 1–4
Caution pointer with a modal dialog box, 1–4
Centered labels in push buttons, 1–22
Changing the cursor, 1–5
Changing the pointer, 1–5
Clarity in Help topics, 5–12
Clear menu item in Edit menu, 2–26
Clearing selected range, 2–11
Close menu item in File menu, 2–21
Close menu item, accelerator for, A–13
Close message box, 2–21
Closing primary windows, 2–21
Color, 4–1 to 4–7
  as redundant cue, 4–3
  brightness and saturation, 4–3
  browser values, 4–7
  codes or keys, 4–3
  customizing, 3–8
  defining, 4–5
  in dialog boxes, 4–5
  in menus, 4–5
  in spreadsheets, 4–1
  of pointer, 4–5
  RGB values, 4–7
  selecting, 4–2 to 4–4
  to correspond with conventions, 4–2
  to indicate differences, 4–1
  to indicate keyboard focus, 4–1
  to show relationships, 4–1
  using, 4–5 to 4–7
  using blue, 4–4
  using different colors together, 4–4
  using red, 4–3, 4–4
  why use, 4–1
Color models, browser, 4–7
Color models, color picker, 4–5
Color models, greyscale mixer, 4–7
Color models, HLS, 4–7
Color models, RGB, 4–7
Colorblindness, 4–2
ColorMix widget, 4–5 to 4–7
  customizing colors with, 3–8
  overview, 4–5
Complementary menu items, 2–4
Conserving space with option buttons, 1–52
Context-sensitive help, 5–1
Context-sensitive Help menu item in Help menu, 2–28
Controls, 1–1 to 1–64
Controls in dialog boxes, placement of, 1–38
Copy, A–14
Copy menu item in Edit menu, 5–16
Ctrl key, A–2
Ctrl+Alt+Left Arrow, A–5
Ctrl+Alt+Right Arrow, A–5
Ctrl+Backslash, A–11
Ctrl+Delete, A–7
Ctrl+Down Arrow, A–7
Ctrl+End, A–5
Ctrl+Home, A–5
Ctrl+Insert, A–11, A–14
Ctrl+Insert Here, A–11, A–14
Ctrl+Left Arrow, A–7
Ctrl+MB1, A–12
Ctrl+MB2, A–12
Ctrl+Next, A–5
Ctrl+PageDown, A–5
Ctrl+PageUp, A–5
Ctrl+Prev, A–5
Ctrl+Return, A–9
Ctrl+Right Arrow, A–7
Ctrl+Shift+Delete, A–7
Ctrl+Shift+Insert, A–11
Ctrl+Shift+Insert Here, A–11
Ctrl+Shift+MB2, A–12
Ctrl+Shift+Space bar, A–10
Ctrl+Slash, A–10

Ctrl+Tab, A–6
Ctrl+Up Arrow, A–7
Cursor
wait, 1–63
Customized settings
life of, 3–4
Customizing
color, 3–8
dialog boxes, 3–5
extent of, scope of, 3–2
general guidelines, 3–1
menu items, 3–5
overriding previous settings, 3–3
text, 3–8
where to customize, 3–3
window positions, 3–8
Customizing applications, 3–1 to 3–11
Cut, A–14
Cut menu item in Edit menu, 2–25

D

Decorations, on windows, 1–14
decw$cursor.h file, 1–5
decw$include/cursorfont.h file, 1–5
Default push button, 1–21
Defaults, providing appropriate ones, 1–51
Delete key, A–3, A–5
Delete menu item in Edit menu, 2–26
Description in SVN widget, 6–10
Deselecting
when not to, 2–11
Dialog box placement with respect to the screen, not the parent, 1–9
Dialog boxes, 1–1 to 1–64
bevel within, 1–43
capitalization in, 1–61 to 1–62
clarifying structure within, 1–50
creating push buttons in, 1–21 to 1–30
definition of, 1–2
determining the type and placement of, 1–2 to 1–13
disabling buttons in, 2–8
error box, 1–34
full application modal, 1–6
Dialog boxes (cont'd)
grouping of, 1–38
gutters within, 1–43
information box, 1–35
initializing values in, 1–53
invoking nested ones, 1–9
layout of, 1–38 to 1–64
levels of, 1–13
maximizing, 1–2
minimizing, 1–2
modal, 1–4 to 1–6
modeless, 1–2
moving, 1–9
nesting, 1–9
placement of, 1–7, 1–9
presenting text in, 1–61 to 1–62
primary application modal, 1–4
Print menu item, 2–20
punctuation in, 1–39
question box, 1–32
saving user placement of, 1–9
spacing within, 1–42
standard message, 1–31 to 1–38
title of, 1–15
types of, 1–2, 1–31 to 1–38
using them to customize an application, 3–5
using window decorations in, 1–14 to 1–17
warning box, 1–36
when to use, 1–1
working box, 1–37, 1–63
Digital-specific widgets, 6–1 to 6–16
Dimensions of push buttons, 1–28
Disabling menu items, 2–6 to 2–8
Down arrow, A–4
DXmCreateCursor, 1–6
DXmFormSpaceButtonsEqually, 1–18, 1–20, 1–24

E

Edit menu, 2–22 to 2–26
  Clear menu item, 2–26
  Cut menu item, 2–25
  Delete menu item, 2–26
  Find menu item, 2–26
  functions of, 2–24 to 2–26
  grouping of items in, 2–24
  in Help window, 5–16
  Paste menu item, 2–25
  Redo menu item, 2–24
  Select All menu item, 2–26
  Undo menu item, 2–24
Elements of a title in a dialog box title area, 1–15
Ellipses
  when to use, 1–10
Ellipses in menu items, 2–5
End key, A–5
Enter key, A–9
Equally spaced push buttons at the bottom of a dialog box, 1–23
Error dialog box, 1–34
Escape key, A–3
  enabling as the Cancel key, 1–22
Exit menu item, 5–16
Exit menu item in File menu, 2–21
Exit message box, 2–21
Export menu item in File menu, 2–20

F

F1 key, A–3
F10 key, A–6
F11 key, A–3
Fields
  margins between, 1–43
  margins within, 1–43
File menu, 2–13 to 2–21
  Close menu item, 2–21
  Exit menu item, 2–21
  Export menu item, 2–20
  functions of, 2–15 to 2–21
File menu (cont'd)
  Import menu item, 2–17
  in Help window, 5–16
  New menu item, 2–15
  Print menu item, 2–20
  Revert menu item, 2–20
  Save As menu item, 2–18
  Save menu item, 2–18
Files
  printing, 6–1
Find menu item in Edit menu, 2–26
Fixed-size controls and no resize borders, 1–16
Flashing, 1–54
Font units
  between vertically spaced push buttons, 1–28
  of the bevel margin, 1–43
  of the gutter, 1–43
  of the window margin, 1–42
Fonts
  customizing, 3–9
  using to show hierarchy, 6–12
Frames, see bevels, 1–43
Full application modal dialog box, 1–6
Full selection line in the SVN widget, 6–12

G

Global selection, 2–11
Glossary menu item in Help menu, 2–29
Go Back menu item, 5–16
Go To menu item, 5–16
Go to overview menu item, 5–16
Graphic arrangement of information in Help topics, 5–13
Grouping in dialog boxes, 1–38
Grouping menu items, 2–3 to 2–5
Grouping related choices using space or bevels, 1–38
Gutter size, 1–43

H

Handling One Push Button That Is Much Bigger Than Others, 1–29
Help, 5–1 to 5–19
  being visual, 5–13
  brevity in topics, 5–11
  clarity in topics, 5–12
  designing, 5–1 to 5–14
  Edit menu, 5–16
  File menu, 5–16
  invoking, 5–1, 5–14
  organizing, 5–5
  planning, 5–2
  pull-down menu, 5–1, 5–14
  Search menu, 5–17
  Using Help menu, 5–17
  View menu, 5–16
  window, 5–14
  writing overviews of, 5–6
  writing specific topics for, 5–11
Help box
  History, 5–17
Help command, 5–2
Help key, 5–1, 5–2, 5–14, A–3
Help menu, 2–27 to 2–29
  Context-sensitive Help menu item, 2–28
  Glossary menu item, 2–29
  Index menu item, 2–28
  Keyboard menu item, 2–28
  Overview menu item, 2–28
  Product Information menu item, 2–29
  Tutorial menu item, 2–28
  Using Help menu item, 2–29
Help push button, 1–10, 1–30, 5–2
Help topics, arrangement of information, 5–13
Help window, 5–14
  additional topics, 5–17
  components, 5–14
  invoking, 5–1, 5–14
  invoking by command, 5–2
  invoking with Help key, 5–2
  invoking with push button, 1–30, 5–2
Help window (cont'd)
  invoking with the Help menu, 5–1
  menu bar, 5–16
  push buttons, 5–17
  title bar, 5–15
  topic, 5–17
Hierarchy
  using fonts within, 6–12
History menu item, 5–17
HLS color model, 4–7
Home key, A–5
Hue, Lightness, Saturation Color Model
  See HLS color model

I

Icon in SVN widget, 6–10
Icon states, 6–11
Import menu item in File menu, 2–17
Increasing size and layout of dialog box proportionally, 1–20
Index menu item in Help menu, 2–28
Index window in SVN widget, 6–9
Information dialog box, 1–35
Input devices
  keyboard, A–1 to A–14
Interface customization, 3–1
Invoking nested dialog boxes, 1–9

K

KActivate, A–9
KAddMode, A–11
KBackSpace, A–3
KBackTab, A–3
KBeginData, A–5
KBeginLine, A–5
KCancel, A–3
KCopy, A–11
KCut, A–11
KDelete, A–5
KDeselectAll, A–11
KDown, A–4

Keeping objects selected, 2–11
KEndData, A–5
KEndLine, A–5
KEnter, A–3
KEraseEndLine, A–7
KExtend, A–10
Keyboard access, A–1 to A–14
Keyboard bindings, A–1 to A–14
Keyboard menu item in Help menu, 2–28
Keyboard traversal, A–1 to A–14
KHelp, A–3
KLeft, A–4
KMenu, A–6
KMenuBar, A–6
KNextFamilyWindow, A–8
KNextField, A–6
KNextPara, A–7
KNextWindow, A–8
KNextWord, A–7
KPageDown, A–4
KPageLeft, A–5
KPageRight, A–5
KPageUp, A–4
KPaste, A–11
KPrevFamilyWindow, A–8
KPrevField, A–6
KPrevPara, A–7
KPrevWindow, A–8
KPrevWord, A–7
KReselect, A–10
KRestore, A–11
KRight, A–4
KSelect, A–10
KSelectAll, A–10
KSpace, A–3
KTab, A–3
KUndo, A–9
KUp, A–4
KWindowMenu, A–8

L

Labeling of radio buttons, 1–49
Labels
  aligning, 1–45
    in push buttons, 1–22
    not overlapping in push buttons, 1–20
    not truncating in push buttons, 1–19
Layout of screen objects
    dialog boxes, 1–38
    Help topics, 5–13
Left Arrow, A–4
Left-justified and horizontal arrangement of controls in dialog boxes, 1–44
Levels of dialog boxes, 1–9
Levels of submenus, 2–4
Limited selection line in the SVN widget, 6–13
Line width in SVN selection line, 6–14
Live Scrolling, 6–8

M

MAlt, A–2
Maximize button, when to have, 1–14
MB1, A–12
MB2, A–12
MB3, A–12
MCtrl, A–2
Menu bar in Help window, 5–16
Menu item
    dimming, 2–6 to 2–8
    disabling, 2–6 to 2–8
    grouping, 2–3 to 2–5
    opposites, 2–4
    using ellipses in, 2–5
Menu map, 2–4
Menus, 2–1 to 2–30
    capitalization in, 1–61 to 1–62
    File menu, 2–13
    naming, 2–1
    presenting text in, 1–61 to 1–62
    submenus, number of, 2–4
    terms for naming, 2–3

Menus (cont'd)
using them to customize an application, 3–5
Message boxes
error, 1–34
information, 1–35
question, 1–32
warning, 1–36
working, 1–37
Message dialog boxes, 1–31 to 1–38, 2–21
Minimize button, when to have, 1–14
Minimum size for resizeable dialog boxes, 1–17
Mnemonic
creating for menu items, 2–3
Mnemonic, definition of, 2–13
Mnemonics in menus, 2–13
Modal dialog boxes, 1–4
Modeless dialog boxes, 1–2
Modeless dialog boxes allow users to work in other windows, 1–2
Most frequent controls in dialog boxes
placement of, 1–9
Most important controls in dialog boxes
placement of, 1–9
Mouseless interaction, A–1 to A–14
Moving controls closer together, 1–18
Moving dialog boxes, 1–9
Moving push buttons closer together when shrinking a dialog box, 1–18
MShift, A–2
Multilevel Undo, 2–24
Multiple dialog boxes
placement of, 1–9

N

Naming menus and menu items, 2–1
Naming push buttons, 1–9
Navigating
between tab groups, 1–55
using the keyboard, A–1 to A–14
within tab groups, 1–59
Navigation, 1–55
lists, 1–55
sashes, 1–55
scales, 1–55
text, 1–55
Nesting dialog boxes, 1–9, 1–13
basic choices in the first box, 1–9
number of, 1–9
when to do so, 1–9
Nesting secondary dialog boxes, 1–9
New menu item in File menu, 2–15
New menu item, accelerator for, A–13
Next key, A–4
Number of submenus, 2–4

O

Open in new window check box, 2–15
Open menu item, 2–15
Open menu item, accelerator for, A–13
Opposite menu items, 2–4
Option button to display an option menu, 1–52
Option buttons conserving space, 1–52
Option menu from an option button, 1–52
Options menu, 2–27, 3–1, 3–8
as a tear-off menu, 2–12
Options... button to indicate a secondary dialog box, 1–9
Options... push button, 1–9
Outer arrows, 6–8
Outlining Help topics, 5–2
Overview menu item in Help menu, 2–28

P

PageDown key, A–4
PageUp key, A–4
Parent window, placing dialog boxes over, 1–7
Paste, A–14
Paste menu item in Edit menu, 2–25
Pasting data, 2–25

Place dialog boxes appropriately, 1–7
Placing dialog boxes, 1–7, 1–9
Placing dialog boxes, saving the users' placement, 1–9
Pop-up menus
  as a tear-off menu, 2–12
  compared with push buttons, 2–11
  decorations of, 2–9
  designing, 2–8 to 2–12
  dimming items in, 2–10
  displaying, 2–11
  implementing, 2–10
  items in, 2–11
  placement of, 2–10
  selection, 2–11
  with submenu, 2–11
Prev Screen, A–4
Primary application modal dialog box, 1–4
Primary windows
  closing, 2–21
  placing dialog boxes over, 1–7
  title bars in, 1–62
Print dialog box, 2–20
Print menu item, 2–20
Print menu item..., accelerator for, A–13
Print queue
  submitting files to, 6–1
Print widget, 6–1
Printing files
  created in applications, 6–1
Product Information menu item in Help menu, 2–29
Programming Hints
  aligning labels with the baseline of text, 1–46
  changing the cursor, 1–5
  creating a bevel, 1–39
  dialog boxes without resize borders, 1–17
  enabling the cancel button, 1–22
  enabling the default push button, 1–22
  making all fields line up after a label, 1–44
  making an extra-long push button, 1–29
  making application modal dialog boxes, 1–33
Programming Hints (cont'd)
  making controls move closer together, 1–18
  making dialog boxes modal or modeless, 1–3
  nesting dialog boxes, 1–13
  placing dialog boxes, 1–8
  resizing dialog boxes, 1–20
  setting a minimum size for a dialog box, 1–18
  setting the title of a dialog box, 1–16
  spacing an even number of push buttons equally, 1–26
  spacing an odd number of push buttons equally, 1–25
  spacing and sizing push buttons, 1–24
  specifying margins in dialog boxes, 1–42
  thickness of bevel shadow, 1–44
Pull-down menu
  as a tear-off menu, 2–12
Punctuation in dialog boxes, 1–39
Push buttons
  arrangement of, 1–22
  centering labels in, 1–22
  creating, 1–21 to 1–30
  default, 1–21
  dimensions of, 1–28
  do not overlap, 1–20
  do not truncate, 1–19
  guidelines for labeling, 1–9
  horizontal arrangement of, 1–22
  in Help window, 5–17
  in standard message dialog boxes, 1–31 to 1–38
  labels in, 1–22
  moving closer together when shrinking a dialog box, 1–18
  order in dialog boxes, 1–9 to 1–13
  placement, 1–30
  vertical arrangement of, 1–22
Push buttons at the side of a dialog box, 1–28

Q

Question dialog boxes, 1–32
  examples of, 1–32

R

Radio buttons, labeling, 1–49
Range of customization, 3–2
Red, Green, Blue color model
  See RGB color model
Redo menu item, 2–24
  disabling, 2–24
Remove, A–11
Remove key, A–5
Resize borders
  when to have, 1–14
Resize borders with text-entry fields or list boxes, 1–17
Resize handles
  see resize borders, 1–14
Resizing dialog boxes, 1–16 to 1–20
  set a minimum size, 1–17
  with text-entry fields or list boxes, 1–16
Return key, A–3, A–9
Revert menu item, 2–19
RGB color model, 4–7
RGB values, 4–7
Right Arrow, A–4

S

Structured Visual Navigation, see SVN
Saturation of color, 4–3
Save As menu item in File menu, 2–18, 5–16
Save menu item in File menu, 2–18
Save menu item, accelerator for, A–13
Saving the users' placement of dialog boxes, 1–9
Saving the users' placement of window positions, 3–8
Scope of customization, 3–2
Screen objects, guidelines for capitalizing and punctuating, 1–61 to 1–62
Scrolling, live, 6–8
Search menu in the Help window, 5–17
Secondary dialog boxes
  nesting, 1–9
Secondary windows
  closing, 2–21
  title bars in, 1–63
Select All menu item in Edit menu, 2–26, 5–16
Select key, A–10
Selected menu, 2–21 to 2–22
Selection, 2–11
  canceling the current one, 2–11
  guidelines relating to pop-up menus, 2–11
  with pop-up menus, 2–11
Selection controls, provide a variety, 1–54
Selection modes in the SVN widget, 6–12
Separators
  in menus, 2–3
Settings
  life of, 3–4
Shift key, A–2
Shift+Ctrl+Space bar, A–10
Shift+Delete, A–11, A–14
Shift+F10, A–6
Shift+F11, A–8
Shift+F8, A–11
Shift+Insert, A–11, A–14
Shift+Insert Here, A–11, A–14
Shift+MB1, A–12
Shift+Remove, A–14
Shift+Select, A–10
Shift+Tab, A–3, A–6
Single-level Undo, 2–24
Size of dialog box
  decreasing, 1–18
  increasing, 1–20
Size of gutter, 1–43
Size of window margin, 1–42

Smooth scrolling, 6–8
Space bar, A–3, A–10
Spacing between labels and fields, 1–44
Spacing in dialog boxes, 1–42
Spacing of push buttons
  along the side of a dialog box, 1–28
Standard message dialog boxes, 1–31 to 1–38
Structure of dialog box
  how to clarify, 1–50
Submenus
  disabling, 2–7
  displaying, 2–7
  number of, 2–4
  with pop-up menu, 2–11
SVN, 6–5 to 6–16
  aligning entries, 6–11
  column format, 6–6
  components of an entry in, 6–10
  description, 6–10
  display formats, 6–6
  extra menu items in the View menu, 6–15
  full selection line, 6–12
  icon, 6–10
  icon states, 6–11
  index window in, 6–9
  limited selection line, 6–13
  modes, 6–6
  outline format, 6–6
  selection line width, 6–14
  selection modes, 6–12, 6–13
  text component, 6–10
  tree format, 6–6

T

Tab groups
  canvases, 1–55
  check buttons, 1–55
  definition of, 1–54
  lists, 1–55
  margins between, 1–43
  margins within, 1–43
  navigating between, 1–55
Tab groups (cont'd)
  navigating within, 1–59
  option menu, 1–55
  radio buttons, 1–55
  sashes, 1–55
  text, 1–55
Tab key, A–3, A–6
Tab key to move between tab groups, 1–56
Tear-off menus
  designing, 2–12
Terminology for menus, 2–3
Terms for menus and menu items, 2–1
Text
  customizing, 3–8
Text in SVN widget, 6–10
Text, selection of
  See Selection
Title
  elements of in a dialog box, 1–15
Title area
  on a dialog box, 1–14
  when to have, 1–14
Title bars
  creating, 1–62 to 1–63
Translation
  of online help, 5–14
Traversal
  using the keyboard, A–1 to A–14
Truncate push buttons, do not, 1–19
Tutorial menu item in Help menu, 2–28

U

UIL
  for aligning labels with baseline of text, 1–46
  for application modal dialog boxes, 1–33
  for creating a bevel, 1–39
  for dialog boxes without resize borders, 1–17
  for enabling the cancel button, 1–22
  for enabling the default push button, 1–22
  for making all fields line up after a label, 1–44

UIL (cont'd)
for margins in dialog boxes, 1–42
for modal dialog boxes, 1–3
for modeless dialog boxes, 1–3
for nesting dialog boxes, 1–13
for placing dialog boxes, 1–8
for setting the title of a dialog box, 1–16
for spacing push buttons equally, 1–25, 1–26
for thickness of bevel shadow, 1–44
Underlining, 1–54
Undo, A–9, A–14
multilevel, 2–24
single level, 2–24
Undo menu item, 2–24
disabling, 2–24
Uniform spacing in dialog boxes, 1–42
Up Arrow, A–4
Using Help menu, 5–17
Using Help menu item in Help menu, 2–29

V

Verbs as menu names and menu items, 2–2
Vertical arrangement of push buttons, 1–28
View menu, 2–26 to 2–27
in Help window, 5–16
items for the SVN widget, 6–15
Visit menu item, 5–16
Visual format in Help topics, 5–13

W

Wait cursor
when to use, 1–63
Warning dialog box, 1–36
Widgets
color mixing, 4–5
Digital's additional, 6–1 to 6–16
Print, 6–1
SVN, 6–5 to 6–16
Window decorations, 1–14
Window decorations on dialog boxes, 1–14
Window decorations on primary and secondary windows, 1–14
Window margins
size of, 1–42
Window menu button, when to have, 1–14
Window menu on a dialog box, 1–14
Windows
customizing positions of, 3–8
Windows menu, 2–29 to 2–30
Working dialog box, 1–37
when to use, 1–63
Writing help, 5–1 to 5–14

X

X11/cursorfont.h, 1–5
XDefineCursor, 1–6
XmBulletinBoard, 1–13, 1–17
XmBulletinBoardDialog, 1–8
Xm/decwcursor.h, 1–5
XmDialogStyle, 1–3
XmDIALOG_FULL_APPLICATION_MODAL, 1–3, 1–33
XmDIALOG_MODELESS, 1–3
XmDIALOG_PRIMARY_APPLICATION_MODAL, 1–3
XmDIALOG_SYSTEM_MODAL, 1–3
XmErrorDialog, 1–32
XmFormDialog, 1–8
XmInformationDialog, 1–32
XmNcancelButton, 1–22
XmNdefaultButton, 1–22
XmNdialogStyle, 1–33
XmNdialogTitle, 1–16
XmNleftOffset, 1–25
XmNnoResize, 1–17
XmNrightOffset, 1–26
XmNshadowThickness, 1–44
XmNshadowType, 1–39
XmNtearoffModel, 2–13
XmNtopOffset, 1–46
XmNunitType, 1–42
XmQuestionDialog, 1–32

XmSHADOW_IN, 1–39
XmWarningDialog, 1–32
XmWorkingDialog, 1–32
Xm_ATTACH_OPPOSITE_WIDGET, 1–46
XtManageChild, 1–9
XtUnmanageChild, 1–9
XUndefineCursor, 1–6