VXP

Visual X windows Programming Interface

USER AND INSTALLATION MANUAL

Yong Chen
stdyxc05@pip.shsu.edu
s05yoch@gtri.harc.edu

Copyright © 1994 by Yong Chen

CONTENTS

DISCLAIMER 2
DISTRIBUTION 3
Platforms 3
INSTALLATION 4
PROBLEMS, REPORTING BUGS AND GETTING MORE INFORMATION 5
User Manual 6
1. VXP User Interface 6
1.1. Main Menu Interface 6
1.2. Tool Box Interface 10
1.3. Objects' Tree View Box Interface 10
1.4. Object Design Interface 10
1.5. Object Resources Editor Interface 11
1.6. Object Callback Code Editor Interface 12
2. Editing Object's Resources 14
3. Editing Object's Callback Routines 15
4. Making the Application 15
5. VXP Internal File Format (.vxp) 16
5.1. Description 16
5.2. Format 16
5.3. Note 17
5.4. Example 17
Index 20

DISCLAIMER

This is beta release 1.0 of VXP -- Visual X windows Programming Interface using OSF/Motif widget set.

This software is provided as is with no warranty expressed or implied. I hope you find it useful, but I won't be held responsible for any damage that may occur from reading, compiling, installing, using, or even thinking about it.

DISTRIBUTION

VXP is distributed in binary format for different platforms. The following list includes platforms that VXP supports and their package name respectively.

For now, I can only provide VXP binaries for those platforms that I can access. Once I can work out how to protect my source code, or some one can tell me how to do it, I will distribute source code.

If you really want a copy of VXP for your specific platform, I can install it for you once you let me have a temporary access to your platform.

Platforms

1. Hewlett-Packard: HP9000, HP-UX 9.0x

   vxp.HP-UX.tar.gz vxp.HP-UX.tar.Z vxp.HP-UX.tar

2. Silicon Graphics: IRIX 5.x

   vxp.SGI-IRIX.tar.gz vxp.SGI-IRIX.tar.Z vxp.SGI-IRIX.tar

3. Digital Equipment Co.: Alpha AXP, OSF/1

   vxp.DEC-OSF.tar.gz vxp.DEC-OSF.tar.Z vxp.DEC-OSF.tar

INSTALLATION

Since VXP is distributed in binary format, only thing need to be done to install is to unpack the package and make sure the file vxp has the permission to be executed.

For example, to install VXP on HP9000 running HP/UX, the package is vxp.HP-UX.tar.gz or vxp.HP-UX.tar.Z or vxp.HP-UX.tar. Here is how to unpack:

```
% gzcat vxp.HP-UX.tar.gz | tar -xvf -
```

or

```
% zcat vxp.HP-UX.tar.Z | tar -xvf -
```

or

```
% tar xvf vxp.HP-UX.tar
```

This will create a sub-directory `./vxp/` under current directory. The executable for VXP system is "`vxp`". Some samples can be found in `vxp/samples` sub-directory and `vxp/docs` contains documentation in ASCII format and postscript format.

PROBLEMS, REPORTING BUGS AND GETTING MORE INFORMATION

See the file `PROBLEMS` in `vxp/` directory for discussion of various often-reported problems. Please read this file before reporting a problem.

When reporting bugs, problems, suggestions or contributions, please be sure to send them to the right place. I (Yong Chen) am really only interested in the working of VXP itself. Anything related to X, Xt and Motif should go to MIT and OSF respectively.

VXP has a central web page on the World Wide Web, its URL is
`http://www.shsu.edu/~stdyxc05/VXP/`
This page contains latest version, news, information and documentation of VXP.

1. VXP User Interface

The VXP system is a GUI based system. Its user interface consists of six major sub-interface, they are:

☑ *Main menu of the system;*

☑ *Tool box interface;*

☑ *Object's Tree View box interface;*

☑ *Object Design interface;*

☑ *Object Resource editor interface;*

☑ *Object Callback editor interface.*

1.1. Main Menu Interface

1.1.1. File

- ***New***

  To create a new application. The system will pop up a window to ask user to input the name of the new application. When another application is currently open, the system will pop up a dialog window to ask user whether to save the current application or cancel the operation. If user chooses save the old application, the system will save the old application first, then ask user to input the name of the application; if user chooses cancel the operation, the system will do nothing and leave the current application untouched.

- ***Open ...***

  To open an application. The system will pop up a file selection dialog to let user to select the application to edit in the VXP system. The file is in "*.vxp*" format which uses the extension *.vxp*. If user chooses a file not in .vxp format, an error message will pop up and ask user to choose again; if user chooses cancel from the file selection dialog, the system does nothing; if user chooses a legal .vxp file, the system will read the file in and restore the whole application so the user can edit it. When another application is currently opening , the system will save it first without asking user's opinion, and then pop up the file selection dialog.

- ***Close***

  To close current application. The system will save it first, and then close the application. If there is no application currently opening, an error message will pop up to tell the user the error.

- ***Save***

To save current application. The system will pop up a message box to tell user the application has been saved. If there is no application currently opening, an error message will pop up to tell the user the error.

- ***Save as ...***

  Same as save except that the system will pop up a file selection dialog to let user to specify the new file name he or she wants to name the application.

- ***Print***

  To print the application related files, which includes *.vxp file, C source code, makefile, Imakefile*. User need to specify the print command of its system. User can select multiple documents to print by clicking on the check box list.

- ***Exit***

  To exit the VXP system. The system will pop up a confirm box to confirm the operation. The choose of cancel will abandon the operation and return to the system.

1.1.2. Run

- ***Execute application***

  To execute the application within the VXP system. If there is no application in developing, an error message will pop up. If application's executable text image does not exist, the system will pop up an error message box to inform the user that the executable file for the application does not exist, use menu item Compile on the main menu to create it first.

- ***Debug application via xdbx***

  Same as above option, except that this via the xdbx, which is an debug program.

1.1.3. Application

- ***Generate C source code***

  To automatically generate the C source code for the application in developing. A message box will pop up to tell the success of generating the source code. If there is no application in developing, an error message will pop up.

- ***Generate makefile***

  This is to generate makefile.

- ***Generate Imakefile***

  This is to generate Imakefile.

1.1.4. Compile

- ***via makefile***

  To compile and link the application to build the executable text image by using makefile. A message box will pop up to tell the success of the operation. If there is no application in developing, an error message will pop up.

- ***via Imakefile***

Same as via makefile except this is done by using Imakefile.

1.1.5. Widget

- ***Pick an Object***

  To choose current widget tool to pick widget. Current widget tool represents the type of the object to be build next. Pick widget means only grab the object to move or resize..

- ***Primitive Widget ...***

  Contains sub menu which includes all primitive widget types. Choose one of them from the sub menu will set the current widget tool to its type.

- ***Manager Widget ...***

  Contains sub menu which includes all manager widget types. Choose one of them from the sub menu will set the current widget tool to its type.

- ***Shell Widget ...***

  Contains sub menu which includes all shell widget types. Choose one of them from the sub menu will create the shell widget whose parent is current active object. If there is no application in developing, an error message will pop up.

- ***Dialog Widget ...***

  Contains sub menu which includes all dialog widget types. Choose one of them from the sub menu will create the dialog whose parent is current active object. If there is no application in developing, an error message will pop up.

- ***Delete current Object***

  Delete the current active object.

1.1.6. Option

- ***Trace Geometry***

  To toggle the status of trace geometry operation. If the status is on, any change of geometric information (start point, width and height) of the current object will immediately reflect on the status bar (X,Y) and (Width, Height).

- ***Code ...***

  To modify the code related options. The system will pops up a dialog shows below.

  A) Select Application Code Style: choose code style to ANSI/C or K/R C.

  B) modify the name of Global Access Array. Global Access Array is a structure to let user to access the object global in the callback code or other functions. The usage is:

  <Global Access Array name>.<object name>

  For example: `vxpOBJ.pushbutton1; myappOBJ.input_label` etc.

- ***Compiler ...***

  To modify one of the following options for compile and link the program:

  A) Compiler;

B) Compiler option;

C) Include file library option;

D) Link library option.

These option are mainly used by makefile creation.

1.1.7. Window

- ***Show widgets Toolbox***

  To toggle the status of showing the widget toolbox.

- ***Show Objects' Tree Viewbox***

  To toggle the status of showing the object's tree viewbox.

- ***Show Callback Code Editor***

  To show the callback code editor. If there is no application is developing, the callback code editor will not pop up, instead, an error message will pop up.

- ***Dismiss Callback Code Editor***

  To dismiss the callback code editor.

- ***Show Resources Editor***

  To show the resources editor. If there is no application is developing, the resources editor will not pop up, instead, an error message will pop up.

- ***Dismiss Resources Editor***

  To dismiss the resources editor.

1.1.8. Help

- ***All About VXP***

  To connect to VXP central home page which contains latest information of VXP. Access is made possible by NCSA Mosaic via World Wide Web on Internet. The URL address is:

  http://www.shsu.edu/~stdyxc05/VXP/

- ***Contents***

  This is help information about the VXP system, served as user manual. Actually this is a HTML version of this document. The URL address is:

  http://www.shsu.edu/~stdyxc05/VXP/Manual.html

- ***Tutorial***

  A quick on-line tutorial for beginner to get start. A HTML version of tutorial. The URL address is:

  http://www.shsu.edu/~stdyxc05/VXP/Tutorial.html

Item 9 through 13 are status bar items, they are:

1.1.9. Application

Display the name of the application.

1.1.10. Widget Tool Selected

Display the current selected widget tool which will be used to create one on the design space by pointing, clicking and dragging via mouse.

1.1.11. Activate Object

Display the current selected object. A current activate object is served as the parent object of any new objects that created based on it. For example, a manager widget is served as the parent object for some primitive objects on it; a primitive widget may be served as the parent object for a dialog widget or a shell widget.

1.1.12. (X,Y)

Display the upper left corner's coordinate of the current activate object.

1.1.13. (Width, Height)

Display the dimension, width and height, of the current activate object.

1.2. Tool Box Interface

Tool box contains icons of the widget tool that supported by the system, it has the same functionality as the main menu item Widget. But, the tool box only contains primitive widgets, manager widget and pick-widget. It is shown below. This interface is based on supporting all Motif widgets.

1.3. Objects' Tree View Box Interface

View box is a scroll window contains a tree which represents the application's objects' hierarchic structure. Each object is represented by a push button, click on the button will set the current active object as this object.

1.4. Object Design Interface

Object design interface is a what-you-see-is-what-you-get (WYSIWYG) interface. As mentioned above, user can create a widget and then put it to certain place, resize it, move it, delete it, and also pick any one of the widgets of the application. The interface contains following operations:

1.4.1. Pick

Move the mouse pointer into the area of the object that user want to pick as current active object and click the left button or first button on the mouse. The status bar of Active Widget will correspondingly change its value to the name of the widget.

1.4.2. Move

Move the mouse pointer into the area of the object that user want to move its position, press the left button or first button on the mouse, hold and drag the object to desired place and release the finger on the button, drop the object. There will be a rectangle represents the object when moving.

1.4.3. Resize

Resize has eight different sub-operations according to the mouse position.

'Ú ': upper left corner. Move mouse pointer into upper left corner of the object, press the left or first button, hold and drag the object to desired size and release the button. There also will be a rectangle represents the object when resizing.

'Ã ': left side. Operation is the same as upper left corner.

'À': lower left corner. Operation is the same as upper left corner.

'Á': bottom side. Operation is the same as upper left corner.

'Ù': lower right corner. Operation is the same as upper left corner.

' ´': right side. Operation is the same as upper left corner.

'¿': upper right corner. Operation is the same as upper left corner.

'Â': top side. Operation is the same as upper left corner.

1.4.4. Delete

Press key <Delete> on the keyboard or use menu item Delete current widget in main menu item Widget.

1.4.5. Mouse button 3 or right button

Press and release mouse button3 or right button will pop up the resources editor and callback editor and set the current active object in both editor's object select list.

1.5. Object Resources Editor Interface

1.5.1. File

- ***Dismiss Editor***

  Pop down the resources editor.

1.5.2. Help

This contains all items for help.

1.5.3. Select Object

This is a list of all objects' name of the application in a scrolled window. User can scroll back and forth the list, click on the name will select the object as the current active object and the resources editor will dynamically update itself to reflect the information of the object.

1.5.4. Change object's name

When an object is selected as active object, user can change its name by type the new name in the input filed and the system will update it immediately. Note: the system will not check the duplicate name, but it will check basic syntax like no space is allowed in between the name etc. If there is no object selected, this operation will be disabled.

1.5.5. Select A Resource

This is a list of all resources name which the current active object has. User can scroll back and forth the list, click on the item wish to edit. After that user can modify the value of the resource. If there is an old value of this resource, it will be put on the Enter resource value field. (See 2.5.7.) If there are some possible values for choose, they will be put on the Possible Choices field. (See 2.5.6.) If there is no object selected, this operation will be disabled.

1.5.6. Possible Choices

This is a list of all the possible choices of the current selected resource type. User can scroll back and forth the list, click on the choice wish to select. If a choice is selected, it will be put on the Enter resource value field. (See 2.5.8.) But, no immediate change will occur until user click on the Apply button (see 2.5.9). If there is no resource selected, this operation will be disabled.

1.5.7. Enter resource value

This is the field that let user to input or modify the resource value. After edit the value, press <Return> to activate the object to change the resource to new value. If there is no resource selected, this operation will be disabled.

1.5.8. Apply

To activate the object to change the selected resource to new value edited in the Enter resource value field. If there is no resource selected, this operation will be disabled. If the Enter resource value field is blank, this operation does nothing.

1.5.9. Delete

To delete the resource selected from the object, which means to use the default value. If there is no resource selected, this operation will be disabled.

1.5.10. Dismiss

To pop down the resource editor, the same as the item Dismiss Editor in menu File.

1.6. Object Callback Code Editor Interface

1.6.1. File

- ***New***

  To clear the callback code and reconstruct a template for user to editor. If no object and callback type are selected, this operation is disabled. An error message will pop up.

- ***Include ...***

  To load a file into the code edit area (see 2.6.12). A file selection dialog will pop up to let user to select the file to load.

- ***Save to Object***

  To save the contents in edit area (see 2.6.12) into the object.

- ***Delete Current Callback***

To delete the callback from the current object's callback. And it will reconstruct a code template of the resource in case the user want to write some code again.

- ***Dismiss Editor***

  To pop down the callback editor.

1.6.2. Edit

- ***Cut***

  To cut the selected area of code into cut buffer.

- ***Copy***

  To copy the selected area of code into buffer for later use.

- ***Paste***

  To paste the content in the most recently buffer to the position of the cursor.

- ***Clear***

  To clear everything in the code area.

1.6.3. Search

- ***Find Next***

  To find the next occurrence of the Search pattern (see 2.6.10).

- ***Show All***

  To highlight all the occurrences of the Search pattern (see 2.6.10).

- ***Replace Text***

  To replace the text which has the same pattern as the Search pattern (see 2.610) with Replace Pattern (see 2.6.11). After the operation, a message box will pop up to tell how many replacement has been made.

- ***Clear***

  Unhighlight all the occurrence of Search pattern.

1.6.4. Help

This contains items for help operation.

1.6.5. Select Object

The same as Resource Editor's Select Object.

1.6.6. Select Callback Type

This is a list of all callback type names which the current active object has. User can scroll back and forth the list, click on the item wish to edit . After that, user can modify the code for that callback. If it is already exist in the object, the old code will be put into code edit area (see 2.6.12). If there is no object selected, this operation will be disabled.

1.6.7. Client Data Type;

1.6.8. Client Data Name;

1.6.9. Client Data Value

These three fields are for the client data which is passed to callback functions when it is invoked. Changes in any one of these three fields will cause the callback function prototype to update to the changes in the code edit area (see 2.6.12). Note: only after pressing <Enter> will activate the change .

1.6.10. Search Pattern;

1.6.11. Replace Pattern

These two fields are for menu Search's operations. If Search Pattern is blank, all operations in menu Search are disabled.

1.6.12. Code Edit Area

This is the area to actually enter the code for the callback. The function prototype is generated and updated automatically by the system. Any change will cause the system to save the updated code to the object. If no object and callback type are selected, this area is disabled to enter anything.

1.6.13. Code status and memo bar

Two pieces of information are shown in this bar. They are:

- ***Code Style***

  Display whether the code style is ANSI/C or old UNIX C.

- ***Object's Global Access Array name***

  Display the name and usage of the Global Access Array.

These two fields will be updated if user has changed them via Code option in main menu Option.

2. Editing Object's Resources

VXP's *resources editor* is an easy to use interface to interactively edit object's resources. There some points that need to be addressed here:

1. VXP's resources editor will list all the resources that an object may have, but not all of them is editable. For example, some resources require an memory pointer or an data structure as input, this cannot not be done via resources editor; some resources only take effect at the time the object is created, so even though user changes its value via resources editor, it will not show the effect on the object.

2. VXP only uses X11 bitmap as its picture format.

3. Not all color can be converted, since they are based on the current colormap of the X server. Future version of VXP will have a color palette to select color.

4. Fonts should use the full format string. Use xfontsel or xlsfonts program to get the full string. Future version of VXP will provide a font selection box.

5. Use different order to modify couple resources may get different result. For example, create a picture for a draw button first and change the background color of the button later will not change the background color of the picture; but not if do it in reverse order the background color will be the same.

3. Editing Object's Callback Routines

VXP's *callback code editor* provides a nice interface to editor object's callback routines as well as application's general code. There are couple points need to be noted here:

1. VXP provides so called *<general>* object in callback code editor. It is not a real object but a fake one to represent all the code that is not part of the object's callback routines. It contains following parts:

- o application include files: include head files other than widgets' head files;
- o application global definitions: global variables, macros, function prototypes;
- o application initial routine: routine to initialize application after building all objects;
- o application functions, routines: other functions and routines of the application.

2. Global Access Array is a structure to let program to access an object any where in the application. It is a pointer to object's Widget ID required by most X toolkit functions.

3. After modifying Global Access Array name, object's name, client data and other functions in <general> object, make sure the changes be updated through out the whole application. For example, if a push button's name has been changed from pushbutton1 to quit_button, all previous references to pushbutton1 become invalid and should be changed to quit_button.

4. Making the Application

VXP will automatically generate C source code as well as makefile, Imakefile of the application. There are several points need to be addressed here:

1. VXP provides both ANSI C and K&R C style for code generation, however, VXP will not check to code style of callback routines and general part of the code. It is the application's responsibility to write the code in selected code style.

2. Different systems may have different file system structure, before generating makefile, be sure to check the compiler options.

3. If the system doesn't have `xmkmf` script, do not use compile via Imakefile option.

5. VXP Internal File Format (*.vxp*)

5.1. Description

VXP internal file Contains all the necessary information of the developing application: 1) objects information; 2) code style and makefile option; 3) application related code segments. Most application information is input via user interface, however, you can write a file follows the format to create an application. This file is pure ASCII file.

5.2. Format

Convention: **Bold words** is required key words; <...> indicates actual value or contents, ("<" and ">" are not keywords); [...] means the contents is optional.

<Copyright information>
Application: <application name>
Code Style: <code style (ANSI or OLD C) >
Objects Global Access Array name: <array name>
Compiler: CC=
<compiler name>
Compiler option: CFLAGS =
<compiler options>
Include file library option: LIBS =
<include file library options>
Link library option: IFLAGS =
<link library options>
Application's general include source code:
<application's general include part>
Application's global definition source code:
<application's global definitions>
Application's initial routine source code:
<application's initialization routine>
Application's functions source code:
<application's other functions, routines>
!VXP! begin-object
<object class type} **:** <object name> ({number of objects in the same class>)
(<upper left X position>**,** <upper left Y position>) (<width> **,** <height>) <parent name>
[**!VXP! begin-resources**
<resource name> **:** <resource value>
:
!VXP! end-resources]
[**!VXP! begin-callback**
<callback name> **: client_data(** <client_data type> <client_data name> <client_data value> **)**

<callback source code segment>
!VXP! end-callback]
:
!VXP! end-object :

5.3. Note

- Copyright is a block of text to describe the copyright information.
- Application name is the name of the application.
- Code style means that the source code of the application follows the ANSI C standard or old UNIX C style.
- Objects Global Access Array name is the name of an array which let the application developer to access each objects globally.
- Compiler, Compiler option, Include file library option, Link library option are used to create the makefile for the application. Actual values are machine and system dependent.
- Application's general include part, global definitions, initial routine, and functions are code segments that is not directed related to the objects of the application.
- Object class type is the widget class type of the object. Each object is an instantiation of a widget class.
- Object name is the name of object.
- Upper left X-Y position, width, height are geometric value of the object.
- Parent name is name the object's parent.
- Resource name is the name or type of the object's resource.
- Resource value is the actual value of the resource in string fashion.
- Callback name is the name of the object's callback. Callback is an action routine registered to the object to response to certain event. The name describes what event it is.
- client_data type, name, value are information related to the callback routine. Client data is the information that passes to callback routine when callback is activated. Type means the data type of the client data, it follows the C syntax; name the just the variable name; value is the actual contents of the client data. The default value is set by the system.
- Callback source code segment are the C source code of the callback routine.

5.4. Example

```
/*************************************************/
/*                                               */
/*    Visual X windows Programming Interface     */
/*    ======================================     */
/*                                               */
/*      Copyright (C) 1994  by  Yong Chen        */
/*                                               */
```

```
/***********************************************/

Application:      untitled
Code Style: ANSI
Objects Global Access Array name: vxpOBJ
Compiler: CC =
 gcc

Compiler option: CFLAGS =


Include file library option: LIBS =
 -L/usr/lib/Motif1.2 -lXm -L/usr/lib/X11R5 -lXt -lX11

Link library option: IFLAGS =
 -I/usr/include/X11R5 -I/usr/include/Motif1.2

Application's general include source code:
/* application's include part */


Application's global definitions source code:
/* application's global definitions */


Application's initial routine source code:
/* application's initial routine */
void vxpInitial()
{

}


Application's functions source code:
/* application's functions */


!VXP! begin-object
applicationShell : untitled ( 0 )
( 170 , 205 )      ( 300 , 200 )     untitled
!VXP! end-object

!VXP! begin-object
BulletinBoard : BulletinBoard1 ( 1 )
( 198 , 192 )      ( 300 , 200 )     untitled
!VXP! end-object

!VXP! begin-object
PushButton : PushButton1 ( 1 )
( 84 , 129 )       ( 126 , 53 )      BulletinBoard1
!VXP! begin-resources
labelString : EXIT
labelType : XmPIXMAP
labelPixmap : Excl
!VXP! end-resources
!VXP! begin-callback
activate :  client_data( XtPointer client_data NULL )
void PushButton1_activate_cb
(Widget w, XtPointer client_data, XmPushButtonCallbackStruct *cbs)
{
      void exit();

      printf("Good bye!\n");
```

```
      exit(0);
}

!VXP! end-callback
!VXP! end-object

!VXP! begin-object
Label : Label1 ( 1 )
( 23 , 30 )        ( 246 , 42 )      BulletinBoard1
!VXP! begin-resources
background : white
labelString : Hello World!
fontList : 9x15bold
foreground : red
!VXP! end-resources
!VXP! end-object

!VXP! begin-object
Separator : Separator1 ( 1 )
( 15 , 78 )        ( 270 , 25 )      BulletinBoard1
!VXP! end-object
```

Index

(Width, Height), 10
(X,Y), 10
.vxp, 6, 16
Activate Object, 10
All About VXP, 9
Alpha AXP, 3
ANSI C, 15
Application, 7, 9
application functions, routines, 15
application global definitions, 15
application include files, 15
application initial routine, 15
Apply, 12
bottom side, 11
button, 10
C source code, 7
callback code editor, 9
central home page, 9
Change object's name, 11
Clear, 13
Client Data Name, 13
Client Data Type, 13
Client Data Value, 13
Close, 6
Code, 8
Code Edit Area, 14
code generation, 15
Code status and memo bar, 14
Code Style, 14
code style, 15
color, 14
colormap, 14
Compile, 7
Compiler, 8
Compiler option, 9
Contents, 9
coordinate, 10
Copy, 13
current active object, 8
current selected widget tool, 10
current widget tool, 8
Cut, 13
Debug application via xdbx, 7
DEC, 3
Delete, 11, 12
Delete Current Callback, 12
Delete current Object, 8
Description, 16
Dialog Widget, 8
Digital Equipemnt Co., 3
Dismiss, 12
Dismiss Callback Code Editor, 9
Dismiss Editor, 11, 13
Dismiss Resources Editor, 9
DISTRIBUTION, 3
Edit, 13
Editing Object's Resources, 14
Enter resource value, 12
Example, 17
executable text image, 7
Execute application, 7
Exit, 7
File, 6, 11, 12
Find Next, 13
first button, 10
Fonts, 15
Format, 16
general, 15
Generate C source code, 7
Generate Imakefile, 7
Generate makefile, 7
Global Access Array, 15
Help, 9, 11, 13
Hewlett-Packard, 3
HP, 3
HP-UX, 3
icons, 10
Imakefile, 7
Include, 12
Include file library option, 9
INSTALLATION, 4
Internet, 9
IRIX, 3
K&R C, 15
left button, 10
left side, 11
link, 7
Link library option, 9
lower left corner, 11
lower right corner, 11
Main Menu, 6
Main menu, 6
makefile, 7
Making the Application, 15
Manager Widget, 8

Motif, 2
mouse, 10
Move, 10
New, 6, 12
Note, 17
Object Callback Code Editor Interface, 12
Object Callback editor, 6
Object Design, 6
Object Design Interface, 10
Object Resource editor, 6
Object Resources Editor Interface, 11
Object's Global Access Array name, 14
Object's Tree View box, 6
Objects' Tree View Box Interface, 10
Open, 6
Option, 8
OSF, 2
OSF/1, 3
Paste, 13
Pick, 10
Pick an Object, 8
pick widget, 8
Platforms, 3
Possible Choices, 12
Primitive Widget, 8
Print, 7
Replace Pattern, 14
Replace Text, 13
Resize, 11
resources editor, 9
right button, 11
right side, 11
Run, 7
Save, 6
Save as, 7
Save to Object, 12
Search, 13
Search Pattern, 14
Select A Resource, 11
Select Callback Type, 13
Select Object, 11, 13
SGI, 3
Shell Widget, 8
Show All, 13
Show Callback Code Editor, 9
Show Resources Editor, 9
Show widgets Toolbox, 9
Silicon Graphics, 3
status bar, 8, 9
Tool box, 6
Tool Box Interface, 10
toolbox, 9
top side, 11
Trace Geometry, 8
Tutorial, 9
unpack, 4
upper left corner, 11
upper right corner, 11
User Interface, 6
User Manual, 6
via Imakefile, 7
via makefile, 7
viewbox, 9
Visual X windows Programming Interface, 2
VXP, 2
VXP Internal File Format, 16
web page, 5
Widget, 8
Widget Tool Selected, 10
Window, 9
World Wide Web, 5, 9
X11 bitmap, 14
xmkmf, 16