

Visual Edge™



Using UIM/X™ PDS

Important Notice

Visual Edge Software Ltd. provides this publication “as is” without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability or fitness for a particular purpose. Some states or jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

Visual Edge Software Ltd. may revise this publication from time to time without notice. While all care has been taken to ensure the accuracy of this publication, Visual Edge Software Ltd. assumes no liability for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

UIM/X

Release 3.0

Document Version 07.97

Copyright © 1997 by Visual Edge Software Ltd. All rights reserved. No part of this work covered by the copyrights herein may be reproduced or copied in any form or by any means—electronic, graphic, or mechanical, including photocopying, recording, taping, or information and retrieval systems—without written permission.

Trademarks

Visual Edge, UIM/X, and GUI Builder Engine are trademarks of Visual Edge Software Ltd.

Motif is a trademark of Open Software Foundation, Inc.

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

X/Open is a trademark of X/Open Company Limited in the UK and other countries.

X Window System is a trademark of the Massachusetts Institute of Technology.

XRT, XRT PDS, XRT/3d, XRT/field, XRT/gear, XRT/graph, and XRT/table are trademarks of KL Group Inc.

Visual Edge Software Ltd.

3950 Côte Vertu	19925 Stevens Creek Blvd.
St. Laurent, Québec	Suite 122
Canada H4R 1V4	Cupertino, CA 95014
Tel: (514) 332-6430	Tel: (408) 973-7823
Fax: (514) 332-5914	Fax: (408) 973-7250

Contents

Preface.....	v
1. Welcome to UIM/X PDS.....	1
UIM/X 3.0.....	1
XRT PDS.....	3
UIM/X 3.0 PDS.....	11
Running UIM/X 3.0 PDS.....	12
2. XRT PDS Widgets.....	13
The Default Palette.....	13
XRT PDS Widgets.....	15
Printing a Selected Interface.....	21
3. Learning About XRT PDS.....	25
XRTHOME Install Directory.....	26
Precompiled Demo Programs.....	27
Demo and Example Source Code.....	27
Documentation.....	33

4. Building a UIM/X PDS To Do List	35
The GUI You Will Build	37
The Steps in Building the To Do List	38
Step #1: Starting UIM/X PDS	39
Step #2: Loading the Start-Up Project	41
Step #3: Testing the Start-Up Project	43
Step #4: Replacing the List Area Widgets	44
Step #5: Saving Your Work	55
Step #6: Setting New List Area Properties	56
Step #7: Replacing the Edit Area Widgets	71
Step #8: Setting New Edit Area Properties	77
Step #9: Adding Behavior to the Edit and Delete Push Buttons	82
Step #10: Adding Behavior to the Tool Bar	87
Step #11: Modifying the To Do List Menus	93
Step #12: Testing the To Do List	97
Step #13: Generating Code	98
Index	101

Preface



UIM/X is the market-leading graphical user interface (GUI) builder for UNIX, combining productivity-enhancing features such as ease of use, specialized editors, and, most importantly, an embedded C++ interpreter.

The XRT Professional Developer's Suite (PDS) is an easy-to-use, fully interactive collection of five widget products for Motif that is completely integrated into UIM/X. Using XRT PDS, you can create dynamic 2-D and 3-D charts and graphs, build professional data entry fields, and manage and display text, images, and widgets in tabular data.

Using UIM/X and XRT PDS together, you can reduce the amount of time and effort spent writing original software, and devote more attention to creating informative, dynamic, and highly-polished interfaces.

This manual discusses the special issues concerned with running UIM/X PDS, explains how to use XRT PDS, and provides a comprehensive tutorial using XRT PDS widgets.

Who Should Use this Guide

Though primarily intended for those users who are evaluating UIM/X PDS, this manual also serves as a guide for users wishing to learn how to use XRT PDS effectively within UIM/X.

This manual assumes you are familiar with the basics of UIM/X. It also assumes that you have some knowledge of programming and a general understanding of the X Window System. You should also know how to use

common items such as menus, buttons, and scroll bars. If you are not familiar with these items, you may find it useful to review the *OSF/Motif User's Guide* and the *UIM/X Motif Developer's Guide*.

Before you begin, check with your system administrator to ensure that the software has been installed as described in the *UIM/X PDS Installation Guide*.

Before You Read this Guide

This guide makes the following assumptions:

- You are familiar with the basic functions of selecting from menus and dialog boxes; opening, moving, resizing and closing windows, and clicking icons.
- You understand the functions of the three mouse buttons, which this guide refers to as the Select button (left button), the Adjust button (middle), and the Menu button (right). See “Using the Mouse” on page x.

The UIM/X Document Set and Related Books

This section lists the UIM/X document set, and provides a suggested list for further reading.

The following list is the complete UIM/X PDS document set:

- *Using UIM/X PDS* (this manual). Explains the special issues concerned with running UIM/X 3.0 PDS.
- *UIM/X PDS Installation Guide*. Explains how to install and run UIM/X. Includes information on the files provided with UIM/X, backwards compatibility issues, and compiler considerations.
- *UIM/X Beginner's Guide*. Introduces UIM/X by presenting Novice Mode, the simplified Palette that enables new users to be productive immediately. Includes information on a number of important features for creating, testing and running applications.
- *UIM/X Tutorial Guide*. A series of step-by-step tutorials, teaching tools and techniques that will greatly assist you in developing your own applications. Features tutorials in Novice Mode, Standard Mode, and on advanced topics.
- *UIM/X User's Guide*. Explores the UIM/X features common to both Motif and cross-platform development. Includes discussions of how to use UIM/X's editors to set properties, add behavior, etc.

-
- *UIM/X Motif Developer's Guide*. An in-depth guide to the widgets, features and capabilities of UIM/X as they relate specifically to Motif development.
 - *UIM/X Advanced Topics*. Describes how to customize UIM/X, including integrating new widget and component classes into the executable. Includes reference information of an advanced technical nature.
 - *UIM/X Reference Manual*. A comprehensive list of properties, methods, and events, plus more, for Motif development. Designed for the experienced developer.

Suggested Reading

For more information on designing GUIs, see any of the following books:

- *OSF/Motif Style Guide release 1.2*
(Prentice Hall, 1993, ISBN 0-13-643123-2)
- *Visual Design with OSF/Motif*
(by Shiz Kobara, Addison-Wesley, 1991, ISBN 0-201-56320-7)
- *New Windows Interface: An Application Guide*
(Microsoft Corporation, 1994, ISBN 1-55615-679-0)
- *Human Interface Guidelines: The Apple Desktop Interface*
(Addison-Wesley, 1987, ISBN 0-201-17753-6)

The XRT PDS Document Set

The following list is the complete XRT PDS document set:

- *XRT/3d™ for Motif Programmer's Guide & Reference Manual, UNIX Edition Version 2.1.1*
(KL Group Inc., RefNo: PRGDE-3D/M/211-03/97)
- *XRT/field™ for Motif Programmer's Guide & Reference Manual, UNIX Edition Version 1.1*
(KL Group Inc., RefNo: PRGDE-FIELD/M/110-03/97)
- *XRT/gear™ for Motif Programmer's Guide & Reference Manual, UNIX Edition Version 2.0*
(KL Group Inc., RefNo: PRGDE-GEAR/M/20-11/96)

-
- *XRT/graph™ for Motif Programmer's Guide & Reference Manual, UNIX Edition Version 3.0*
(KL Group Inc., RefNo: PRGDE-GRAPH/M/300-03/97)
 - *XRT/table™ for Motif Programmer's Guide & Reference Manual, UNIX Edition Version 2.2.1*
(KL Group Inc., RefNo: PRGDE-TABLE/M/221-03/97)
 - *MVC Kit for XRT Widgets Programmer's Guide & Reference Manual, UNIX Edition Version 1.0*
(KL Group Inc., RefNo: PRGDE-MVC/M/100-03/97)

How this Guide Is Organized

Chapter 1, "Welcome to UIM/X PDS" introduces the XRT PDS widgets and covers the basic concepts of starting and running UIM/X.

Chapter 2, "XRT PDS Widgets" describes the XRT PDS Widgets category of the XRT PDS Motif Palette.

Chapter 3, "Learning About XRT PDS" provides examples and demos using the XRT PDS widgets.

Chapter 4, "Building a UIM/X PDS To Do List" provides a comprehensive tutorial using XRT PDS and Motif widgets.

Index, a comprehensive index.

Some Terms You Should Know

Certain basic terms recur throughout this guide, and it helps to understand them from the outset.

An *object* is a building block you can use to build an interface with UIM/X.

A *Motif widget* is an object whose appearance and behavior precisely follows the *OSF/Motif Style Guide*. The novice mode of UIM/X supports a number of popular Motif widgets, including Push Button, Label, Text Field, and more.

A *compound object* consists of several Motif widgets combined into one object for your convenience. The novice mode of UIM/X supports a number of compound objects, including Application Window and Group Box, that save you the time you might otherwise spend creating them.

An *interface* is a window or dialog box that you build up from objects with UIM/X. The novice mode of UIM/X supports four different types of interfaces: Application Window, Secondary Window, Message dialog box, and File Selection dialog box. Certain menu options refer to an interface, such as Save Interface; these act only on your selected interface.

A *project* contains all the interfaces (i.e., windows and dialog boxes) and their associated files for a certain GUI you are building with UIM/X. The program can automatically save and generate code for an entire project in one step. Certain menu options refer to a project, such as Save Project; these act on all the windows and dialog boxes in your project.

Conventions Used in this Guide

Typographic Conventions

The following table describes the typographic conventions used in this guide.

Typeface or Symbol	Meaning	Example
AaBbCc12	The names of commands, files, and directories; or onscreen output; or user input.	Edit your <code>.login</code> file. %You have mail. Use <code>ls -a</code> to list all the files.
<i>AaBbCc12</i>	A placeholder you replace with your actual value; or words to be emphasized; or book titles.	To delete a file, type <code>rm filename</code> . You <i>must</i> be <code>root</code> to do this. See Chapter 6 in the <i>User's Guide</i> .
File⇒Open	The Open option in the File menu.	Choose the File⇒Open command.
Alt+F4	Press both Alt and F4 at once.	Press Alt+F4 to exit.
Return	The key on your keyboard marked Enter, Return, or ↵.	Press Return.

Installation Directories

Product installation directories can depend on the platform or the user's preferences. To keep things simple, this guide uses general names for product installation directories. The following table lists the name and the corresponding product installation directory:

Name	Description
<i>uimx_directory</i>	The UIM/X installation directory.
<i>XRTHOME</i>	The XRT PDS installation directory (always defined as <i>uimx_directory/xrt</i>).

Using the Mouse

Before starting the tutorial, take a moment to review the location and usage of your mouse buttons, as illustrated in Figure P-1 and the following table:

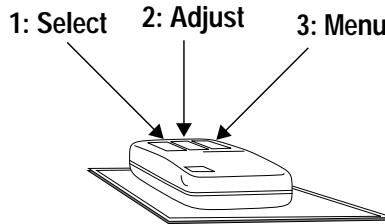


Figure P-1 The Mouse Buttons

Button:	Called:	Is used for:
1	Select	Selecting objects, menus, toggles, and options.
2	Adjust	Resizing and moving objects.
3	Menu	Displaying popup menus.

Throughout this book, you use the mouse buttons along with the mouse pointer to make selections, move the input pointer, or position the text insertion point. You can perform any of the following mouse operations.

Operation	Description
Point to	Move the mouse to make the pointer go as directed.
Press	Hold down a mouse button.
Release	Release a mouse button after pressing it.
Click	Quickly press and release a mouse button without moving the mouse.
Drag	Move the mouse while pressing a mouse button.
Double-click	Click a mouse button twice in rapid succession without moving the mouse pointer.
Triple-click	Click a mouse button three times in rapid succession without moving the mouse pointer.

In general, instructions for mouse operations include the name of the mouse button. The exceptions are Click, Double-click, and Drag. These common operations may be described without specifying a mouse button. For example:

- Click on the `appShell1` icon in the Interfaces Area of the Project Window.
- Drag the Push Button icon from the Palette.

In these cases, use the Select button to click and double-click, and the Adjust button to drag.

Setting Application Defaults

Application Defaults configure the way UIM/X looks and set the default preferences for many of its operations. You can set the Application Defaults for all UIM/X users or for a single user. For more details on setting your Application Defaults see the *UIM/X User's Guide*.

For optimum performance, set the following resources in your Application Defaults.

```
Mwm*autoKeyFocus: false
Mwm*clientAutoPlace: false
Mwm*focusAutoRaise: false
Mwm*focusFollowsPointer: true
Mwm*keyboardFocusPolicy: pointer
```

If you have a gray-scale monitor, you might try the following settings:

```
Mwm*activeBackground: #666666 (gray40)
Mwm*activeForeground: #e5e5e5 (gray90)
Mwm*background: #666666 (gray40)
Mwm*foreground: #e5e5e5 (gray90)
Uimx3_0*calculatedColors: false
Uimx3_0*background: #ededed (gray93)
Uimx3_0*BottomShadowColor: #000000 (black)
Uimx3_0*foreground: #000000 (black)
Uimx3_0*TopShadowColor: #ffffff (white)
Uimx3_0*XmText.background: #b3b3b3 (gray70)
Uimx3_0*XmTextField.background: #b3b3b3 (gray70)
```

Note – The resources above prefixed with `Mwm` are specific to the Motif Window Manager. If you are using a different window manager consult your Systems Administrator for the equivalent settings.

UIM/X 3.0

UIM/X is the market-leading graphical user interface (GUI) builder for UNIX, combining productivity-enhancing features such as ease of use, specialized editors, and, most importantly, an embedded C++ interpreter.

The XRT Professional Developer's Suite (PDS) is an easy-to-use, fully interactive collection of five widget products for Motif that is completely integrated into UIM/X. Using XRT PDS, you can create dynamic 2-D and 3-D charts and graphs, build professional data entry fields, and manage and display text, images, and widgets in tabular data.

Using UIM/X and XRT PDS together, you can reduce the amount of time and effort spent writing original software, and devote more attention to creating informative, dynamic, and highly-polished interfaces.

UIM/X 3.0 is a major functional upgrade to UIM/X. It preserves all the productivity-enhancing features inherent to UIM/X, but now includes several new editors, ease-of-use enhancements, and a C++ interpreter.

UIM/X 3.0's new features enable developers to work entirely in C++ or C, or to mix and match the two languages. This allows users to make the transition from C to C++ at their own pace. In this way, UIM/X ensures that previous investment in GUI development is leveraged to the greatest extent possible.

Other new features include:

- **Connection Editor**
UIM/X 3.0 contains a new editor that allows users to visually connect and define application behavior.
- **Constraint Editor**
UIM/X 3.0 contains a new editor that enables users to visually set Form constraints for Motif interfaces.
- **C/C++ Method Editor**
To define interface behavior, UIM/X includes a Method Editor. Using this editor, developers can create methods that contain behavior specific to that interface. Once defined, users can access the methods using the Connection Editor to graphically connect interfaces together.
- **Enhanced User Interface**
UIM/X 3.0 contains an enhanced user interface that makes it even easier to use. Every interface now contains an iconic toolbar, which improves developer productivity by providing quick shortcuts to all common operations. The new Bubble Help system offers useful reminders about icon definitions, and can be adjusted to fit novice as well as advanced users. Moreover, UIM/X's menus have been reorganized to provide convenient and consistent access to other commonly used editors.
- **“Non-Visual” Objects**
New in UIM/X 3.0 is the ability to define and manipulate non-visual objects—objects that do not necessarily have a visible user interface. This allows users to build and share application level classes as if they were GUI objects.
- **Novice Mode**
To help new users, Novice Mode provides a simplified set of pre-configured objects and tools for building interfaces; moreover, it exposes only a limited number of the properties and callbacks available for each object. As users expand their knowledge of Motif and UIM/X, they can progress from Novice Mode to standard UIM/X, while fully preserving their previous work. In Novice Mode, users can build useful, working applications with virtually no learning curve.

XRT PDS

XRT Professional Developer's Suite (PDS) integrates five Motif widget products into a single, easy-to-use suite. These widgets are described in the following paragraphs.

XRT/3d

XRT/3d allows you to build dynamic 3-D charts and graphs into Motif applications. XRT/3d creates 3-D surface plots, contour graphs and bar charts from your data, and automatically performs all rotation, scaling, annotation and perspective calculations.

XRT/3d has resources that determine how the graph will look and behave. Writing programs using XRT/3d is very similar to writing any other kind of Motif program; you now have one more Motif widget to use.

XRT/3d may be used in conjunction with the XRT/graph and XRT/table widgets. XRT/graph displays 2D data in X-Y plots, bar and pie charts. XRT/table displays rows and columns of tabular information in a scrollable window.

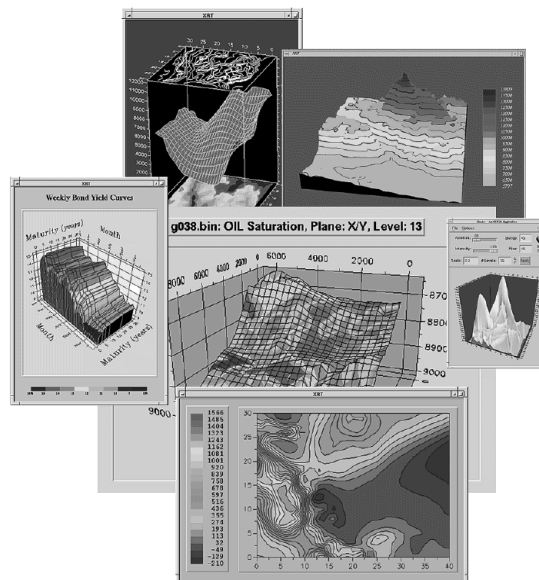


Figure 1-1 XRT/3d Widget

XRT/3d has resources which allow control of:

- Drawing (whether to draw the mesh, surface, contours and/or zones) and graph type (surface or bar chart).
- Contour styles: line widths, colors and patterns, and fill colors.
- Distribution method, distribution table, number of distribution levels.
- Hidden-line display, surface colors and mesh colors.
- 3D rotation and perspective.
- Legend positioning, orientation, style, border style, anchor, font and color.
- Header and footer positioning, border style, text, font and color.
- Graph positioning, border style, color, width, and height.
- Axis maximums and minimums, fonts, gridlines, titles and labels.
- Window background and foreground color.

XRT/3d also provides several procedures and methods which:

- Allocate and load data structures containing the numbers to be graphed.
- Output a representation of the graph in PostScript or CGM format.
- Assist the developer in dealing with user-events.
- Assist the developer in dealing with interactive rotation, zooming and shifting of the 3D view by the end-user.
- Assist the developer with setting and getting indexed resources.

XRT/field

XRT/field widgets store text and allow you to convert, complete and validate the text once it is stored. They provide an easy way to enter and verify data in Motif applications that require intensive data entry, such as database applications. Additional widgets allow you to create spin boxes and combo boxes that contain and manipulate XRT/field widgets. Each field widget has resources that determine how the field will look and behave.

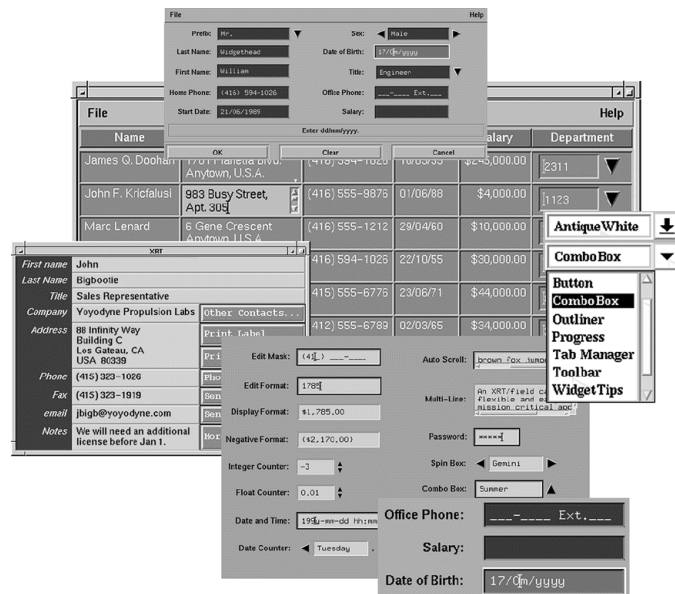


Figure 1-2 XRT/field Widget

The XRT/field widget family consists of three widget classes:

- XrtField and its subclasses, which store and manipulate text entered by the user.
- XrtComboBox, which allows users to choose an item from a pick list.
- XrtSpinBox, which can be used to increment numeric values or cycle through a pick list.

The XrtField widget class is defined as a subclass of the Motif XmText widget. This means that XRT/field widgets can be plugged directly into any application that uses text widgets, including applications that use the XRT/table widget. Similarly, the XrtComboBox and XrtSpinBox widgets are defined as subclasses of XmXrtFldManager, a subclass of the XmManager widget.

The XrtField widget class provides the following capabilities:

- Validation: data can be validated using masks, data types, value ranges, or pick lists.

- Conversion: values can be converted using callback routines.
- Completion: partial values can be completed using callback routines or pick lists.
- Format specification: you can specify the edit and display format for the value.
- Enter, exit, validation, and error-detection callback routines can be used by applications.
- You can specify which input characters are to be treated as valid or invalid.
- Built-in scrollbars can be displayed as needed.

XrtField widgets also inherit the features of the XmText widget.

An XrtComboBox widget can control the placement and widget class of its child arrow widget.

XRT/gear

XRT/gear is an integrated collection of add-on widgets for Motif that allows you to build improved Motif user interfaces. This set consists of the following:

- *Enhanced Pushbutton* and *Enhanced Label*, which provide the features of the Motif PushButton and Label widgets plus additional capabilities.
- *Enhanced Toggle*, which provides the features of the standard Motif ToggleButton widget plus additional capabilities.
- *Tab Manager* and *Tab Button*, which enable you to store and display your application widgets in a sequence of file-folders.
- *Toolbar*, which is a way of containing a row of buttons or other widgets, providing an alternative to the standard Motif menu bar.
- *Aligner*, which enables you to create and align columns of label-widget pairs.
- *Outliner*, which enables you to display data organized in a hierarchical fashion.
- *Progress*, which enables you to update the end-user on the status of a task.
- *Combo Box*, which provides a convenient way to enable an end-user to choose from a list of alternatives.

XRT/gear also includes the following utilities:

- *Widget Tips*, which enables you to add a pop-up help balloon to any widget in your application.
- *Widget Snapshot*, which prints the contents of any Motif widget in PostScript format, and enables you to get a pixmap image or XImage of any Motif widget.
- *Icon Library*, which is a set of 3000 public-domain icons suitable for use with any of the XRT/gear widgets. An icon viewer is included.

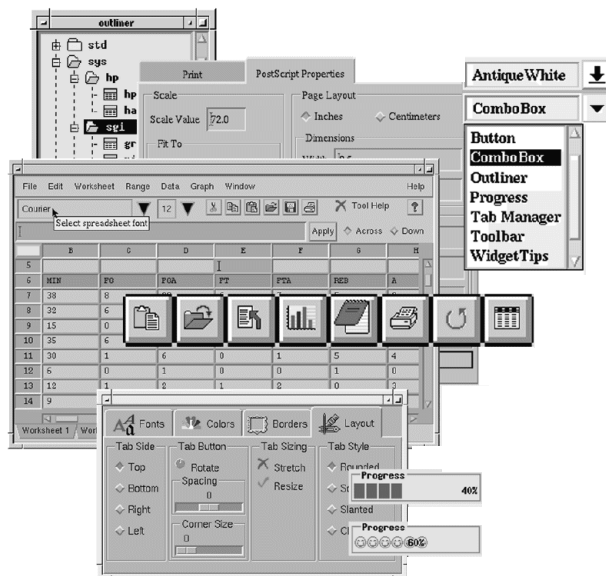


Figure 1-3 XRT/gear Widget

XRT/graph

XRT/graph allows you to build 2-D charts and graphs into Motif applications. Graph formats include bar charts, X-Y plots, pie charts, area graphs, financial graphs, logarithmic scientific charts, and combination charts.

XRT/graph displays data graphically in a window and can interact with a user. The graph widget has resources that determine how the graph will look and behave. Writing programs using XRT/graph is similar to writing any other kind of Motif program; you now have one more Motif widget to work with.

XRT/graph may be used in conjunction with the XRT/3d and XRT/table widgets. XRT/3d displays 3D data in surface, contour, and bar graph representations. XRT/table displays rows and columns of tabular information in a scrollable window.

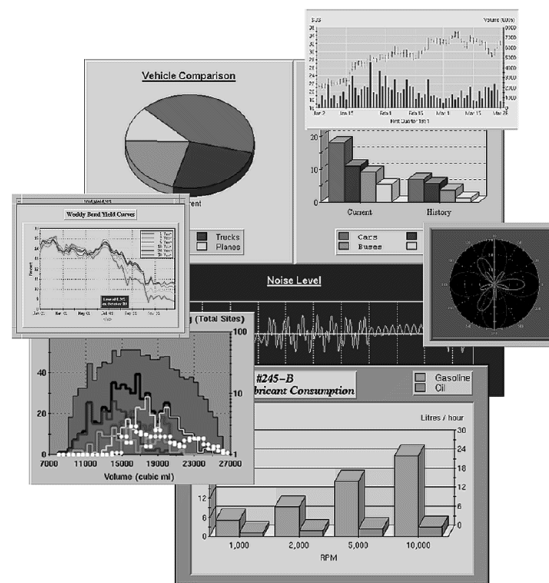


Figure 1-4 XRT/Graph widget

XRT/graph has resources which allow control of:

- Graph type (plot, area, bar, stacking-bar, pie, HiLo, HiLoOpenClose, candle, polar, radar, filled-radar, or combination).
- Header and footer positioning, border style, text, font, and color.
- Data styles: line colors and patterns, fill colors and patterns, line thickness, point style, size, and colors.
- Legend positioning, orientation, border style, anchor, font, and color.
- Graph positioning, border style, color, width, height, and 3D effect.
- Axis labelling using Point-labels, Set-labels, Value-labels, or Time-labels.
- Axis and data minimums and maximums, axis numbering methods, numbering and ticking increments, grid increments, font, origins, axis directions and precisions.

- Placement of axes, annotation, and origins. Inversion of axes.
- Control of user-interaction with the widget using callback resources.
- Markers and Text Areas.

XRT/graph also provides several procedures and methods which:

- Allocate and load data objects containing the numbers to be displayed.
- Update elements and data contained in data objects.
- Display new data very quickly in certain circumstances.
- Enable a property editor that developers or end-users can use to view and change graph resources at run time.
- Output a representation of the graph in PostScript format.
- Assist the developer with setting and getting indexed resources.

XRT/table

XRT/table is a multi-purpose widget for displaying and manipulating tabular information and forms in Motif applications.

XRT/table lets you manage text values, pixmaps, and widgets inside table cells. You can manage any Motif widget, including primitives like buttons, and composites like option menus, forms, bulletin-boards, or other XRT Widgets like graph, tables and data-entry fields.

XRT/table has resources that determine how the table will look and behave. Writing programs using XRT/table is very similar to writing any other kind of Motif program; you now have one more Motif widget to work with.

XRT/table may be used in conjunction with the XRT/graph, XRT/3d and XRT/field widgets. XRT/graph displays 2-D data in X-Y plots, bar and pie charts. XRT/3d displays 3-D data in surface, contour, and bar chart representations. XRT/field stores text and allows you to convert, complete and validate the text once it is stored.

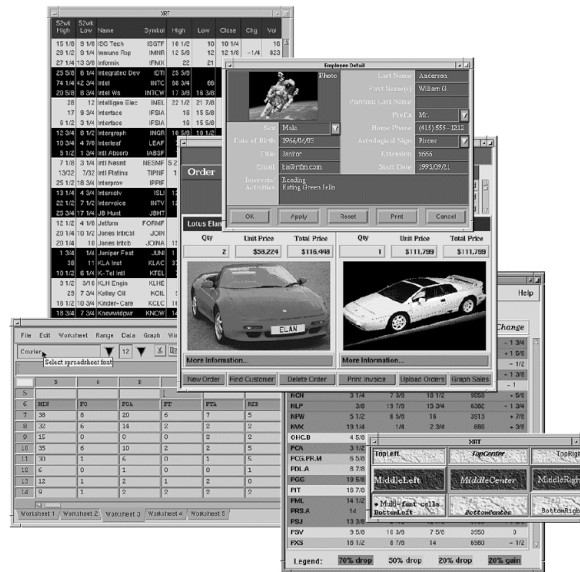


Figure 1-5 XRT/table Widget

XRT/table has resources which allow control of:

- Number of rows and columns, both in the table, and visible on the screen.
- Cell contents: cells and labels can contain text values, pixmaps, or widgets.
- Row height and column width.
- Cell/label attributes including border, text alignment, font, and color.
- User interaction including selection methods.
- Cell editability and traversability.

Compound Strings, which allow cell/label text to use multiple fonts.

XRT/table also provides several procedures and methods which:

- Add, delete, and move table rows and columns.
- Output a representation of the table in PostScript format.

- Read table cell values from an ASCII file, as well as from XRT/graph and XRT/3d datasets.
- Perform data validation of cell values.
- Assist the developer with setting and getting cell ranges.
- Write table cell values to an ASCII file, and in SYLK format.

UIM/X 3.0 PDS

UIM/X 3.0 PDS is a powerful and flexible integration of UIM/X with the XRT Professional Developer's Suite of widgets for Motif.

UIM/X PDS's new features and unique architectural approach has made it the most widely used GUI builder for UNIX. The UIM/X PDS core architecture embodies a chassis design that provides developers the greatest flexibility in customizing, extending or "shrinking" the tool to address their particular needs. The Builder Engine offers comprehensive access to UIM/X PDS's internal API, which enables developers to easily extend UIM/X PDS with foreign components, or allows them to turn-off and hide any of UIM/X PDS's features and functions. In short, the Builder Engine functionality allows developers to "build their own builder".

Combining the power of UIM/X's unique Builder Engine architecture with the fully interactive and easy-to-use widgets offered by XRT PDS allows you to build your own unified, custom development environment from among the best technology the industry has to offer.

Running UIM/X 3.0 PDS

Before running UIM/X PDS, you must ensure that the `XRTHOME` environment variable is defined properly, and that `$XRTHOME/bin` is put in the search path.

To do so, you can type the following two lines at the command-line level, or add them to your `.cshrc` file:

```
setenv XRTHOME xrt_directory
set path=($XRTHOME/bin $path)
```

The `xrt` directory is a subdirectory of `uimx_directory`. It is in this directory that the entire distribution of XRT PDS software is installed.

Once you have defined the `XRTHOME` and `PATH` environment variables, you can proceed to run UIM/X PDS in the usual fashion, that is, type:

```
cd uimx_directory
bin/uimx
```

or

```
uimx_directory/bin/uimx
```

After a brief pause, a copyright notice window appears on the screen, indicating that UIM/X PDS is being initialized. The Project Window and the default UIM/X PDS palette appear and the copyright notice disappears from view.

Using UIM/X PDS is the same as using regular UIM/X, with the following exceptions and enhancements:

- The default Palette displayed when starting UIM/X PDS is augmented.
- XRT PDS Widgets are provided.
- A new File menu option for printing interfaces is provided.

These issues are discussed in the remainder of this chapter.

The Default Palette

The X RTPDS Motif Palette, shown in Figure 2-1, is the default palette displayed when you start UIM/X PDS. Its filename is `X RTPDS Motif .pal` and it is found in `uimx_directory/palettes`. This palette contains a new category called XRT PDS Widgets, which are described in detail later in this chapter.

You can change the default palette that UIM/X PDS displays on start-up by using resources. The UIM/X PDS resource file contains the following resource specification:

```
Uimx3_0*UxStartingPalettes.value: X RTPDS Motif .pal
```

The `UxStartingPalettes` resource specifies the palette files loaded at start-up. These palette files are loaded before any files specified on the UIM/X PDS command line. These palettes are not saved with projects saved by the user.

If you wish to have UIM/X PDS start with the Ux Palette, with no XRT PDS Widgets category, change the UxStartingPalettes resource to look as follows:

```
Uimx3_0*UxStartingPalettes.value: Ux.pal
```

Note - You can load more than one palette file at once. To load more than one palette, you must insert \n between palette file names. For more information about working with palettes, see the *UIM/X User's Guide*.

The XRTPDSMotif Palette is shown below. Most categories have been collapsed for display purposes:

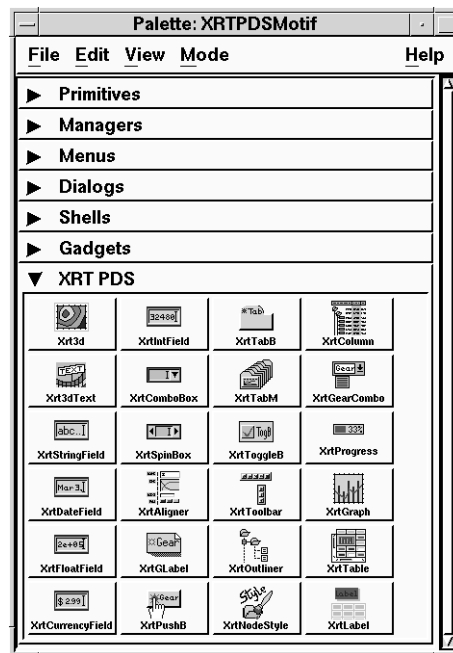


Figure 2-1 The XRTPDSMotif Palette, Showing the XRT PDS Widgets Category

XRT PDS Widgets

The XRT PDS Widgets provided by the XRTPDSMotif Palette are shown in the following table, along with their names, suggested uses, and how the end user activates them in your interface.

Table 2-1 XRT PDS Widgets (Sheet 1 of 7)

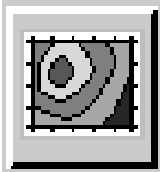
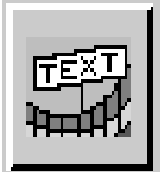
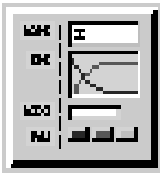
Name	Icon	Suggested Uses
Xrt3d		<p>Purpose: to display 3d information in bar chart, contour or surface graphs.</p> <p>Use an Xrt3d object when you want to present graphical data in a 3-dimensional format.</p> <p>User can interactively rotate, zoom, and move the graphic it contains.</p>
Xrt3dText		<p>Purpose: to attach text annotations to a surface or bar chart.</p> <p>Use an Xrt3dText to attach an annotation to a graph in one of three ways: to a particular grid index, to a point in 3D space, or to a set of pixel coordinates on the widget.</p> <p>An Xrt3dText cannot be activated.</p>
XrtAligner		<p>Purpose: to arrange a group of objects in a vertical column with associated labels.</p> <p>Use an XrtAligner to create and align columns of label-widget pairs. Controllable resources include margins, spacing and orientation.</p> <p>An XrtAligner cannot be activated.</p>

Table 2-1 XRT PDS Widgets (Sheet 2 of 7)

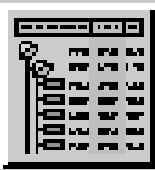
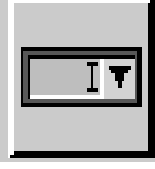

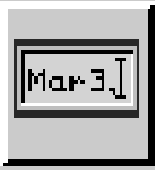
Name	Icon	Suggested Uses
XrtColumn		<p>Purpose: to add a column to an XrtOutliner object.</p> <p>Use an XrtColumn to add a column to an outline, for example, to add a date column to a list of files or directories.</p> <p>An XrtColumn cannot be activated.</p>
XrtComboBox		<p>Purpose: to enable end users to select from a list of alternatives.</p> <p>Use an XrtComboBox to enable the end user to choose from a list of alternatives.</p> <p>Activate an XrtComboBox by pressing and holding its dropdown button, then choosing from the list.</p>
XrtCurrencyField		<p>Purpose: to accept and display floating-point numbers plus an optional currency symbol.</p> <p>Use an XrtCurrencyField to display numerical data using a currency format.</p> <p>Activate an XrtCurrencyField by clicking within it and typing a value.</p>
XrtDateField		<p>Purpose: to accept and display dates and times in a variety of pre-specified or program-specified formats.</p> <p>Use an XrtDateField to display a date.</p> <p>Activate an XrtDateField by clicking within it and typing a value.</p>

Table 2-1 XRT PDS Widgets (Sheet 3 of 7)

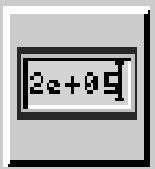


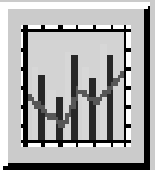
Name	Icon	Suggested Uses
XrtFloatField		<p>Purpose: to display and accept floating point numerical data.</p> <p>Use an XrtFloatField to display data in fixed or scientific format. XrtFloatField accepts digits, a decimal point, an exponent, and “+” or “-”.</p> <p>Activate an XrtFloatField by clicking within it and typing a value.</p>
XrtGearCombo		<p>Purpose: to enable end users to select from a list of alternatives, with validation.</p> <p>Use an XrtGearCombo to enable the end user to choose from a list of alternatives.</p> <p>Activate an XrtGearCombo by clicking its dropdown button, then clicking an item from the pick list.</p>
XrtGLabel		<p>Purpose: a Label widget that provides the ability to combine pixmaps with text labels.</p> <p>Use an XrtGLabel to customized an Enhanced Label or Enhanced Pushbutton by controlling such features as pixmap, spacing, layout, rotation, borders, and clipping.</p> <p>An XrtGLabel cannot be activated.</p>
XrtGraph		<p>Purpose: to display data graphically in a window, according to any one of a set of graph formats.</p> <p>Use an XrtGraph to display and update elements and data contained in data objects.</p> <p>User can interactively rotate, zoom, and move the graphic it contains.</p>

Table 2-1 XRT PDS Widgets (Sheet 4 of 7)

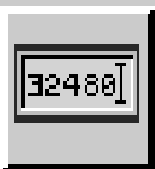
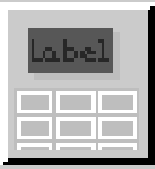

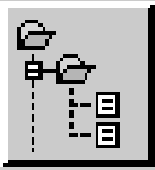
Name	Icon	Suggested Uses
XrtIntField		<p>Purpose: to display and accept numerical data using integer format.</p> <p>Use an XrtIntField to display an integer value. XrtIntField accepts digits and a leading “+” or “-” sign, and can handle octal, hexadecimal and binary formats (specified by a leading 0o, 0x or 0b).</p> <p>Activate an XrtIntField by clicking within it and typing a value.</p>
XrtLabel		<p>Purpose: To display text or pixmaps on an XrtTable.</p> <p>Use an XrtLabel to provide an identifying name or icon to part of your XrtTable.</p> <p>An XrtLabel cannot be activated.</p>
XrtNodeStyle		<p>Purpose: to define a specific style for a node in an XrtOutliner object.</p> <p>Use an XrtNodeStyle to define the appearance and behavior of nodes being displayed in an outline. Using an XrtNodeStyle, you can control such resources as color, font, icon spacing, icon pixmaps, connecting lines, and shortcut buttons.</p> <p>An XrtNodeStyle cannot be activated.</p>
XrtOutliner		<p>Purpose: to provide a means to display data in a graphically organized, hierarchical fashion.</p> <p>Use an XrtOutliner as an easy means to view the relationships between given items.</p> <p>Activate an XrtOutliner by clicking its lines and nodes.</p>

Table 2-1 XRT PDS Widgets (Sheet 5 of 7)

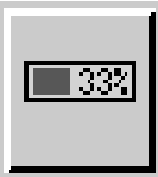
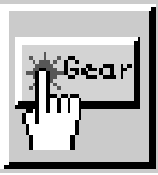
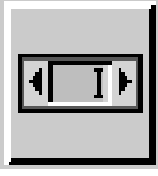
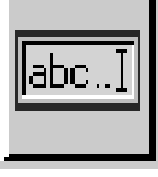
Name	Icon	Suggested Uses
XrtProgress		<p>Purpose: to keep the user informed of the status of a task.</p> <p>Use an XrtProgress to let the user follow the progress of an action.</p> <p>An XrtProgress cannot be activated.</p>
XrtPushB		<p>Purpose: to provide the features of the standard PushButton widget plus the ability to combine pixmaps with text labels.</p> <p>Use an XrtPushB to create a customized PushButton by controlling such features as button type, pixmap, spacing, layout, rotation, borders, and clipping.</p> <p>Activate an XrtPushB by clicking it.</p>
XrtSpinBox		<p>Purpose: provides the means to increment or decrement numeric values or cycle through a pick list.</p> <p>Use an XrtSpinBox as a counter, or to select an item from a pick list.</p> <p>Activate an XrtSpinBox by clicking its arrow buttons to cycle through the items in the pick list.</p>
XrtStringField		<p>Purpose: to accept text from the end user.</p> <p>Use an XrtStringField object to accept data in the form of text characters.</p> <p>Activate an XrtStringField by clicking within it, and then typing text.</p>

Table 2-1 XRT PDS Widgets (Sheet 6 of 7)

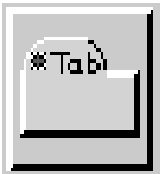
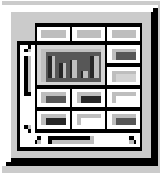
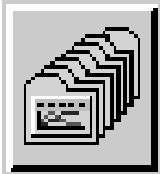
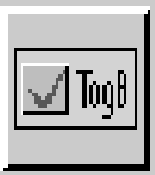
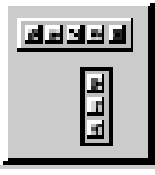
Name	Icon	Suggested Uses
XrtTabB		<p>Purpose: a tab button that enables selection of the file folder where your application widgets are stored.</p> <p>Use an XrtTabB in conjunction with an XrtTabM to store and display your application widgets in a sequence of file folders.</p> <p>Activate an XrtTabB by clicking it.</p>
XrtTable		<p>Purpose: to display rows and columns of text information, pixmaps, and widgets in a scrollable window that can interact with the user.</p> <p>Use an XrtTable to display and allow the user to edit data presented in tabular format.</p> <p>Activate an XrtTable by clicking in a table cell and entering or changing data, or by dragging and dropping data from one cell to another.</p>
XrtTabM		<p>Purpose: a tab manager that enables you to store and display application widgets in a sequence of file-folders.</p> <p>Use an XrtTabM to simulate a sequence of file folders. You can control the location, orientation, style, spacing and rotation of tabs.</p> <p>An XrtTabM cannot be activated.</p>

Table 2-1 XRT PDS Widgets (Sheet 7 of 7)

Name	Icon	Suggested Uses
XrtToggleB		<p>Purpose: an enhanced toggle button that lists a variety of choices, one or more of which can be selected.</p> <p>Use an XrtToggleB object to present a group of choices to the end user. You can specify any number of widget states you like.</p> <p>Activate an XrtToggleB object by clicking it.</p>
XrtToolbar		<p>Purpose: a manager which presents buttons in a single row or column.</p> <p>Use an XrtToolbar as a quick alternative to a menu bar, to present a group of choices to the end user.</p> <p>An XrtToolbar cannot be activated; only the pushbuttons within it can be activated by clicking on them.</p>

Printing a Selected Interface

The File menu of the UIM/X 3.0 PDS Project Window now contains a new option for printing a selected interface.

To print an interface:

1. Select it by clicking on the interface itself or on its icon in the Interfaces area of the Project Window.
2. Choose File⇒Print Selected... from the Project Window menu bar.

A Print Application tab dialog appears, with the Print folder selected, as illustrated in Figure 2-2.

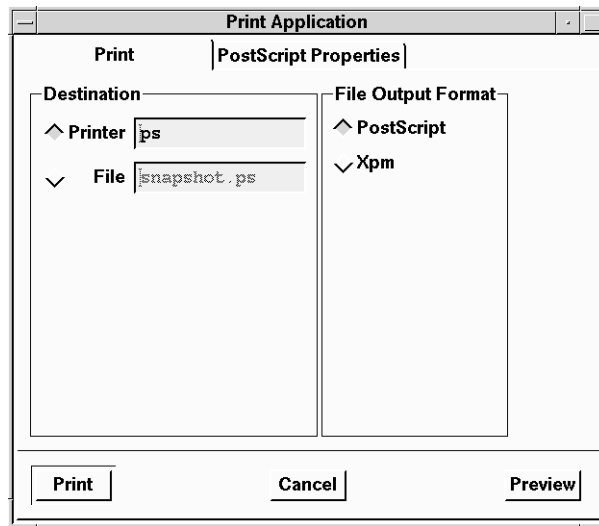


Figure 2-2 Print Folder of Print Application Dialog

From the Print folder, you can select the print destination and file output format, and preview your printed interface in a scrollable and resizeable Preview window.

3. To view the currently selected printing properties, select the PostScript Properties folder by clicking its tab button.

The PostScript Properties folder appears, as illustrated in Figure 2-3.

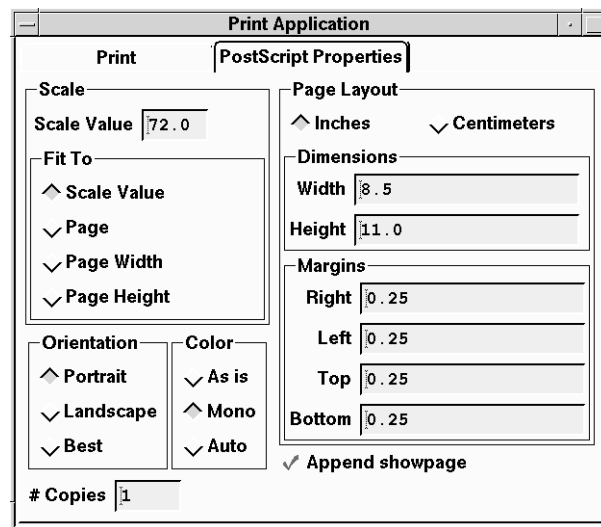



Figure 2-3 Print Properties Folder of Print Application Dialog

From the Print Properties folder, you can select the scale, orientation, color, page layout, and number of copies of your printout.

4. When you are satisfied with the selected PostScript properties, return to the Print folder by clicking its tab button.
5. Select Print to print your interface.

Your printout is sent to the selected printer or file, as you specified.

Learning About XRT PDS

3 

This chapter describes the material provided to assist the developer in learning about XRT PDS, specifically:

- An explanation of the XRTHOME installation directory tree.
- The precompiled demos and examples provided with the XRT PDS installation medium.
- The documentation.

XRTHOME Install Directory

Figure 3-1 illustrates the *XRTHOME* tree and its main subdirectories.

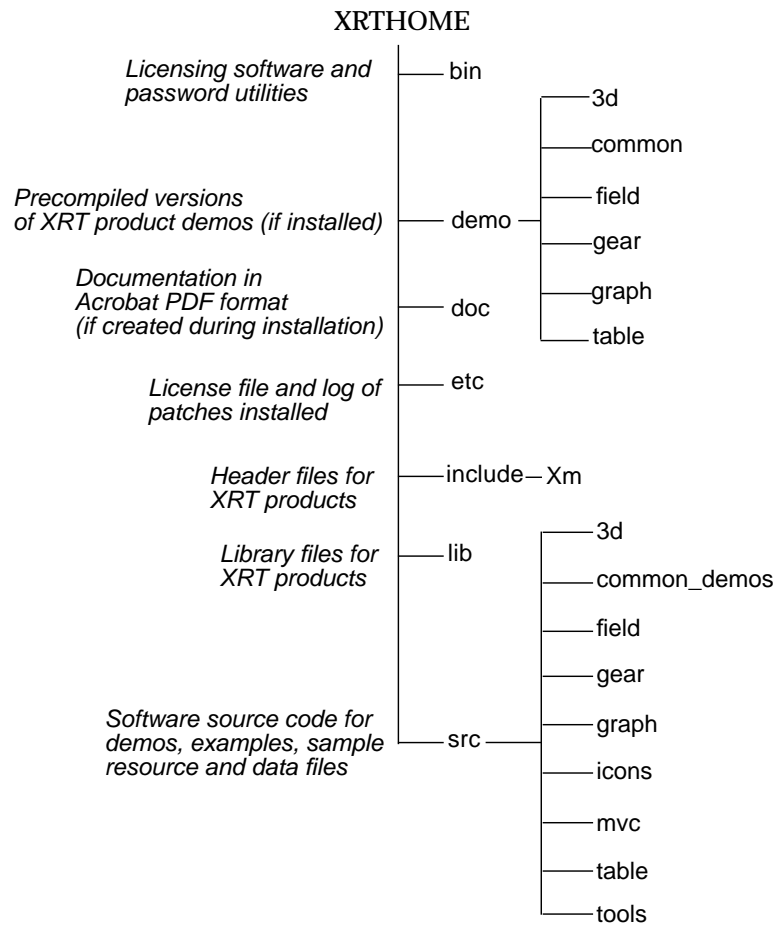


Figure 3-1 The XRTHOME Directory Tree

Precompiled Demo Programs

Once PDS is installed, the easiest way to familiarize yourself with its components is by running the precompiled demo programs.

The precompiled demos are in `$XRTHOME/demo`. Each product has a separate directory for its demos.

Demo and Example Source Code

The best way to start with PDS is simply to use it. Copy the source code for a program to a writable location and *make* it. Assuming `XRTHOME` is set up correctly, the makefile will compile, link, and authorize the application with `xrt_auth`.

PDS products include both examples and demos. Examples are short programs that illustrate specific features. Demos are more complete applications that illustrate several features working together. Most common programming tasks are covered in the examples and demos.

XRT/3d

Examples (located in <i>\$XRTHOME/src/3d/examples</i>)	
<code>bars.c</code>	A simple bar chart that allows the user to change the color of individual bars by clicking on them.
<code>simple.c</code>	A simple 3D surface example program.
<code>simple_cpp.cxx</code>	Implements <code>simple.c</code> in C++.
<code>simple_uil.c</code>	A version of <code>simple.c</code> that uses UIL to set the resources.
<code>texts.c</code>	Shows how to create text objects.
Demos (located in <i>\$XRTHOME/src/3d/demos</i>)	
<code>feedback3d</code>	Demonstrates use of the mapping and picking feedback functions.
<code>math3d</code>	Graphs built-in mathematical functions.
<code>play3d</code>	Provides direct access to all XRT/3d and text object resources. Loads and saves graph descriptions in resource files.
<code>shader</code>	Enhances the visual effect of surfaces by shading them. Shades each facet of a surface according to the angle and intensity of a light source.
<code>spline</code>	Shows how to reduce the size of large datasets using cubic and linear spline interpolation.
<code>view_dir</code>	Shows how to display data as a 3D bar histogram. Gives a visual display of files in a selected directory.

XRT/field

Examples (located in <i>\$XRTHOME/src/field/examples</i>)	
<code>combo.c</code>	Displays a simple <code>XrtComboBox</code> widget.
<code>counter.c</code>	Displays an <code>XrtSpinBox</code> that counts from 0 to 10.
<code>datespin.c</code>	Creates an <code>XrtSpinBox</code> with four counters as children, containing the day of the week, the day of the month, the month, and the year.

Examples (located in <i>\$XRTHOME/src/field/examples</i>)	
<code>fields.c</code>	Displays an example of each of the five <code>XrtField</code> subclasses.
<code>password.c</code>	Uses a field to obtain a user password, which is then written to <code>stdout</code> .
<code>phone.c</code>	Displays a phone number field. The area-code parentheses and dash are automatically supplied by means of a mask.
<code>simple.c</code>	A simple example program.
<code>spin.c</code>	Displays a simple <code>XrtSpinBox</code> widget.

Demos (located in <i>\$XRTHOME/src/field/demos</i>)	
<code>form</code>	Shows how you can use <code>XRT/field</code> widgets to create an employee data entry form.
<code>locale</code>	Contains the <code>printlocale</code> program, which displays local information for a selected locale or the current locale.
<code>sampler</code>	Shows all the features and capabilities of <code>XRT/field</code> . Separate windows (accessible via radio buttons) display collections of fields, spin boxes, combo boxes, counter boxes, and currencies.

XRT/gear

Examples (located in <i>\$XRTHOME/src/gear/examples</i>)	
<code>aligner.c</code>	Displays an <code>Aligner</code> widget.
<code>combo.c</code>	Displays a <code>Combo Box</code> widget.
<code>complex_cpp.cxx</code>	Creates <code>XRT/gear</code> widgets and manipulates their resources; written in C++.
<code>label.c</code>	Displays an <code>Enhanced Label</code> widget.
<code>multicol.c</code>	Displays an <code>Outliner</code> widget containing multiple columns with labels.
<code>outliner.c</code>	Creates an <code>Outliner</code> widget displaying the data in a specified file.
<code>progress.c</code>	Shows the features of the <code>Progress</code> widget.

Examples (located in <i>\$XRTHOME/src/gear/examples</i>)	
<code>pushb.c</code>	Displays an Enhanced Pushbutton widget.
<code>racecar.c</code>	Displays a Progress widget, illustrating the use of icons.
<code>simple_cpp.cxx</code>	Creates XRT/gear widgets; written in C++.
<code>tabs.c</code>	Displays Tab Manager and Tab Button widgets.
<code>toggle.c</code>	Displays an Enhanced Toggle widget.
<code>toolbar.c</code>	Displays a Toolbar widget.

Demos (located in <i>\$XRTHOME/src/gear/demos</i>)	
<code>fileviewer</code>	Uses the Outliner widget to display and modify your system file hierarchy.
<code>sampler</code>	Shows how many of the XRT/gear widgets work.

XRT/graph

Examples (located in <i>\$XRTHOME/src/graph/examples</i>)	
<code>bar2tran.c</code>	Displays a simple transposed bar chart.
<code>bar2tram_cpp.cxx</code>	Implements <code>bar2tran.c</code> in c++.
<code>common.c</code>	
<code>panel.c</code>	Defines a form containing several Motif toggle widgets. A graph is created below the toggles.
<code>plot1.c</code>	A simple example program.
<code>radar.c</code>	Displays a simple radar chart.
<code>scrollgraph.c</code>	Displays a graph in a scrolled window widget.
<code>simple_ui1.c</code>	A simple graph program that uses UIL to set resources.
<code>slider.c</code>	Shows how to use graph axes and datastyles to simulate a slider user interface.
<code>stock.c</code>	Shows HiLo and Candle charts.
<code>time.c</code>	A simple example of using time-axis label.

Demos (located in <i>\$XRTHOME/src/graph/demos</i>)	
<code>builder</code>	Uses the Property Editor to provide a simple environment for interactively building the visual look of a graph or viewing the sample <code>.XRT</code> graph description files.
<code>drill_down</code>	Shows one way to manage and present a large amount of data at different levels of granularity.
<code>feedback</code>	Demonstrates some techniques for mapping and picking, and for displaying visual feedback of those events. Also illustrates one way to use markers.
<code>flow</code>	Demonstrates a type of pulse chart, and is a good example of using color to differentiate ranges of data values.
<code>hilow</code>	Plots the open, high, low, and close prices for two stocks. Illustrates one way to manipulate data objects (on the fly) as a user requests various options.
<code>polar</code>	Shows different ways to display polar data using the polar resources. Uses a number of mathematical functions to generate the data.
<code>scroll_time</code>	Demonstrates usage of the time-axis features and shows the use of <code>Xlib</code> routines to select a subset of data to display.
<code>stripper</code>	Demonstrates how to use the Fast Update procedures to build a real-time strip chart. You can adjust the rate of incoming data and the scroll-back percentage.
<code>text_area</code>	Demonstrates how to create and attach text areas to plots, bar graphs, stacking bar graphs, and pie charts.
<code>vmgraph</code>	Displays your system's virtual memory statistics in three graphs that are updated every second. Demonstrates one way to read data from a pipe. Requires the <code>vmstat</code> program, which is not available on OpenVMS or some UNIX systems.

XRT/table

Examples (located in <i>\$XRTHOME</i>/src/table/examples)	
<code>entry.c</code>	Shows the use of the <code>TraverseCell</code> and <code>EnterCell</code> callbacks to control traversal and text entry into a simple form. Also shows the use of buttons and option menus within cells.
<code>list.c</code>	Shows how to mimic an <code>XmScrolledList</code> with a table by setting <code>XmNxrtTblMode</code> . Also shows sorting a table and searching a table for a string.
<code>multiply.c</code>	Shows how to use <code>XmNxrtTblCellValueCallback</code> to specify the value of each cell on demand, an efficient way to handle very large tables.
<code>simple.c</code>	Displays a very basic table that you can use to learn some XRT/table basics.
<code>simple_cpp.cxx</code>	Implements <code>simple.c</code> in C++.
<code>styles_c.c</code>	Shows how to switch between three different table looks with the click of a button.
<code>styles_res.c</code>	A version of <code>styles_c.c</code> that creates all resources from a resource file (<code>styles.res</code>).
<code>styles_util.c</code>	A version of <code>styles_c.c</code> that uses UIL to set all resources.
Demos (located in <i>\$XRTHOME</i>/src/table/demos)	
<code>bombhunt</code>	Uses XRT/table in a simple game. The object is to find all of the bombs hidden under some of the squares.
<code>form</code>	a data-entry form that uses two tables. Shows how to use spanned cells, widgets in cells, and pixmaps in cells.
<code>play</code>	Provides direct access to all XRT/table's resources. A user can change any resource value and immediately see how the changes affect the table. Saves table descriptions in resource files.
<code>stocks</code>	Illustrates run-time setting of colors.
<code>widget</code>	Illustrates how to manage widgets in cells.

Multi-Product Demos

Demos (located in `$XRTHOME/src/common_demos`)

		XRT/3d	XRT/field	XRT/gear	XRT/graph	XRT/table
graph_editor	A table displays data that is graphed. A user can edit the data in the table and the graph updates to reflect the change.				✓	✓
mvc	Demonstrates MVC Kit for XRT widgets.	✓			✓	✓
profile	Displays a 2D profile graph of a 3D surface.	✓			✓	
spreadsheet	Mimics a spreadsheet application, allowing a user to load or enter data into cells. Demonstrates most of XRT/table's features.		✓	✓	✓	✓
surface_editor	A table displays Z-values displayed in a 3D graph. A user can edit the Z-values and the graph updates to reflect the change.	✓				✓
Table_entry	A simple table uses different field subclasses for validating user editing in different columns.		✓			✓

Documentation

PDS includes a complete programming and reference manual for each XRT product. Each manual offers several ways to help you learn quickly how to use the product:

- Product overview, new features, and changes between releases are detailed in each manual's Preface.
- Step-by-step tutorial outlining the basic steps of working with the product is available in the "Getting Started" chapter.
- Important basic information on terminology, widget hierarchy, programming certain types of resources, and more is provided in the "Basics" chapter.

UIM/X provides all the objects and tools needed to develop sophisticated applications that are one-hundred-percent Motif-compliant. The default palette includes a rich set of widgets and objects to suit most needs. With UIM/X PDS—short for UIM/X *Professional Developer's Suite*—the palette is expanded to include the complete set of enhanced XRT PDS widgets with extended functionality designed to make building professional applications even easier. This chapter provides a hands-on tutorial to familiarize you with the XRT PDS widgets.

The looks, functionality, and built-in behavior of the XRT PDS widgets present a great advancement over common Motif widgets. For example, Motif Push Buttons can contain a label or a bitmap, but not both at the same time. XRT Push Buttons overcome this limitation to allow both a label and a bitmap in the same widget. In addition, the PDS palette includes useful widgets simply not found in the motif environment, such as an XRT Table, an XRT Spin Box, and an XRT Graph.

This chapter provides a complete tutorial on building your first project with UIM/X and the XRT PDS widgets. First, you will load a start up project containing a To Do List built using typical Motif widgets. Then you will replace selected portions of the interface with the more advanced XRT PDS widgets that serve the same purposes—only better.

In this tutorial you will learn how to quickly design, test, generate code, and run a sample interface. You don't have to complete the whole tutorial in one sitting. You can stop at any point, save your work, and continue later. You do not need to be a programmer to understand and complete this tutorial.

Note – The project in this chapter was created using the Motif Window Manager (mwm) and its default resource values, except for `clientAutoPlace`, which was set to `false`. If you are using a different window manager, or have other than default values for window manager resources, you may see slightly different object appearance and behavior from that described in this chapter.

The GUI You Will Build

In this tutorial, you will create a To Do List, which lists tasks to be done along with their due dates and priorities. This interface provides push button control for modifying and deleting tasks from your list. The To Do window consists of three main areas, as shown in Figure 4-1:

- *Menu Bar*: Provides two pull-down menus: A File menu for opening and saving files, and exiting from the program; and a Help menu for seeing on-line help.
- *List Area*: Contains two lists of tasks, for home and office, with a description, priority and due date for each task. Push Buttons are provided for editing and deleting a selected task.
- *Edit Area*: Provides a Tool Bar for adding a new task, replacing the current task, and clearing the Task field. A Spin Box lets you quickly assign a priority to a task, and a Date Field provides automatic validation for the task's due date. A Combo Box allows you to select between *Home* and *Office* tasks.

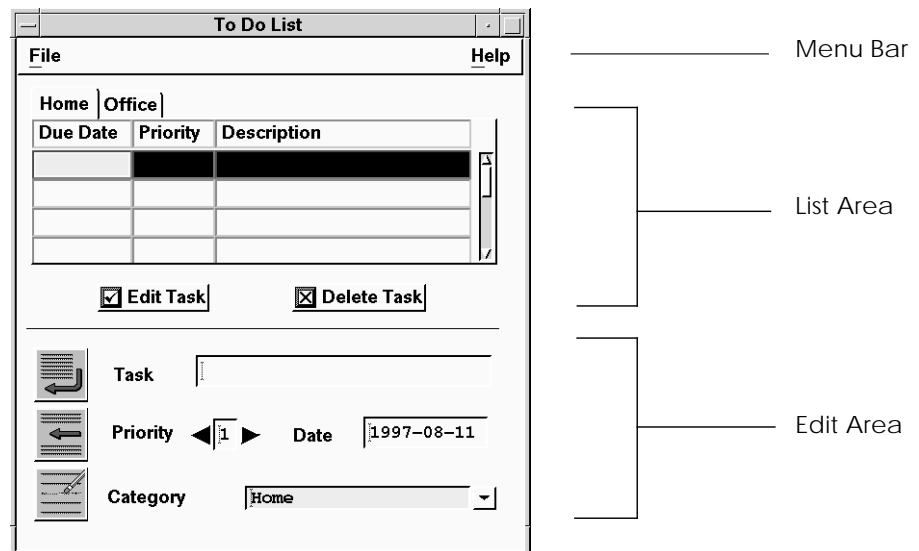


Figure 4-1 XRT PDS To Do List

The Steps in Building the To Do List

This tutorial takes about 120 minutes to complete. It contains the following steps:

Step #1: Starting UIM/X PDS

Step #2: Loading the Start-Up Project

Step #3: Testing the Start-Up Project

Step #4: Replacing the List Area Widgets

Step #5: Saving Your Work

Step #6: Setting New List Area Properties

Step #7: Replacing the Edit Area Widgets

Step #8: Setting New Edit Area Properties

Step #9: Adding Behavior to the Edit and Delete Push Buttons

Step #10: Adding Behavior to the Tool Bar

Step #11: Modifying the To Do List Menus

Step #12: Testing the To Do List

Step #13: Generating Code

Step #1: Starting UIM/X PDS

Before you begin building the interface, set up a new directory as follows:

1. Start the X Window System.
2. Bring up a terminal window.
3. Make a directory to hold the files you create in this tutorial:

```
mkdir todo
```

4. Change to the directory you just created:

```
cd todo
```

5. If you have not already done so, set the XRTHOME variable to the XRT root directory:

```
setenv XRTHOME xrt_directory
```

6. Start UIM/X PDS from your new directory:

```
uimx &
```

If your PATH variable does not provide the full path to the UIM/X PDS executable, you will have to specify it when you run the program:

```
uimx_directory/bin/uimx &
```

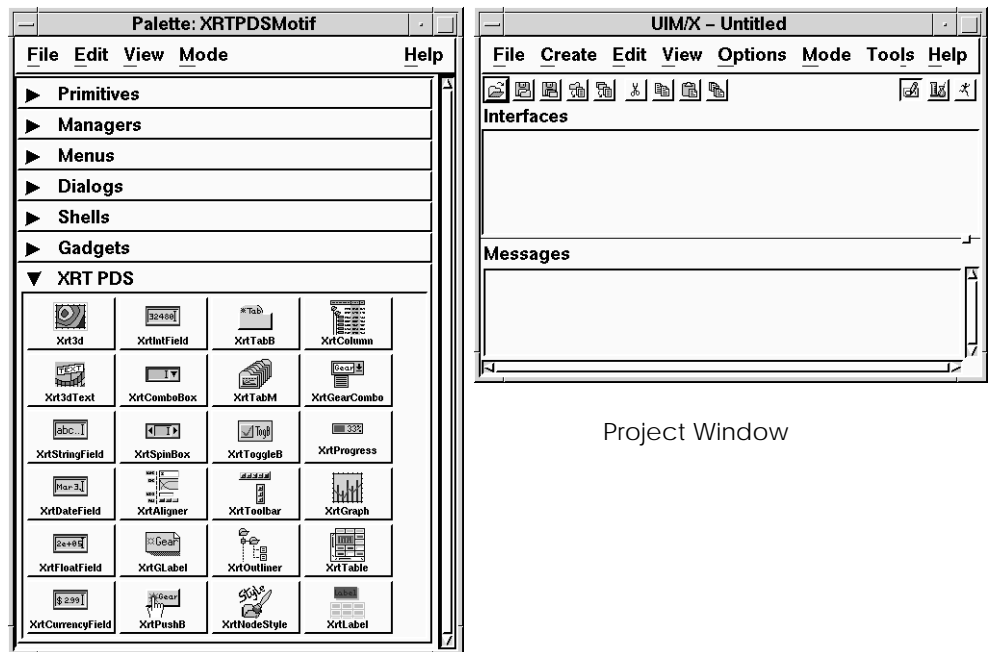
uimx_directory is the base directory where you installed UIM/X PDS.

After a brief pause, a copyright notice window appears, to show that UIM/X PDS is being initialized. When UIM/X PDS is ready, the Project Window and Palette appear, as shown in Figure 4-2. (For display purposes,

only the XRT PDS widgets are shown, with a view of the widgets by both name and icon. By default, all categories are shown, with a view by icon only.)

7. Iconify the terminal window.

Note – To restart this tutorial at any time, begin again from Step 4 above.



UIM/X PDS Palette

Figure 4-2 UIM/X PDS Palette and Project Window

Step #2: Loading the Start-Up Project

To facilitate development of the To Do List, a start-up project has been provided. It contains the To Do List main interface with menus already defined, plus a Message Box. In this step you will load the start-up project, and copy some graphics files you will need (for the Push Buttons and Tool Bar) to your working directory.

1. Copy the To Do List project files to your work directory:


```
cp uimx_directory/contrib/ToDoList/* .
```

2. In the same way, copy the graphics you will need:

```
cp xrt_directory/icons/desktop/toolbar/z_button_check.xbm .
cp xrt_directory/icons/desktop/toolbar/z_button_tick.xbm .
cp xrt_directory/icons/desktop/editing/insert.xpm .
cp xrt_directory/icons/desktop/editing/change.xpm .
cp xrt_directory/icons/desktop/editing/append.xpm .
```

3. Change the permissions on the files you copied to make them writable:

```
chmod a+w *
```

4. Choose File⇒Open in the Project Window, or click on the Open icon  in the Tool Bar.
5. Select `ToDo.prj` and click on OK.

Messages will appear indicating that you are loading an interface created in Novice Mode, and asking if you would like to replace compound widgets with individual widgets. Novice Mode is UIM/X PDS' mode of operation with simplified menus, fewer widgets, and limited options that allows new users to become productive immediately. Compound widgets are collections of widgets that work together in Novice Mode. Since you are running in Standard Mode, the compound widgets can be made available as individual widgets. This is done automatically.

6. Select Replace.

The To Do List start-up interface appears, as shown in Figure 4-3.

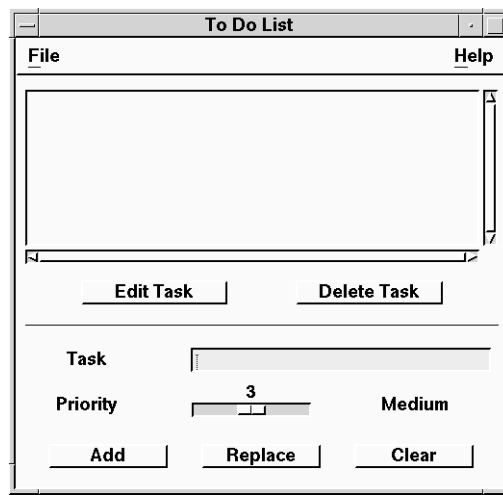


Figure 4-3 To Do List Start-Up Project

The Message Box provided is not visible by default, though an icon for it appears in the Project Window. This hidden interface is displayed by a callback provided in the menu. You will test it later.

Step #3: Testing the Start-Up Project


Before modifying the interface to use XRT PDS widgets, you can test the interface's functionality. UIM/X PDS provides a Test mode that enables you to test interfaces instantly, without having to generate code, compile and link the program.

1. Switch to Test Mode by clicking the Test icon  in the Project Window.

The Palette, the Property Editor, and any other open editors disappear from your screen.

2. Try the various features of the interface:

- Type a task in the Task field, assign it an importance using the Priority scale, then click on the Add push button to add it to the List area.
- Click on the Push Buttons to enter, modify and delete tasks. (Do not use commas in your task descriptions. Commas interfere with the start-up project's relatively simple formatting abilities.)
- Use the Scale to set their priorities.
- Pull down the File and Help menus, and try each of their options.
- Notice that if you try to exit the program, the Application Window's default `DeleteResponse` property intercepts the call and shows a message instead of allowing you to exit.
- Pop up the Help About dialog.

3. When you are through, switch back to Design mode by clicking on the Design icon .

The Palette and the editors reappear on your screen.


Step #4: Replacing the List Area Widgets

The *Palette* is a toolbox of objects you can use to build interfaces. By default, objects are shown as icons only. You can change this to display objects by name only, or by both name and icon. To create an interface object in UIM/X PDS, you drag it from the Palette and drop it in the desired location on your screen, or you can click on it then drag and draw it on your interface. Dragging and dropping creates an object in its default size, while dragging and drawing lets you create one exactly the size you want. (However you create them, you can always resize objects later.)

In this step you will remove the List Area widgets provided in the start up project, replacing them with XRT PDS widgets; namely, an XRT Tab Manager, XRT Tab Buttons, XRT Table, and XRT Push Buttons. To delete the current widgets, you will use click-selection and marquee selection. To add the new widgets, you will use drag and drop, and drag and draw.

Selecting and Removing the List Area Widgets

In UIM/X PDS you can select widgets by clicking on them, or by marquee selection. In this step you will delete widgets using click selection and marquee selection.

1. Check that you are in Design mode. If not, click on the Design icon  in the Project Window.
2. Click on the Scrolled List to select it, using the Select mouse button (the left one).

Selection handles appear around the Scrolled List, as shown in Figure 4-4. A widget must be selected before you can delete it, duplicate it, or change its properties.

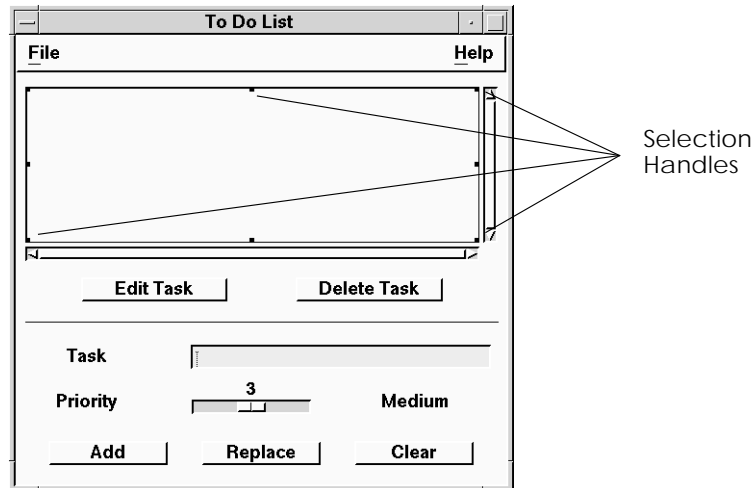


Figure 4-4 Scrolled List Selected

3. Display the Selected Objects popup menu by pressing and holding the Menu mouse button (the right-most one) while over the Scrolled List.

The Selected Objects popup menu appears, as shown in Figure 4-5.

Selected Objects (scrolledList1)	
Tools	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Duplicate	
Align	
Arrange	
Delete	
Other	
Managers	
Primitives	
Gadgets	
Menus	
XRT PDS	
Instance	

Figure 4-5 Selected Objects Popup Menu

4. Choose Selected Objects⇒Delete and confirm your choice in the dialog box.
In UIM/X PDS you are always asked to confirm widget deletions.
5. To select the Edit Task and Delete Task Push Buttons by marquee selection, begin by pointing above and to the left of both Push Buttons.
6. Press and hold the Select mouse button, then drag the mouse pointer below and to the right of both Push Buttons.

The mouse cursor changes to an “O” shape, and a marquee—a dashed box—follows the pointer, as shown in Figure 4-6.

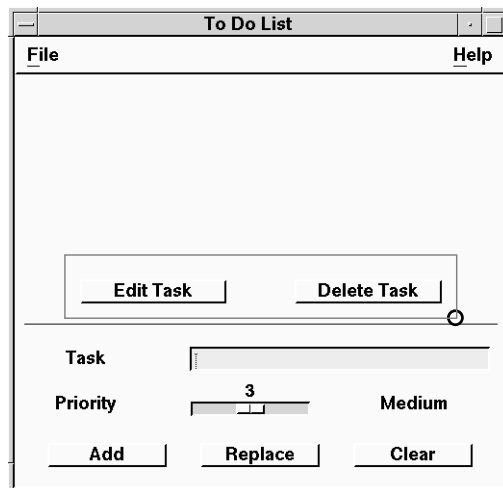


Figure 4-6 Selecting Both Push Buttons at Once

7. Remove the objects from the interface by choosing Selected Objects⇒Delete.

Adding an XRT Tab Manager

In this step you will create an XRT Tab Manager by dragging and drawing.

1. In the XRT PDS category of the Palette, click on the Tab Manager icon with the Select mouse button as shown in Figure 4-7.

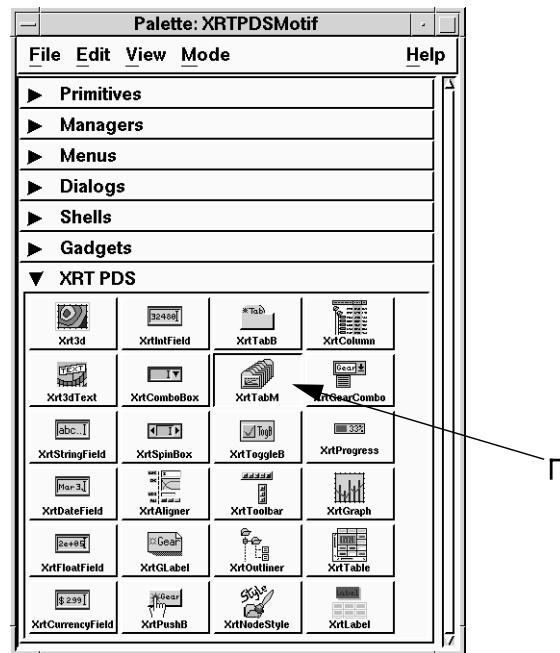



Figure 4-7 Selecting a Tab Manager from the Palette

The mouse pointer changes to a “corner” shape  representing the widget’s upper-left corner.

Note – You can cancel any operation performed with the Select or Adjust mouse button by pressing the Esc key. Pressing the Esc key is a convenient way to cancel drag and draw operations, for example.

2. Press and hold down the Select button where you want the top left corner of the Tab Manager to appear.
3. While holding down the Select button, drag the mouse down and to the right to define the size of the new widget, as shown in Figure 4-8.

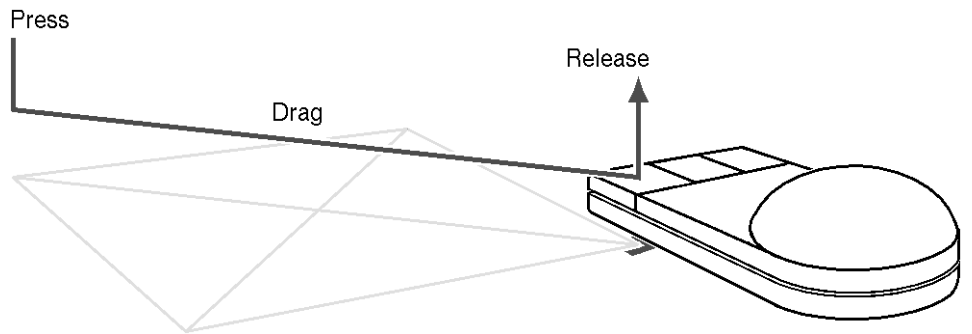


Figure 4-8 Creating a Tab Manager by Dragging and Drawing

4. Release the mouse button to complete the operation.

The Tab Manager appears as shown in Figure 4-9. Don't worry if the Tab Manager is not the size or shape you want. You will learn how to reposition and resize it in a moment.

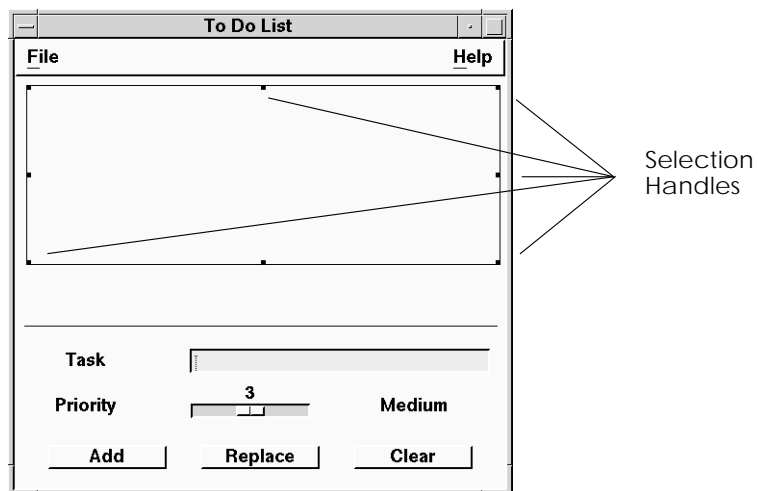
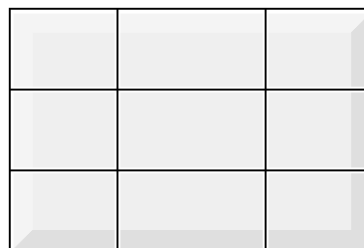


Figure 4-9 Tab Manager Added to the Interface

Also notice the selection handles appearing around the new widget. As noted, widget must be selected before you can move it, resize it, or change its properties. Newly drawn widgets are automatically selected.

Moving and Resizing Objects

UIM/X PDS simplifies moving and resizing objects with the Resize grid. By pressing the Adjust mouse button (the center one) over a selected object you cause the grid to appear, as shown in Figure 4-10. Depending on the position of the mouse you can stretch the object horizontally or vertically, or enlarge it in both directions at the same time. You don't even have to click on the selection handle itself, just in a region. By positioning the mouse pointer in the center square, you can move the object without resizing it.



Each widget has nine invisible regions for moving and resizing

Figure 4-10 Resize Grid

Note – Do not move or resize an interface using its window decorations (the box that appears around it). This communicates information to the window manager only, and will result in the object returning to its original size and location at runtime.

Adding a Tab Button and Table

In this step you will add an XRT Tab Button and an XRT Table to the interface by dragging and dropping.

1. Point to the Tab Button icon in the XRT PDS category of the Palette.
2. Press and hold down the Adjust button.

The pointer changes to a compass pointer, and an outline of the object appears. This means the object is ready for you to drag and drop.

3. Drag the outline of the object to the top of your Tab Manager.

4. Release the mouse button.

The new Tab Button appears in its default size, as shown in as shown in Figure 4-11.

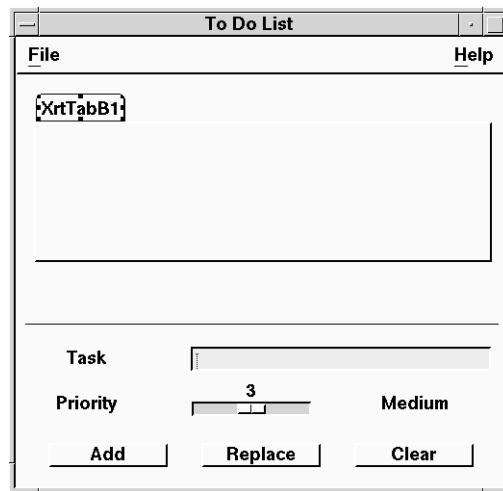


Figure 4-11 Interface with Tab Button Added

5. Similarly, add an XRT Table by pointing to the Table icon in the XRT PDS category of the Palette, pressing and holding the Adjust button, then dragging the outline of the object onto the Tab Manager.

A default sized Table appears, as shown in as shown in Figure 4-12.

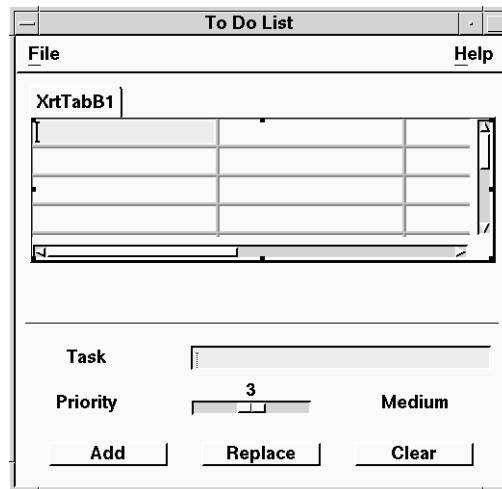


Figure 4-12 Interface with Table Added

Adding Enhanced XRT Push Buttons

In this step you will create the two XRT Push Buttons to replace those you deleted earlier, placing them below the Tab Manager. You will use drag and drop to create the first XRT Push Button. Next, you will create the second one by duplicating the first. Duplicating objects saves time and gives an uniform look to your interface.


1. Point to the XRT Push Button icon in the XRT PDS area of the Palette.
2. As before, press and hold down the Adjust button.
3. Drag the outline to your interface, on the left below the Tab Manager.
4. Release the mouse button.

A default sized XRT Push Button, called `XrtPushB1`, appears where you dropped it. Don't worry if it is not the size you want. You will resize it in a moment.

Note – The font used for all labels in your interface depends on your default font, set in your `.xdefaults` file. You can change this font if necessary.

5. With `XrtPushB1` selected, press the Menu button (the right-most one).
6. Choose Selected Objects⇒Duplicate.

Another Push Button, called `XrtPushB2` appears, overlapping the first.

Note - You can also duplicate a selected object by choosing the Edit⇒Duplicate command from the Project Window or by clicking on the Duplicate icon .

7. To move the duplicated XRT Push Button, point to its center, press the Adjust button, and drag it to the right under the Tab and Table widgets, as shown in Figure 4-13.

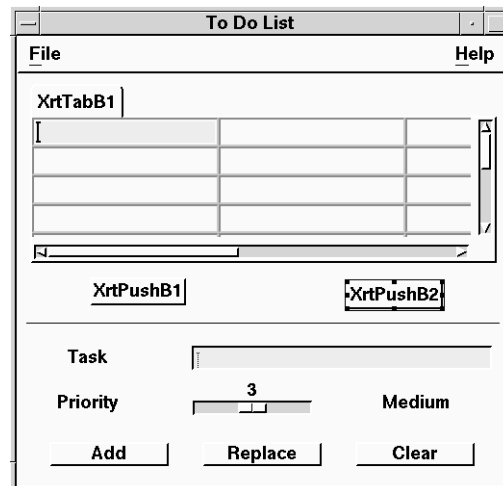


Figure 4-13 List Area with both XRT Push Buttons Added

Resizing and Aligning the Push Buttons

In this step you will work with both XRT Push Buttons at once to resize and align them.

1. Select the two XRT Push Buttons using marquee selection, or click selection.

To use marquee selection, press the Select mouse button and drag the dotted outline to surround both XRT Push Buttons. To click select, click on the first XRT Push Button to select it, then hold Ctrl and click on the second.

2. To resize both XRT Push Buttons at once, point to one of them and use the resize grid.

The other resizes automatically.

3. To align the XRT Push Buttons, press the Menu button, and choose Selected Objects⇒Align (with both Push Buttons still selected).

The Align submenu appears, as shown in Figure 4-14.

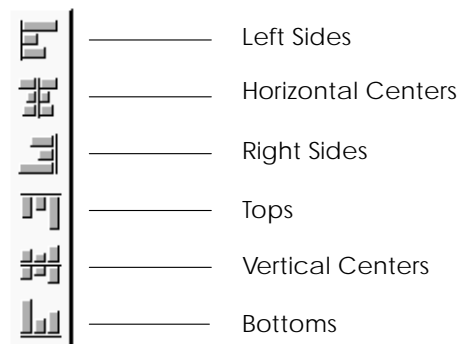




Figure 4-14 Align Submenu

4. Choose Align⇒ to align both Push Buttons by their bottoms.
5. Space the Push Buttons out equally by selecting both Push Buttons, and choosing Selected Objects⇒Arrange ⇒ 

When complete, the interface should appear as shown in Figure 4-15.

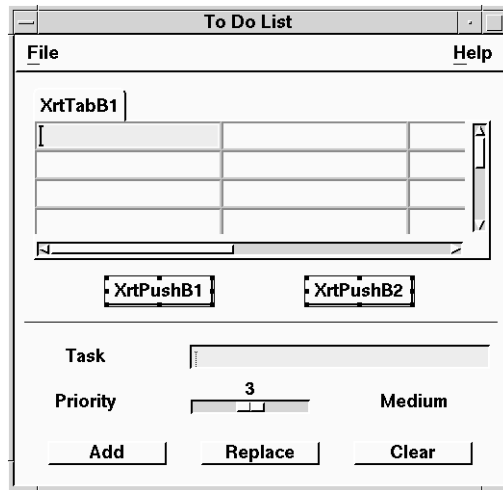


Figure 4-15 Push Buttons Aligned and Arranged

Step #5: Saving Your Work

As with any type of software, you should often save your work in UIM/X. UIM/X PDS facilitates the task of saving (and reloading) your interface with the notion of a *project*.

1. Choose the File⇒Save Project As command from the Project Window.

The File Selection dialog box appears, with the name of the start-up project, `ToDo.prj`, in your new `todo` directory, as shown in Figure 4-16.

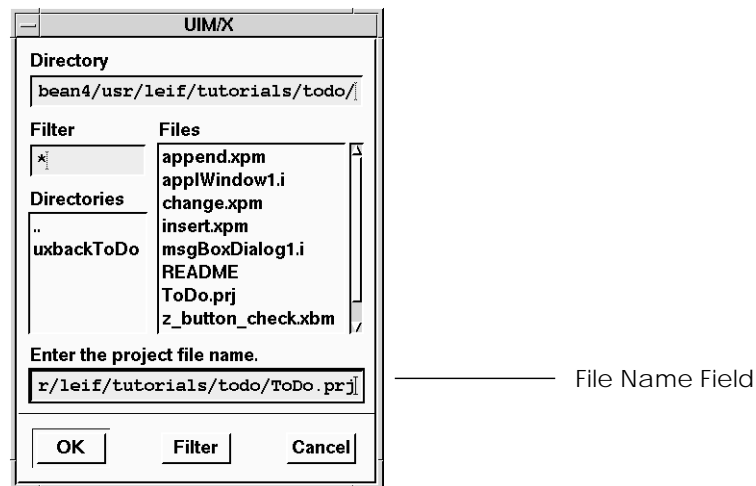




Figure 4-16 File Selection Dialog Box

2. If you were to rename the project, you could replace the name provided in the File Name field with a new name. For the purposes of the tutorial the default name will suffice.
3. Click OK to save your work.
4. Since you loaded a start up project, you will be asked if you want to overwrite the existing files. Confirm your choice by clicking OK.

Project information is saved in `ToDo.prj`, and other files are saved with the project. You can reload the project file and associated files at any time, using the File⇒Open command in the Project Window or by selecting the Open icon .

For the rest of this session, choosing File⇒Save Project or its corresponding icon  will save the latest version of your project in the file `ToDo.prj`.

Step #6: Setting New List Area Properties

Now that the widgets are in place in the List Area you are ready to begin changing their labels and other properties. A *property* is a value assigned to a widget to determine its size, shape, color, font, behavior, or other characteristics. Each widget in UIM/X PDS has core properties that control its height, width, background and foreground colors, as well as its position in the interface or on the desktop (depending on the object). Each widget also has a set of properties unique to itself. For more details on specific properties, see the UIM/X documentation set.

You can view and change all the available properties for widget using the built-in Property Editor. For example, the XRT Push Buttons you created were given the default labels `XrtPushB1` and `XrtPushB1`, respectively. Using the Property Editor, you can change these labels to something more meaningful.

UIM/X PDS features an editor called the Browser that presents a hierarchical view of your interface. Widget names and their relationships to one another are shown, but their graphical elements are not displayed. You can perform operations in the Browser just as you would in the interface. You can select a widget in the Browser and drag and drop it into the Property Editor, for example. The Browser makes it easy to work with nested widgets (such as the Tab Manager, Tab Button, and Table).

In this step you will use the Property Editor to give the Push Buttons new labels and icons. Next, you will change the Tab Button's properties. Then you will change properties for the Table. Finally, you will use the Browser to duplicate the Tab Button and Table, and adjust the properties for the "Home" task list.

Changing the XRT Push Button Labels

In this step you will load each XRT Push Button into the Property Editor to give them new labels and icons.

1. Double click on the first XRT Push Button, `XrtPushB1`, to load it into the Property Editor in one move.

The Property Editor appears, loaded with the XRT Push Button, as shown in Figure 4-17.

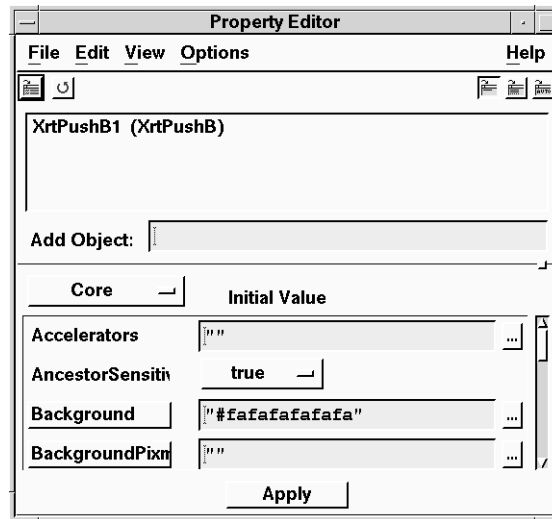


Figure 4-17 Property Editor Loaded with an XRT Push Button

Note – You can also load an object into the Property Editor by selecting it and choosing Selected Objects⇒Tools⇒Property Editor. To load an object into an already open Property Editor, drag and drop the widget from the interface.

Notice that all properties are listed in alphabetical order. For convenience, properties are grouped by category.

2. Change from the Core category of properties to the Specific category by clicking on the category button as shown in Figure 4-18.
3. Scroll down the list of properties to locate the `LabelString` property.

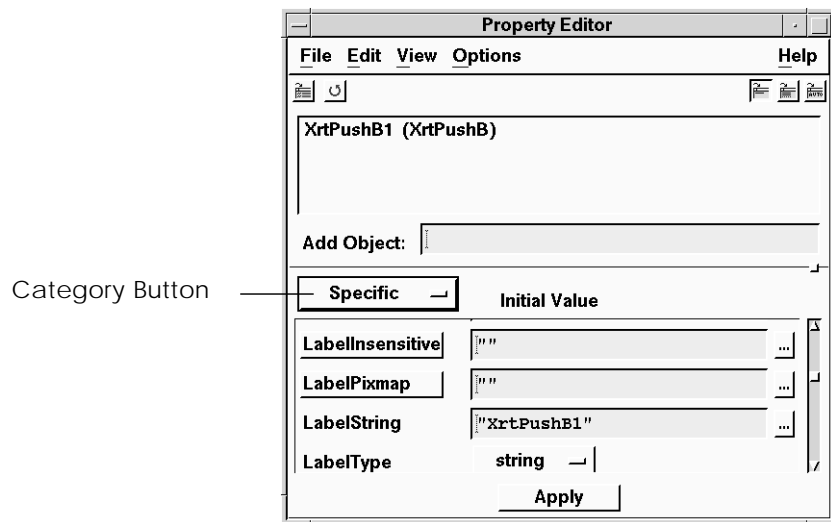


Figure 4-18 Property Editor Showing LabelString Property

4. Replace "XrtPushB1" with "Edit Task".

Be sure to include the quotation marks around the string.

5. Apply the change to the Push Button by clicking on the Apply button at the bottom of the Property Editor.

Note the change in appearance of the Push Button, as shown in Figure 4-19.

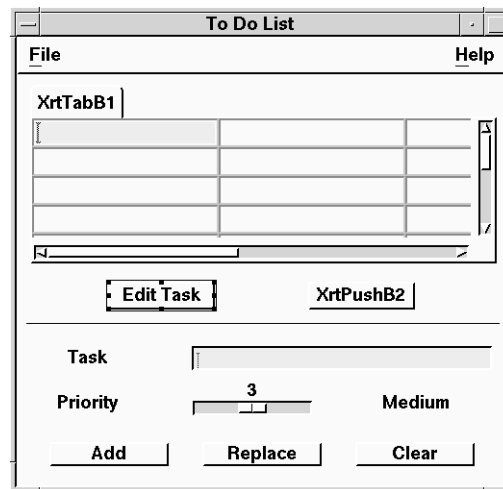


Figure 4-19 To Do List with New Label for *Edit Task* Push Button

6. Locate the `XrtGearLabelPixmap_PB` property in the Specific category, and click on the `XrtGearLabelPixmap_PB` button to open the Icon Viewer.

The Icon Viewer appears, as shown Figure 4-20.

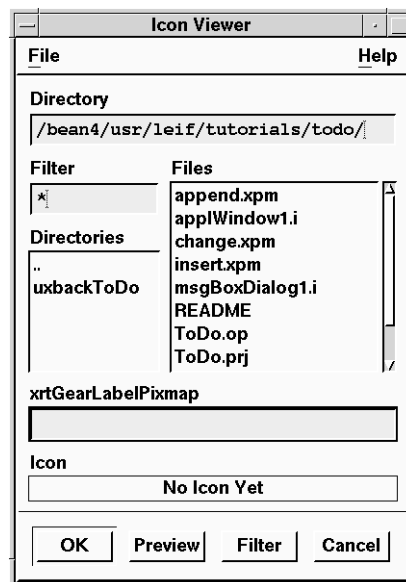


Figure 4-20 Icon Viewer

7. Locate the `z_button_tick.xbm` file, and click on Ok in the Icon Viewer.
8. Finally, locate the `XrtGearLabelType_PB` property, changing it from `Type String` to `Type String Pixmap`.
9. Apply the changes to the Push Button by clicking on the Apply button at the bottom of the Property Editor.

Note the change in appearance of the Push Button, as shown in Figure 4-21.

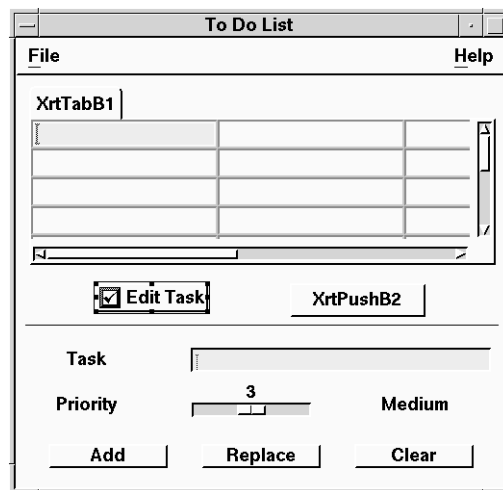


Figure 4-21 Push Button with New Caption and Icon

10. To load the second Push Button into the Property Editor, click on it using the Adjust mouse button, as if you were going to move the widget.

The mouse pointer changes to the compass shape, and an outline of the widget appears. If the resize grid appears, press Esc and try again, closer to the center of the widget.

11. Drag the outline of the widget to the List area of the Property Editor, as shown in Figure 4-22, then release the mouse button.

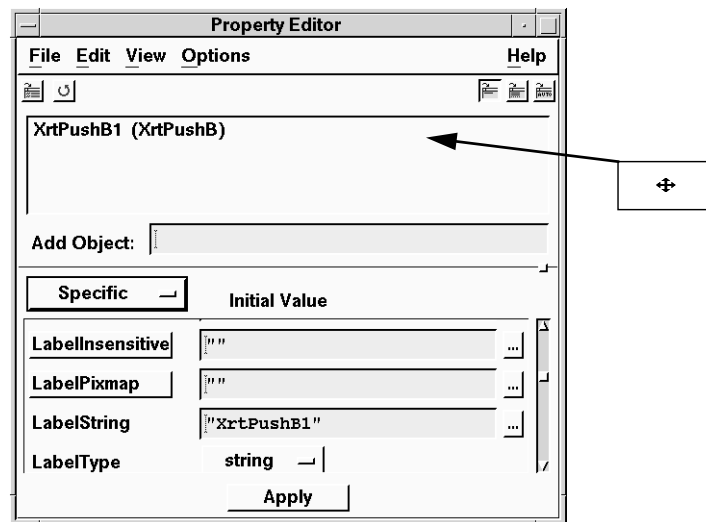


Figure 4-22 Using Drag and Drop with the Property Editor

If the widget is too large to “fit” in the List area, be sure to move the compass shape within the area.

12. As before, locate the `LabelString` property in the Specific category, this time replacing `"XrtPushB2"` with `"Delete Task"`.
13. Locate the `LabelPixmap` property in the Specific category, and click on the `LabelPixmap` button to open the Icon Viewer.
14. Locate the `z_button_check.xbm` file, and click on `Ok` in the Icon Viewer.
15. Change the `XrtGearLabelType_PB` property from `Type String` to `Type String Pixmap`.
16. Apply the changes to the interface by clicking on `Apply` in the Property Editor.

The To Do List is updated to display the icons, as shown in Figure 4-23.

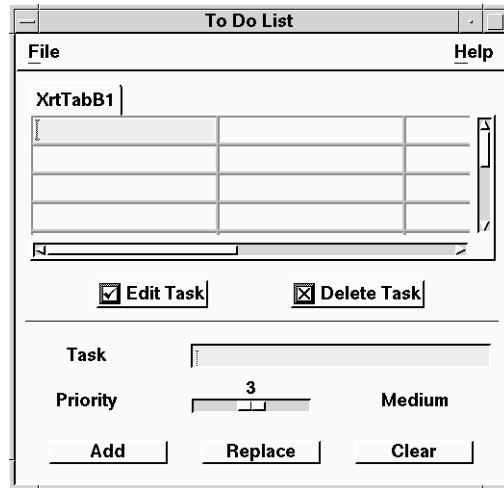


Figure 4-23 XRT Push Buttons with Labels and Icons

17. Close the Property Editor by choosing File⇒Close.
18. Save your work.

Changing the Tab Button's Label

In this step you will change the Tab Button's label to something more meaningful.

1. Double-click on the Tab Button to open a new Property Editor.
2. Locate the `LabelString` property in the Specific category, changing it from "XrtTabB1" to "Home".
3. Apply your changes by clicking on Apply.

The Tab Button is updated, as shown in Figure 4-24.

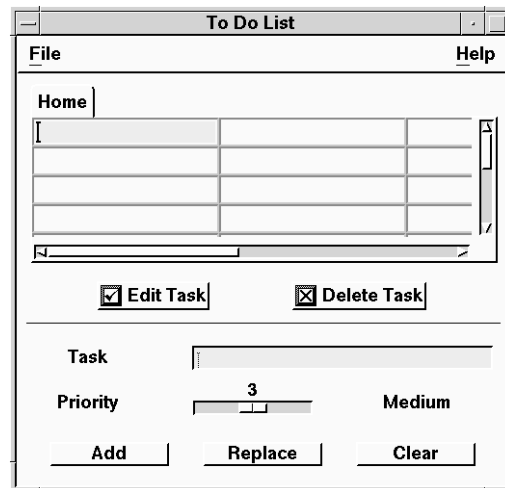


Figure 4-24 Tab Button Showing New Label

4. Save your work.

Changing the Table's Properties

In this step you will load the Table into the Property Editor to change its properties. First you will change the Table so the user can select a row at a time, rather than a cell at a time. Next you will prevent the user from resizing the table's cells. You will also give the Table column headings, and make it more attractive.

1. Drag and drop the Table widget into the already open Property Editor.
2. In the Specific category locate the `XrtTblMode` property, changing it from `Mode Table` to `Mode List`.
3. Locate the `XrtTblSelectionPolicy` property, changing it from `Select None` to `Select Single`.
4. Locate the `XrtTblEditableSeries` property, changing it from `"(ALL ALL True)"` to `"(ALL ALL False)"`.
5. To prevent the user from resizing cells, change the `XrtTblAllowCellResize` property from `Resize All` to `Resize None`.

6. Locate the `XrtTblNumColumns` property, changing it from the default to 3.
7. Locate the `XrtTblColumnLabels` property, changing it to "Due Date, Priority, Description".
8. Add a little space between the column label and the columns themselves by locating the `XrtTblColumnLabelOffset` property, changing it from 0 to 2.
9. Locate the `XrtTblCharWidthSeries` property, and click on `XrtTblCharWidthSeries` button to open the Series Editor.

The Series Editor appears, as shown in Figure 4-25.

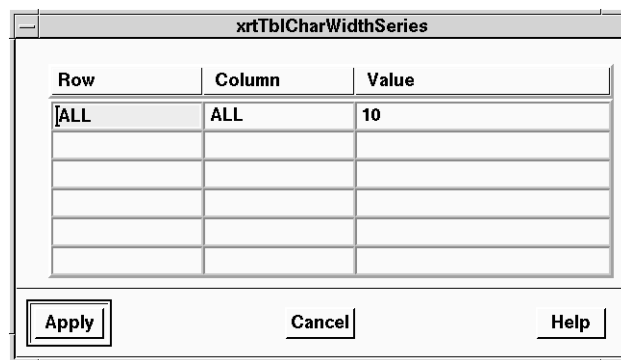
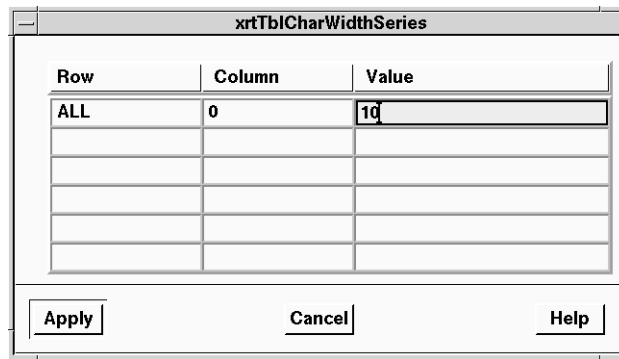


Figure 4-25 Series Editor for `xrtTblCharWidthSeries` Property

10. To set the width of the Date column to 10, begin by clicking in the first row. Enter ALL, 0, and 10 for the Row, Column, and Value, respectively, as shown in Figure 4-26.



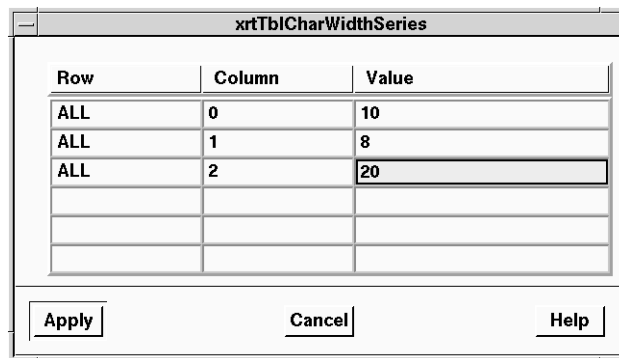
Row	Column	Value
ALL	0	10

Apply Cancel Help

Figure 4-26 Series Editor with Constraints For Date Column

11. Add a second and third row of constraints for the Priority and Description columns:
 - ALL, 1, 8
 - ALL, 2, 20

When complete the Series Editor should appear as shown in Figure 4-27.



Row	Column	Value
ALL	0	10
ALL	1	8
ALL	2	20

Apply Cancel Help

Figure 4-27 Series Editor Showing All Constraints

12. Click on Apply in the Series Editor.

The property is updated to show "(ALL 0 10) (ALL 1 8) (ALL 2 20)".

13. Give the table a decorative border by locating the `XrtTblFrameBorderStyle` property, changing it from `Border None` to `Border Etched In`.

For convenience, the properties to be changed are listed in Table 4-1.

Table 4-1 Table Properties

Property	New Value
<code>XrtTblAllowCellResize</code>	Resize None
<code>XrtTblCharWidthSeries</code>	"(ALL 0 10) (ALL 1 8) (ALL 2 20)"
<code>XrtTblColumnLabelOffset</code>	2
<code>XrtTblColumnLabels</code>	"Due Date, Priority, Description"
<code>XrtTblEditableSeries</code>	"(ALL ALL False)"
<code>XrtTblFrameBorderStyle</code>	Border Etched In
<code>XrtTblMode</code>	Mode List
<code>XrtTblNumColumns</code>	3
<code>XrtTblSelectionPolicy</code>	Select Single

14. Apply your changes by clicking on Apply.

The Table is updated, as shown in Figure 4-28.

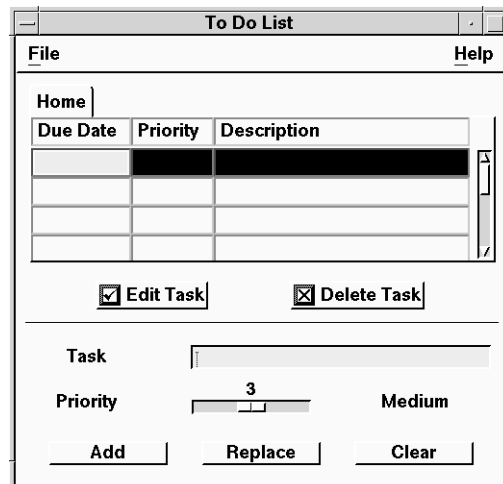


Figure 4-28 Table Showing New Property Settings

15. Close the Property Editor by choosing File⇒Close.
16. Save your work.

Duplicating the Tab Button and Table Using the Browser

In this step you will duplicate the Tab Button and Table using the Browser, then customize the duplicates.

1. Select the Tab Manager, Tab Button, Table, or any other widget in the interface.

To open the Browser a widget must be selected.

2. Open the Browser by selecting Selected Objects⇒Tools⇒Browser.

(Recall that to display the Selected Objects popup menu, you press and hold the Menu mouse button—the right most one—while over the selected interface.)

The Browser appears, loaded with the To Do List, as shown in Figure 4-29. (For display purposes, the Browser view is by name only. By default the Browser shows the widgets by icon.)

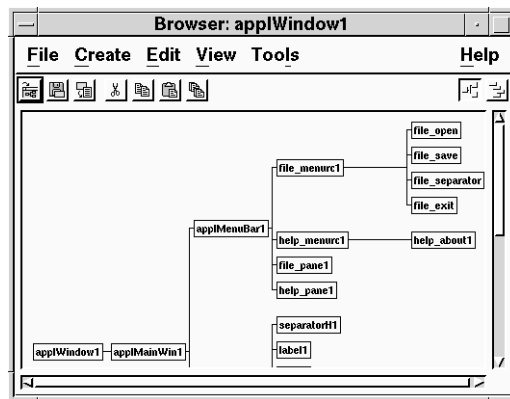


Figure 4-29 Browser Loaded with the To Do List

3. Choose the view you prefer using the View menu.

- You can view the interface's widgets by name, icon, or both name and icon.
 - You can display the structure of your interface in tree or outline format.
4. Scroll the Browser window to locate the Tab Button.

Following the UIM/X PDS naming conventions for widgets, the Tab Button is named `XrtTabB1`.

In the Browser, click on the Home Tab Button, `XrtTabB1`, press the `Ctrl` key, then click on the Table, `XrtTable1`, to select both widgets at once, as shown in Figure 4-30.

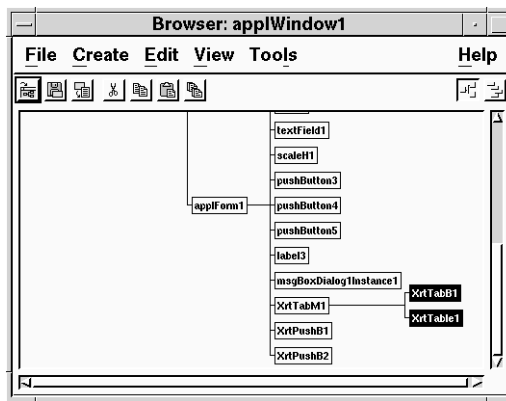


Figure 4-30 Selecting Widgets in the Browser

Note that by clicking on a widget's representation in the Browser, you select it in the interface.

5. Duplicate the widgets by choosing `Selected Objects⇒Duplicate` while over the Browser.

UIM/X PDS copies both objects, placing the copies within the Tab Manager. Notice the Browser is updated to show the new objects, as is the interface. The updated interface is shown in Figure 4-31.

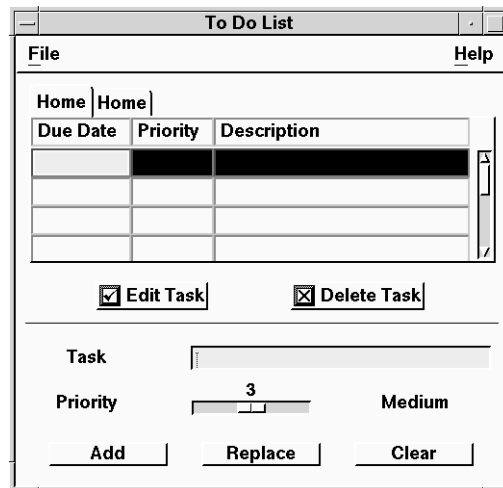


Figure 4-31 Interface with New Tab Button

6. In the Browser select the new Tab Button, `XrtTabB2` by clicking on it.
7. Load the Tab Button into the Property Editor by pressing the Menu mouse button while over the Browser and choosing Selected Objects⇒Tools⇒Property Editor.

The Property Editor appears, loaded with the Tab Button.

8. Locate the `LabelString` property in the Specific category, changing it from "Home" to "Office".
9. Apply your changes by clicking on Apply.

The Tab Button is updated, as shown in Figure 4-32.

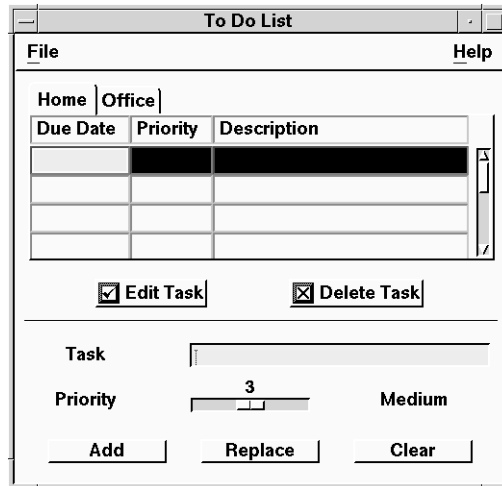


Figure 4-32 Interface Showing Both Tab Buttons

10. Close the Property Editor by choosing File⇒Close from the Property Editor menu.
11. Close the Browser by choosing File⇒Close from the Browser menu.
12. Save your work.

Step #7: Replacing the Edit Area Widgets

The Edit Area currently includes a Field where you enter tasks, a Scale to set priorities, and Labels that identify the areas. It also contains Push Buttons to Add, Replace, or Clear the currently selected task. You will replace these with the more advanced XRT PDS widgets: an XRT String Field for entering the task, an XRT Date Field for setting its due date, an XRT Spin Box for the priority, and an XRT Combo Box for its category (either *Home* or *Office*). You will also add an XRT Tool Bar containing XRT Push Buttons to add, replace, or clear the current task.

Removing the Constraints

When the start up interface was constructed, size and position constraints were added to it using the Constraint Editor. Before replacing the widgets, delete the constraints as follows (otherwise it will be difficult to move the widgets):

1. Select any widget on your interface.
2. Open the Constraint Editor by choosing Selected Objects⇒Tools⇒Constraints.

The Constraint Editor appears, as shown in Figure 4-33.

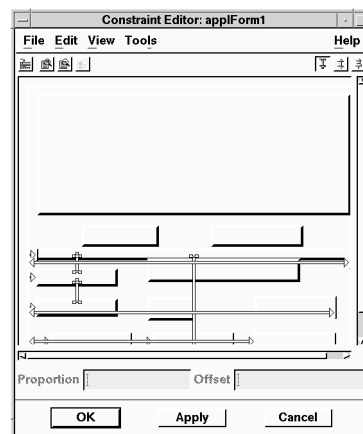


Figure 4-33 Constraint Editor

3. Choose Edit⇒Select All.

4. Choose Edit⇒Delete.

The constraints disappear.

5. Click OK in the Constraint Editor to apply the changes and close the editor.

Replacing the Widgets

In this step you will replace the Text Field with an XRT String Field. You will also replace the Scale with a Spin Box. You will add some new widgets too: a Label and Combo box (to hold the category of the task), and a Date Field next to the current *Medium* Label (you will change the label later).

1. Click on the Text Field to select it, as shown in Figure 4-34, then delete the widget.

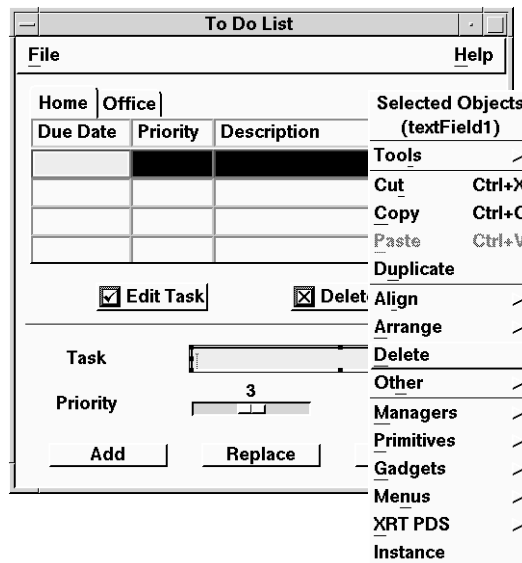


Figure 4-34 Deleting the Text Field

2. Replace the Text Field with an XRT String Field by dragging and dropping, or dragging and drawing.
3. In the same way, replace the Scale with an XRT Spin Box.
4. Double click on the XRT Spin Box to open the Property Editor and load it in.

5. In the Specific category, locate the `XrtSpinFieldType` property changing it from `String` to `None`, and apply the change by clicking on `Apply`.
6. Close the Property Editor by choosing `File⇒Close` from the Property Editor.
7. Drag and drop an XRT Integer Field into the Spin Box.

The Spin Box and Integer Field combination will work together to set the task's priority.

8. Add an XRT Date Field to the interface, positioning it to the right of the *Medium* Label. (You might want to move the Label a little to the left, first.)

Resize it so just the date shows (you will change its properties later).

9. Drag and drop an XRT Gear Combo Box (not an XRT Combo Box) onto the interface.

XRT Gear Combo Boxes are well suited for pick lists, while XRT Combo Boxes are designed for more complex structures.

10. Add a Label (from the Primitives category) to identify the Gear Combo Box. When complete, the interface should appear as shown in Figure 4-35.

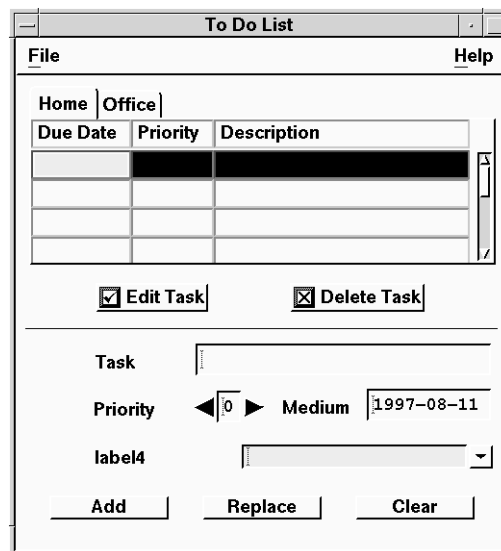


Figure 4-35 Edit Area Objects Replaced

11. Save your work

Creating the Tool Bar

In this step you will create a tool bar using the PDS widgets designed for this purpose. Since Tool Bars are oriented horizontally by default, you will load it into the Property Editor to change its orientation right away. Next you will add Push Buttons to the Tool Bar. Finally, you will resize all three Push Buttons at once to ensure uniform results.

1. Delete the *Add*, *Replace*, and *Clear* Push Buttons provided with the start up project.
2. Add an XRT Tool Bar to the interface, by dragging and drawing, as shown in Figure 4-36.

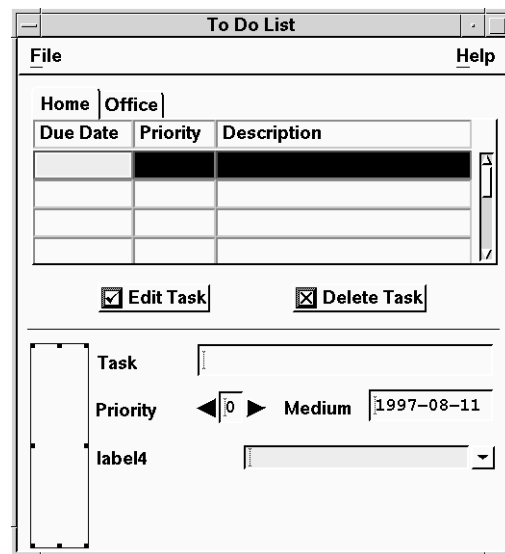


Figure 4-36 Edit Area with Tool Bar Added

3. Double click on the Tool Bar to open the Property Editor and load it in.
By default, Tool Bars are given a horizontal orientation. For the purposes of the tutorial, it should have a vertical orientation.
4. In the Specific category, locate the `XrtGearOrientation_Tbar` property changing it from horizontal to vertical, and apply the change by clicking on Apply.

5. Close the Property Editor by choosing File⇒Close from the Property Editor menu.
6. Drag and drop an XRT Push Button inside the Tool Bar, as shown in Figure 4-37.

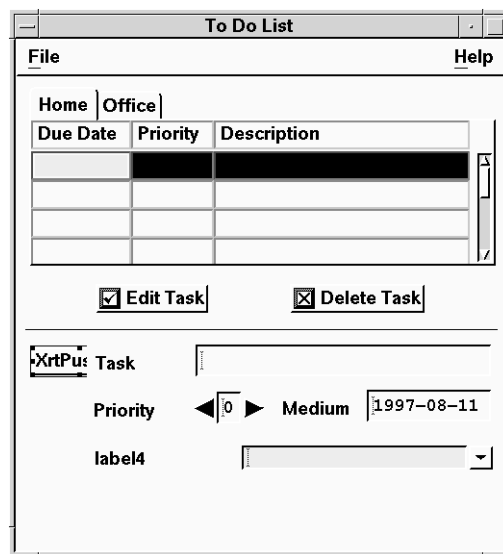


Figure 4-37 Tool Bar with Push Button Added

7. Duplicate the Push Button by choosing Selected Objects⇒Duplicate.
8. Create the third and final Push Button, also by duplication.
9. Since you oriented the Tool Bar vertically, it automatically places the Push Buttons one under the other, as shown in Figure 4-38.

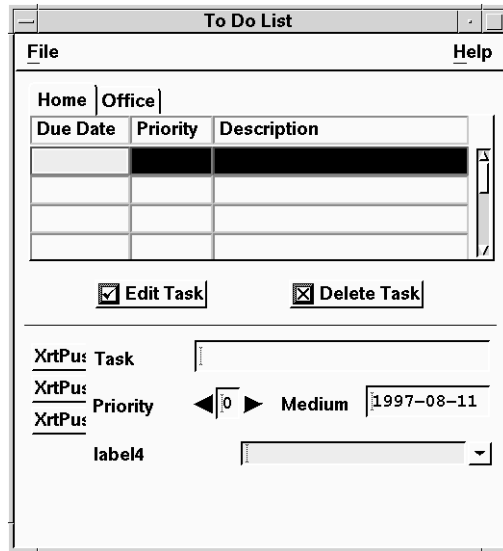


Figure 4-38 Tool Bar with all Three Push Buttons Added

10. Save your work.

Step #8: Setting New Edit Area Properties

Now that the Edit Area objects are in place, you are ready to change their properties. In the To Do List, many objects are labelled to indicate their purpose. You will change the labels to make them more appropriate. You will also change properties to make the other objects better suited to their tasks.

Changing Widget Properties

In this step you will load the new widgets into the Property Editor to change their properties.

1. Load the String Field beside the *Task* Label into a Property Editor by double-clicking on it.
2. Locate the `MaxLength` property in the Specific category, changing it from the default to 20.
3. Update the interface with your change by clicking on Apply in the Property Editor.
4. Load the *Medium* Label into the already open Property Editor by dragging and dropping it.
5. Locate the `LabelString` property in the Specific category, changing it from "Medium" to "Date".
6. Apply the change by clicking on Apply in the Property Editor.
7. Drag and drop the Date Field into the Property Editor.
8. Locate the `XrtDateFldEnterFormatList` property in the Specific category. This property specifies the formats permitted in the Date Field. Change it from the default to read as follows:

```
"YYYY-MM-DD"
```

9. Apply the change.

10. Continue loading objects into the Property Editor by dragging and dropping, updating properties according to Table 4-1.

For convenience, the table lists all the properties to be changed.

Table 4-2 Edit Area Widget Properties

Widget Name	Property	New Value
XrtStringField1	MaxLength	20
XrtDateField1	XrtDateFldEnterFormatList	"YYYY-MM-DD"
XrtIntField1	XrtIntFldInternalValue	1
	XrtIntFldPickList	"1,2,3"
Label3	LabelString	"Date"
Label4	LabelString	"Category"
XrtGearCombo1	Editable_Combo	false
	Value_Combo	"Home"
	XrtGearPickList_Combo	"Home,Office"

11. Apply your changes at each step.

When complete, the interface should appear as shown in Figure 4-40.

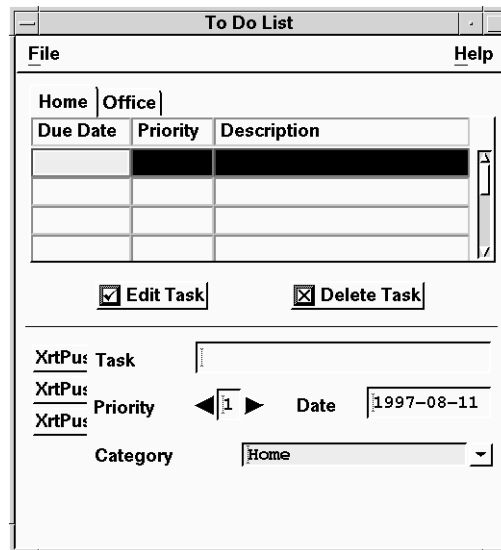


Figure 4-39 Edit Area with Properties Changed

12. Close the Property Editor
13. Save your work.

Changing the Tool Bar Properties

1. Double-click on `XrtPushB3` (the top Push Button in the Tool Bar) to select it and load it into the Property Editor in one move.
2. In the Specific category, locate the `XrtGearLableType_PB` property, changing it from `Type String` to `Type Pixmap`.
3. Locate the `XrtGearLabelPixmap_PB` property, and click on the `XrtGearLabelPixmap_PB` button to open the Icon Viewer.

Using the Icon Viewer, locate the `append.xpm` bitmap, and click on OK.

4. Apply the changes by clicking on Apply.

The Push Button reappears with its new icon, as shown in Figure 4-40.

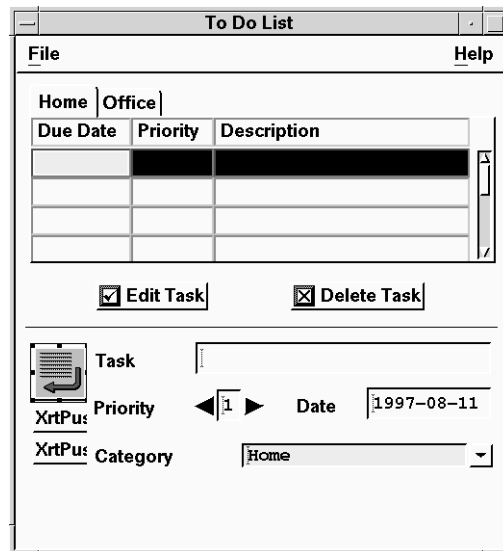


Figure 4-40 XrtPushB3 with New Icon

5. Repeat the same process for all the other Push Buttons in the Tool Bar, changing properties as shown in Table 4-3 (for convenience, changes for all three Push Buttons are shown).

Table 4-3 Tool Bar Push Button Properties

Object Name	Property	New Value
XrtPushB3	XrtGearLablePixmap_PB	Type Pixmap
	Icon	"append.xpm"
XrtPushB4	XrtGearLablePixmap_PB	Type Pixmap
	Icon	"insert.xpm"
XrtPushB5	XrtGearLablePixmap_PB	Type Pixmap
	Icon	"change.xpm"

6. If necessary, resize the Tool Bar to show all three Push Buttons, and rearrange the other widgets as desired, until your interface looks as shown in Figure 4-41.

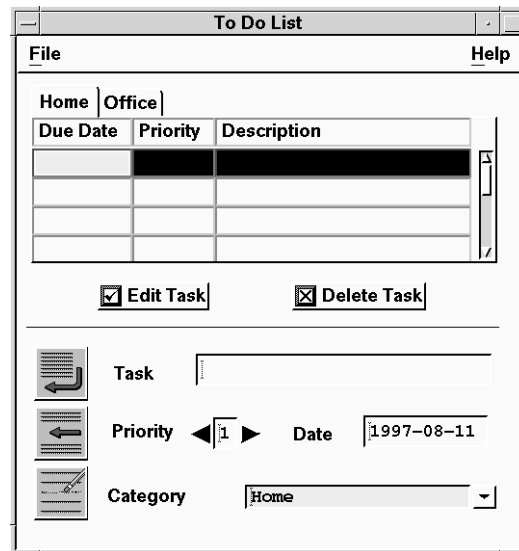


Figure 4-41 Edit Area with New Labels and Icons

7. Close the Property Editor.
8. Save your work.

You have now completed the look of your interface. In the next steps you will add its behavior.

Step #9: Adding Behavior to the Edit and Delete Push Buttons

In the To Do List project, there are two Push Buttons as follows:

- Edit Task copies the selected task from the List Area to the Edit Area so you can modify it
- Delete Task deletes the selected task from the List Area

In this step you will use the Declaration Editor to include application header files and declare a variable for the priority label, and the Callback Editor to add a callback for both the Push Buttons. To make a Push Button work, you provide it with an `ActivateCallback`.

1. To open the Declaration Editor, select any widget in the interface and choose Selected Objects⇒Tools⇒Declaration Editor.

The Declaration Editor appears, as shown in Figure 4-42.

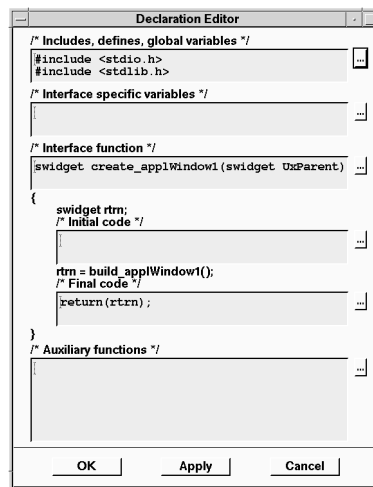


Figure 4-42 Declaration Editor

1. Click on the Text Editor button beside the `/* Includes, defines, global variables */` area.

2. In the Text Editor that appears, delete the following code (left over from the start up project):

```
char *priority_labels[5] = { "Very Low",  
    "Low",  
    "Medium",  
    "High",  
    "Very High" };
```

3. Enter the following declarations, after the existing code (all code is shown, for convenience, and the code you add is shown in bold):

```
#include <stdio.h>  
#include <stdlib.h>  
#ifdef __cplusplus  
#include <iostream.h>  
#endif /* __cplusplus */  
  
#include <Xm/XrtTable.h>  
#define MIN_TABLE_ROWS 10
```

4. Click on OK in the Text Editor, then click on OK in the Declaration Editor.
5. Double-click on the Edit Task Push Button to load it into the Property Editor.
6. In the Behavior category of properties, click on the Text Editor button beside ActivateCallback.

The Callback Editor appears, with an empty text window ready for your `ActivateCallback`, as shown in Figure 4-43.

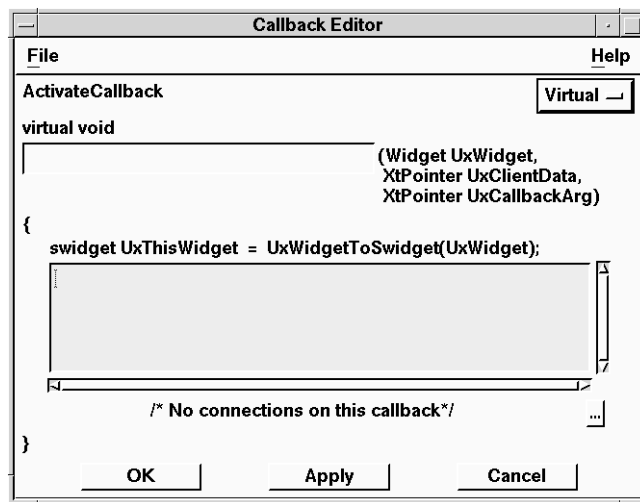


Figure 4-43 Callback Editor

7. Click in the Text Field, and type in the following code exactly as it appears:

```

swidget swTodoList;
swidget swTabB;
int selectedRow, selectedCol;
XrtTblCellValues *cellValues;

/* find the currently selected row */
swTodoList = UxFindSwidget
(UxGetXrtGearActivePageWidget_TabM(XrtTabM1));
XrtTblGetCurrentCell(UxGetWidget(swTodoList), &selectedRow,
&selectedCol);

```



```
/* get the cell values for the selected row */

cellValues = XrtTblGetCellValues(UxGetWidget(swTodoList),
    selectedRow, 0, selectedRow, UxGetXrtTblNumColumns
    (swTodoList)-1, TRUE);
if (cellValues != NULL)
{
    UxPutText(XrtDateField1, (char *) CELL_VALUE(cellValues,0,0));
    UxPutText(XrtIntField1, (char *) CELL_VALUE(cellValues,0,1));
    UxPutText(XrtStringField1, (char *)
        CELL_VALUE(cellValues,0,2));
}

XrtTblDestroyCellValues(cellValues);

/* now set the combo box to the correct category */

swTabB = UxFindSwidget
    (UxGetXrtGearActiveTabWidget_TabM(XrtTabM1));
UxPutValue_Combo(XrtGearCombo1, UxGetLabelString(swTabB));
UxPutXrtGearPickListIndex_Combo(XrtGearCombo1,
    UxGetXrtGearActivePageNumber_TabM(XrtTabM1));
```

This code does the following:

- Determines which task in the Table is selected
- Extracts the selected task's priority and description
- Copies the selected task's description into the *Description* Text Field
- Copies the selected task's priority into the *Priority* Spin Box
- Switches the Combo Box to the correct category.

8. Click OK in the Callback Editor.

9. Click Apply in the Property Editor.

If you made any errors typing the code, an error dialog box appears, an "X" appears beside the error in the Property Editor, and a message appears in the Messages Area of the Project Window.

Note – When you click Apply in the Property Editor, your callback code is parsed and stored with the associated object, but not run. You use the Test mode to test your callback code.

Clear up any error messages that appear, by checking your code against the sample code above.

Make sure to click Apply in the Property Editor.

10. Drag and drop the Delete Task Push Button into the Property Editor.
11. Click on the Text Editor button beside ActivateCallback, and type in the following code:

```
swidget swTodoList;
Widget wgtTodoList;
int selectedRow, selectedCol;

swTodoList = UxFindSwidget
    (UxGetXrtGearActivePageWidget_TabM(XrtTabM1));
wgtTodoList = UxGetWidget(swTodoList);

XrtTblGetCurrentCell(wgtTodoList, &selectedRow, &selectedCol);
XrtTblDeleteRows(wgtTodoList, selectedRow, 1, TRUE);

if (UxGetXrtTblNumRows(swTodoList) < MIN_TABLE_ROWS)
{
    XrtTblAddRows(wgtTodoList, MIN_TABLE_ROWS-1, 1, FALSE,
        NULL, 0);
}
```

This code deletes the selected task from the Table.

12. Click OK in Text Editor, OK in the Callback Editor, and Apply in the Property Editor.
13. Close the Property Editor.
14. Save your work.

Step #10: Adding Behavior to the Tool Bar

The Tool Bar contains three Push Buttons. From top to bottom, these are the following:

- Add copies a new task from the Edit Area to the List Area, adding it to the end of the table.
- Replace copies the task from the Edit area to the List Area, overwriting any selected task.
- Clear erases the current task from the Edit Area

To Add Behavior to the Tool Bar

1. Double-click on the *Add* Push Button (the top one) to load it into the Property Editor.
2. In the Behavior category, locate the `ActivateCallback` property, then click on the Text Editor (...) button beside it.
3. In the Text Editor that appears, type the following code:

```
swidget    swTodoList;
Widget     wgtTodoList;
char       *buf[3], *cellValue;
int        selectedRow, selectedCol;
int        indxPickList, indxPageNum;

buf[0] = UxGetText(XrtDateField1);
buf[1] = UxGetText(XrtIntField1);
buf[2] = UxGetText(XrtStringField1);

/* get the current state of things */

indxPickList = UxGetXrtGearPickListIndex_Combo(XrtGearCombo1);
indxPageNum = UxGetXrtGearActivePageNumber_TabM(XrtTabM1);
```

```

/* make sure we are dealing with the target todo list,
 * which might not be the same as the one currently selected
 * in the tab manager
 */

if (indxPickList != indxPageNum)
{
    UxPutXrtGearActivePageNumber_TabM(XrtTabM1, indxPickList);
}

swTodoList =
    UxFindSwidget(UxGetXrtGearActivePageWidget_TabM(XrtTabM1));
wgtTodoList = UxGetWidget(swTodoList);

/* find next available row */

selectedRow = 0;
cellValue = "";
while (cellValue != NULL)
{
    XtVaSetValues(wgtTodoList, XmNxrtTblContext,
        XrtTblSetContext(selectedRow, 0), NULL);
    XtVaGetValues(wgtTodoList, XmNxrtTblCellValueContext,
        &cellValue, NULL);
    selectedRow++;
}

selectedRow--;

/* apply the values */

if (selectedRow == UxGetXrtTblNumRows(swTodoList))
{
    /* have to append a new row */

    XrtTblAddRows(wgtTodoList, selectedRow, 1, TRUE,
        (void **)&buf, 3);
}

```

```
else
{
    /* replace values in existing row */

    for (selectedCol = 0; selectedCol < 3; selectedCol++)
    {
        XtVaSetValues(wgtTodoList, XmNxrtTblContext,
                    XrtTblSetContext(selectedRow, selectedCol),
                    XmNxrtTblCellValueContext, buf[selectedCol], NULL);
    }
}

/* display the row that was affected */

XrtTblTraverseToCell(wgtTodoList, selectedRow, 0, TRUE);
```

This code inserts the priority from the Spin Box, the date from the Date Field, and the description from the Text Field, at the bottom of the Table.

4. Load the *Replace* Push Button (the middle one) into the Property Editor and repeat the process, this time typing the following code:

```
swidget    swTodoList;
Widget     wgtTodoList;
char       *buf[3], *cellValue;
int        selectedRow, selectedCol;
int        indxPickList, indxPageNum;

buf[0] = UxGetText(XrtDateField1);
buf[1] = UxGetText(XrtIntField1);
buf[2] = UxGetText(XrtStringField1);

/* get the current state of things */

indxPickList = UxGetXrtGearPickListIndex_Combo(XrtGearCombo1);
indxPageNum = UxGetXrtGearActivePageNumber_TabM(XrtTabM1);

swTodoList =
    UxFindSwidget(UxGetXrtGearActivePageWidget_TabM(XrtTabM1));
wgtTodoList = UxGetWidget(swTodoList);
XrtTblGetCurrentCell(wgtTodoList, &selectedRow, &selectedCol);
```

```

/* if the selected todo list is not the same as the target
 * todo list, we need to delete the selected row in the current
 * list, and find the appropriate row to use in the target list.
 */

if (indxPickList != indxPageNum)
{
    /* delete selected row */

    XrtTblDeleteRows(wgtTodoList, selectedRow, 1, TRUE);

    /* switch to the page corresponding to the selection in the
    combo box */

    UxPutXrtGearActivePageNumber_TabM(XrtTabM1, indxPickList);
    swTodoList =
    UxFindSwidget(UxGetXrtGearActivePageWidget_TabM(XrtTabM1));
    wgtTodoList = UxGetWidget(swTodoList);

    /* find next available row */

    selectedRow = 0;
    cellValue = "";
    while (cellValue != NULL)
    {
        XtVaSetValues(wgtTodoList, XmNxrtTblContext,
            XrtTblSetContext(selectedRow, 0), NULL);
        XtVaGetValues(wgtTodoList, XmNxrtTblCellValueContext,
            &cellValue, NULL);
        selectedRow++;
    }

    selectedRow--;
}

```

```
/* apply the values */

if (selectedRow == UxGetXrtTblNumRows(swTodoList))
{
    /* have to append a new row */

    XrtTblAddRows(wgtTodoList, selectedRow, 1, TRUE,
        (void **)&buf, 3);
}
else
{
    /* replace values in existing row */

    for (selectedCol = 0; selectedCol < 3; selectedCol++)
    {
        XtVaSetValues(wgtTodoList, XmNxrtTblContext,
            XrtTblSetContext(selectedRow, selectedCol),
            XmNxrtTblCellValueContext, buf[selectedCol], NULL);
    }
}

/* display the modified row */

XrtTblTraverseToCell(wgtTodoList, selectedRow, 0, TRUE);
```

This code replaces the selected task in the List Area with the task information from the Edit Area. If your new task has a different category than the one displayed, it deletes the currently selected task, switches tables, then adds your new task to the bottom of the table.

5. Apply your changes by clicking OK in the Text Editor and the Callback Editor and Apply in the Property Editor.
6. Repeat the process one last time for the *Clear* Push Button (the bottom one), typing in the following code:

```
/* reset the edit fields */

UxPutText(XrtDateField1, "");
UxPutText(XrtIntField1, "1");
UxPutText(XrtStringField1, "");
```

This code clears the current information in the Edit area.

7. Remember to click OK in the Text Editor and Callback Editor and to click Apply in the Property Editor.
8. Close the Property Editor.
9. Save your work.

Step #11: Modifying the To Do List Menus

The To Do List menus contain callbacks designed to work with the earlier version of the To Do List you loaded as a start up project. In this step you will modify the *Open* and *Save* menu commands to work with the more sophisticated To Do List.

- File⇒Open opens a saved file of tasks
- File⇒Save saves your current list of tasks in an ASCII file
- File⇒Exit exits from the program
- Help⇒About Application displays information about the interface version

1. Double-click on the File menu in the To Do List.

The Menu Bar Editor appears, loaded with the menu bar for the window, as shown in Figure 4-44.

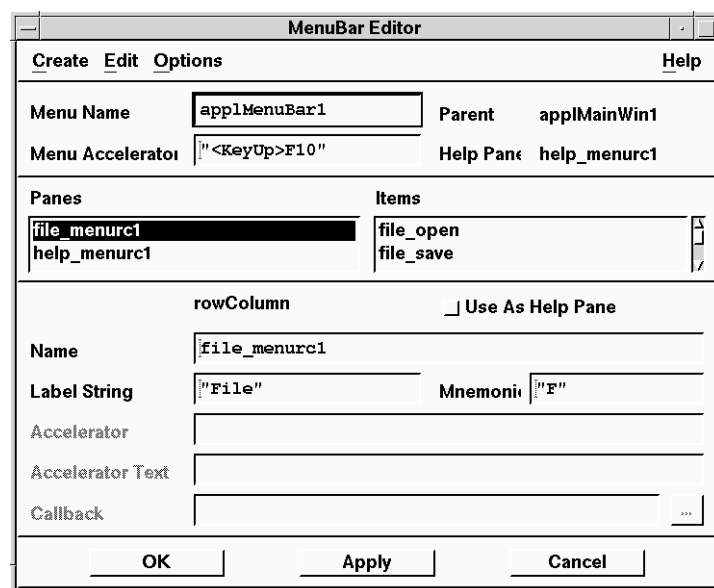


Figure 4-44 Menu Editor

2. Click on `file_open` in the Items area.
3. Click on the Text Editor button beside the Callback property.

4. In the Text Editor that appears, delete the code provided with the start up project.
5. Click OK in the Text Editor.
6. Click on `file_save` in the Items area, then on the Text Editor button beside the Callback property.
7. Once again, delete the code provided with the start up project
8. Click OK in the Text Editor, and Apply in the Menu Bar Editor.

You are now ready to add the new code.

9. Return to the `file_open` item, opening a Text Editor for its Callback property and typing the following code for the callback:

```
FILE      *file;
char      filename[32];
swidget   sw[2];
swidget   swTodoList;
Widget    wgtTodoList;
int       i;
XrtTblCellValues *cellValues;

sw[0] = XrtTabB1;
sw[1] = XrtTabB2;
for (i = 0; i < 2; i++)
{
    /* determine which list we are loading */

    sprintf(filename, "todo%s.out", UxGetLabelString(sw[i]));

    if ( (file = fopen(filename, "r") ) != NULL )
    {
        swTodoList =
        UxFindSwidget (UxGetXrtGearPageWidget_TabM(sw[i]));
        wgtTodoList = UxGetWidget (swTodoList);

        /* clear the todo list */

        XtVaSetValues (wgtTodoList, XmNxrtTblContext,
        XrtTblSetContext (XRTTBL_ALL, XRTTBL_ALL),
        XmNxrtTblCellValueContext, NULL, NULL);
    }
}
```

```
/* get the new data */
if ((cellValues = XrtTblReadAscii(wgtTodoList, file, ','))
    != NULL)
{
    XrtTblSetCellValues(wgtTodoList, 0, 0, cellValues, TRUE);
    XrtTblDestroyCellValues(cellValues);
}

fclose(file);
}
}
```

This code loads the *Home* and *Office* Tables with the contents of the `todoHome.out` and `todoOffice.out` files in your current directory.

10. Click OK in the Text Editor, then on Apply in the Menu Bar Editor.
11. Click on `file_save`, and repeat the process, entering the following code.

```
FILE      *file;
char      filename[32];
swidget   sw[2];
swidget   swTodoList;
Widget    wgtTodoList;
int       i;

sw[0] = XrtTabB1;
sw[1] = XrtTabB2;

for (i = 0; i < 2; i++)
{
    /* find out which todo list we are saving */

    sprintf(filename, "todo%s.out", UxGetLabelString(sw[i]));

    if ( (file = fopen(filename, "w")) != NULL )
    {
        swTodoList =
            UxFindSwidget(UxGetXrtGearPageWidget_TabM(sw[i]));
        wgtTodoList = UxGetWidget(swTodoList);
        XrtTblWriteAscii(wgtTodoList, file, ',', NULL);
        fclose(file);
    }
}
```

This code saves your current To Do Lists in two ASCII files in your current directory: `todoHome.out`, and `todoOffice.out`, respectively.


12. Click OK in the Menu Bar Editor.
13. Save your work.

Step #12: Testing the To Do List

Your interface is now ready to test. UIM/X PDS provides a Test mode that enables you to test your interface instantly, without having to generate code, compile and link the program.

1. Click on the Test icon  in the Project Window.

The Palette, the Property Editor, and any other open editors disappear from your screen.

2. Add new tasks to the task table:
 - Enter a description in the *Task* Field.
 - Use the Spin Box to set priorities.
 - Enter a date in the Date Field. (Note that the Date Field automatically validates your entry.)
 - Choose between a *Home* and *Office* task using the Combo Box.
 - Click on the Tool Bar to add the task to the Table, replace the currently selected task, or clear the description in the Edit Area.
3. Work with existing tasks:
 - Switch between viewing home or office tasks by clicking on the Tab Buttons.
 - Click in the Table to select a task for editing or deletion.
4. Test the menus:
 - Save your current tasks to a file by choosing File⇒Save.
 - Load tasks from a file by choosing File⇒Open.
 - If you try to exit the program by choosing File⇒Exit, the Application Window's default `DeleteResponse` property intercepts the call and shows a message instead. This is the expected behavior in Test mode.
 - Find out about the program by choosing Help⇒About.
5. When you are done switch back to Design Mode by clicking on the Design icon .

The Palette and the Property Editor reappear on your screen.

Step #13: Generating Code

Now that your interface is designed and tested, you can generate its code.

1. Choose the File⇒Generate Project Code As command from the Project Window.

The Generate Code Options window appears, as shown in Figure 4-45.

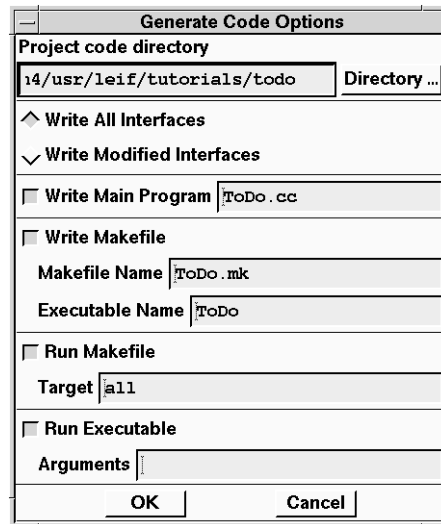



Figure 4-45 Generate Code Options

2. Ensure that the following radio buttons and toggle buttons are selected:
 - Write All Interfaces
 - Write Main Program
 - Write Makefile
 - Run Makefile
 - Run Executable
3. Click on OK.

In a few seconds, your code is generated. Then the code is compiled and linked and the executable is run. You can follow this process by watching the Messages area of the Project Window.

When "Done" appears in the Message Area, the code generation and compilation is complete.

Note – You can also generate the code in one step by clicking the Run icon  .

4. Save your work.

Exiting from UIM/X PDS

To exit from UIM/X PDS, follow these steps.

1. Choose the File⇒Exit command from the Project Window.

Note – If the Project Window is active, you can also exit from the program by pressing Alt + F4, or Alt+F, then X, or by double-clicking on the Window menu button.

2. Click OK to confirm that you want to exit.

Your interface, any open editors, the Palette, and the Project Window all disappear from your screen.

Index

Symbols

`.xdefaults` file, 51
... button (in Property Editor), 87

A

Adjust mouse button, x
Alt key, ix
Alt+F, X (to exit from UIM/X), 99
application defaults, xii

C

Callback Editor, 84
`clientAutoPlace` resource, 36
closing
 UIM/X, 99
compound objects
 definition, viii

D

`DeleteResponse` property, 97
demo and example source code, 27
demos
 multi-product, 33
 XRT/3d, 28
 XRT/field, 28, 29

XRT/gear, 30
XRT/graph, 31
XRT/table, 32

E

Enter key *See* Return key
environment variables
 PATH, 12
 XRTHOME, 12
examples
 XRT/3d, 28
 XRT/gear, 29
 XRT/graph, 30
 XRT/table, 32

F

features
 resize grid, 49
File Selection dialog box, 55

G

Generate Project Code command (in Project Window), 98

I

interface
definition, ix

M

Menu mouse button, x

Motif widget
definition, viii

mouse

- Adjust button, x
- Menu button, x
- Select button, x
- usage, x

mouse buttons, naming conventions
for, vi

moving and resizing widgets, 49

multi-product demos, 33

N

naming conventions
menu options, ix
mouse buttons, vi
Return key, ix
shell prompts, ix

new features, 2

O

object
definition, viii

objects

- Xrt3d, 15
- Xrt3dText, 15
- XrtAligner, 15
- XrtColumn, 16
- XrtComboBox, 16
- XrtCurrencyField, 16
- XrtDateField, 16
- XrtFloatField, 17
- XrtGearCombo, 17
- XrtGLabel, 17
- XrtGraph, 17
- XrtIntField, 18

- XrtLabel, 18
- XrtNodeStyle, 18
- XrtOutliner, 18
- XrtProgress, 19
- XrtPushB, 19
- XrtSpinBox, 19
- XrtStringField, 19
- XrtTabB, 20
- XrtTable, 20
- XrtTabM, 20
- XrtToggleB, 21
- XrtToolbar, 21

OSF/Motif Style Guide, viii

P

palette

- Ux, 14
- XRTPDSMOTIF, 14

PATH environment variable, 12

printing a selected interface, 21

project

- definition, ix

properties

- DeleteResponse, 97

Property Editor

- ... button, 87
- purpose, 56

R

resize grid, 49

resources

- setting, xii

Return key, ix

running UIM/X PDS, 12

S

Save Project command (in Project
Window), 55

saving

- your work in UIM/X, 55

Select mouse button, x

T

To Do List project, 35 to 99
description of GUI, 37
typography in this guide, ix

U

UIM/X
saving your work, 55
UIM/X PDS
running, 12
Ux palette, 14
UxStartingPalettes, 13

W

widget operations
aligning, 53
cancelling, 47
duplicating, 52
widgets
moving and resizing, 49
Window menu button
double-clicking to exit, 99

X

xrt directory, 12
XRT PDS
introduction, 3
XRT/3d, 3 to 4
demos, 28
examples, 28
XRT/field, 4 to 6
demos, 28, 29
XRT/gear, 6 to 7
demos, 30
examples, 29
XRT/graph, 7 to 9
demos, 31
examples, 30
XRT/table, 9 to 11
demos, 32
examples, 32

Xrt3d object, 15
Xrt3dText object, 15
XrtAligner object, 15
XrtColumn object, 16
XrtComboBox object, 16
XrtCurrencyField object, 16
XrtDateField object, 16
XrtFloatField object, 17
XrtGearCombo object, 17
XrtGLabel object, 17
XrtGraph object, 17
XRTHOME environment variable, 12
XRTHOME installation directory, x, 26
XrtIntField object, 18
XrtLabel object, 18
XrtNodeStyle object, 18
XrtOutliner object, 18
XRTPDSMotif palette, 14
XrtProgress object, 19
XrtPushButton object, 19
XrtSpinBox object, 19
XrtStringField object, 19
XrtTabB object, 20
XrtTable object, 20
XrtTabM object, 20
XrtToggleB object, 21
XrtToolBar object, 21

