VXP

Tutorial

1. Introduction 1
2. Start to create a new application 1
    2.1. Start the VXP system 1
    2.2. Start to create a new application 1
3. Build the user interface 2
    3.1. Create a manager widget 2
    3.2. Create the label widget 3
4. Modify the resources 4
    4.1. Open the Resources Editor 4
    4.2. Change the resources for "Label1" 4
    4.3. Add exit button and a decoration line 6
    4.4. About Object's Tree View Box 7
5. Add event action code 7
    5.1. Open the Callback Code Editor 7
    5.2. Add event--"push" action code for the exit button 7
6. Generate application related files 8
    6.1. Generate .vxp file, save the application 8
    6.2. Generate C source code 9
    6.3. Generate makefile and Imakefile 9
    6.4. Generate the executable text file 9
    6.5. Run the application 9
Appendix. Listings 10
    A-1 "hello.vxp" file 10
    A-2 "hello.c" file 11
    A-3 makefile for hello.c 14
    A-4 Imakefile for hello.c 14

All rights reserved Written and Revised by Yong Chen 1994

Tutorial

1. Introduction

The basic steps to develop an X windows application is to design the user interface first and write the code to combine user interface part, event response action part and other non-interface related part. Using VXP to develop an application involves iteration of serveral steps:

- design user interface visually;
- set properties or resources for each GUI component of the application;
- fill out action code of events driven by each GUI component of the application;
- write or include non-interface related code of the application;
- generate source code, compile, link and execute the application;

To use VXP, you should have some idea of the X windows environment, C programming language and basic concept of GUI.

In this tutorial, we will build a tradition hello program to print out "*Hello, X World!*", it will look like:

2. Start to create a new application

2.1. Start the VXP system

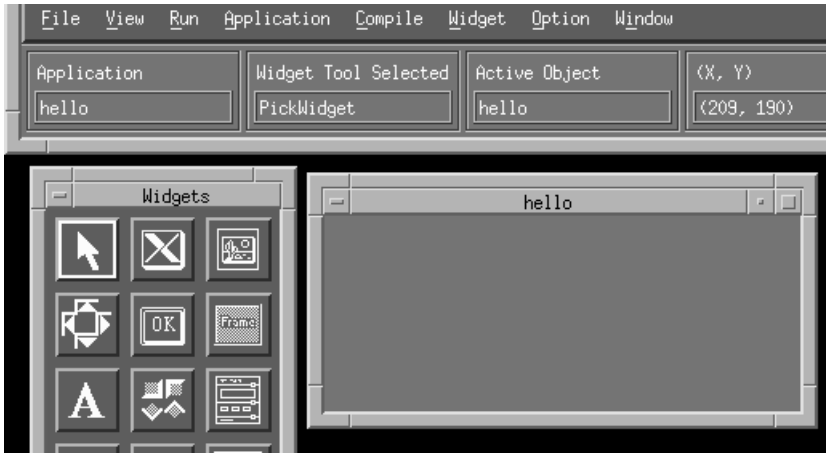
If VXP system has been installed on your machine and in your PATH, you may start the system by simply type vxp on the shell command line like:

```
/users/joe/x $ vxp
```

2.2. Start to create a new application

Once you have VXP system running, the main menu will pop up. Choose sub-item *New* from the main menu item *File* to start the new application. After doing that, a dialog box will pop up to ask you to input the name of the applcation.

Type the name of the application -- in this case is "*hello*" -- in the input field, and click button "*OK*". The system will start to create a new application named "hello" and it will give you the top level shell of the application. See the following figure.



Now, you can build your application by creating the user interface first.

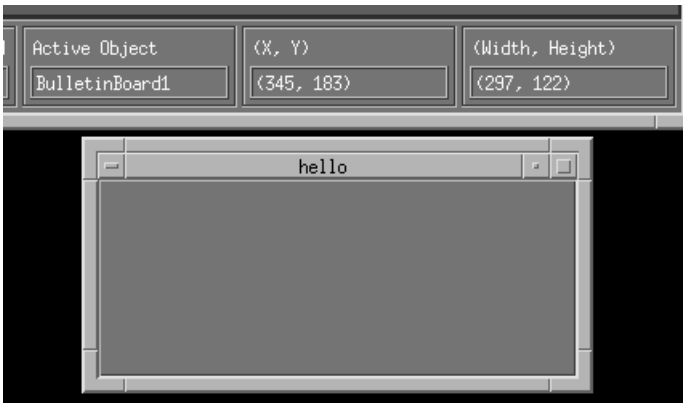
3. Build the user interface

3.1. Create a manager widget

An widget is an object to do a specific job in the user interface. User interface design is just to create and arrange those widgets. In the example program "hello", we will put a label which display the sentence "*Hello, X World!*", and put a button which will servered as the exit button. To make it more fancy looking, we are going to add a line and put a picture on the button.

There are two ways to select a widget tool to create the user interface, one is to choose it from main menu item *Widget*, another is click the tool icon directly in the tool box. Following is a part of the tool box which contains the widget tool we will use in this program.

Now, we first build a manager widget which is used as a parent widget that can contains primitive widgets like the label and the button. We choose bulletin board as the parent manager widget.



3.2. Create the label widget

Click on the label widget icon and use the mouse to point on the start position, drag to desired size and drop it to create the label.

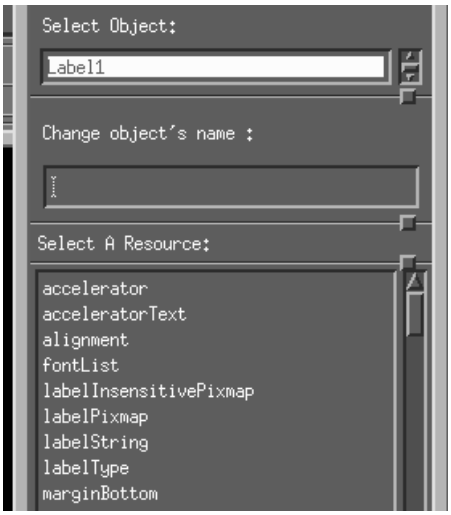
Now, you can modify the resources of the objects of the user interface.

4. Modify the resources

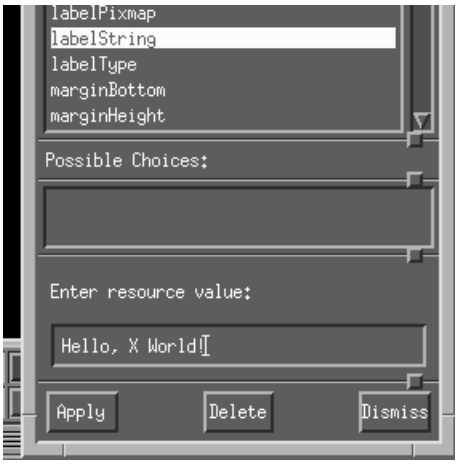
4.1. Open the Resources Editor

A widget has a set of resources to control its appearance and behavior. We can use the VXP system's Resources Editor to modify and edit those resources. There are two ways to open the Resources Editor. One way is select "*Show Resources Editor*" in the main menu item "*Window*"; another way is pointint the mouse in the area of the application, and click the right button (or third button). It can be closed or dismissed by select "*Dismiss Resources Editor*" in either the main menu item "*Window*" or in the resources editor's menu item "*File*", also push the "*Dismiss*" button on the bottom of the resources editor does the same thing.

4.2. Change the resources for "Label1"

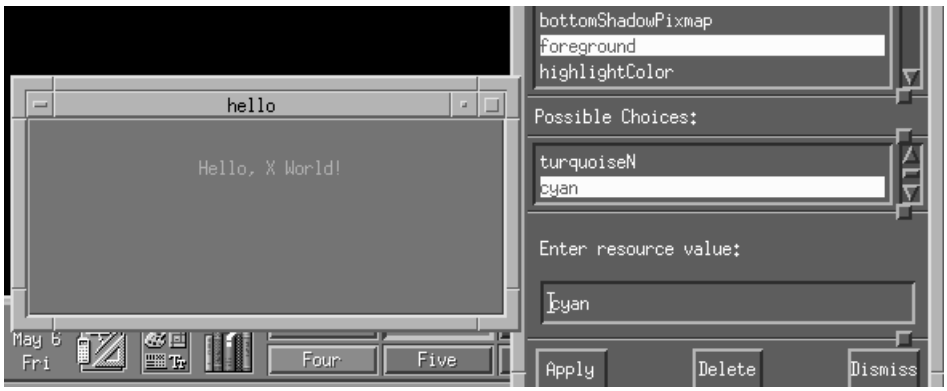


First you need select an object to edit from the top field "*Select Object*", scroll back and forth the list and click on the object that you want to edit. This will set the resources editor to bring out the related resources which will be listed on the field "*Select A Resource*".



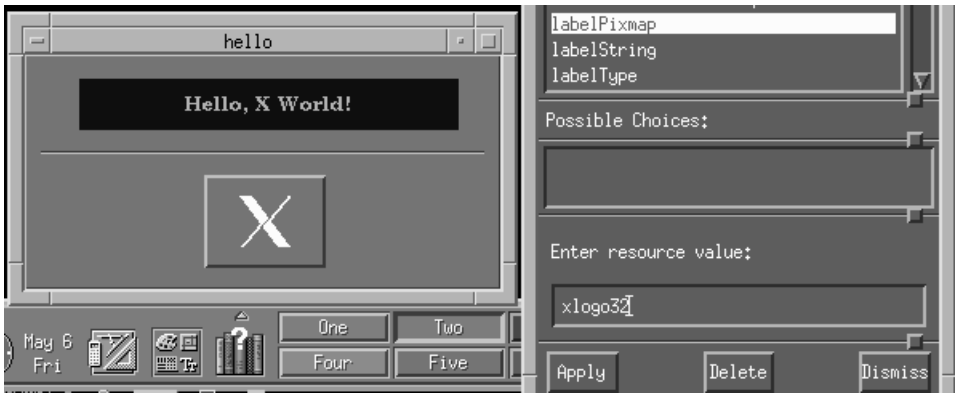
We now select the resource "labelString" to change its value to the sentence "Hello, X World!". Hit Enter or push the "*Apply*" button to change it.

We can also change some other resources, like foreground, background color, font etc. Note: some resources have some possible values which will displayed in the "*Possible Choices*" field, you can scroll the list and click on the choice that desired. See following pictures.



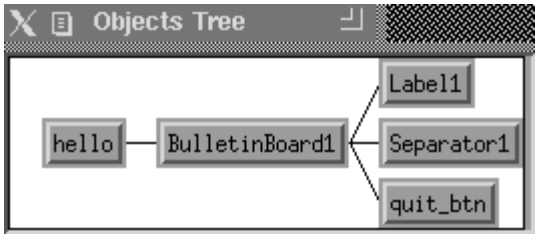
4.3. Add exit button and a decoration line

Use the same way, you can add the exit button with a xlogo on it and a decoration line.



4.4. About Object's Tree View Box

VXP provides a scrolled window to display the object's hierarchic structure. Each object is represented by a push button in the view box. User can click on the button to pick that object as current active object. After building the interface, the view box will be look like:



Now, you can add the necessary code of the response action for certain event of the widget.

5. Add event action code

5.1. Open the Callback Code Editor

Each widget has a set of events associated with it, also the widget has its own default action routines to handle different events. User can add his/her own code to response to certain events. VXP system provides the Callback Code Editor to allow you to edit the your code for the event action.

There are two ways to open the Resources Editor. One way is select "*Show Callback Code Editor*" in the main menu item "*Window*"; another way is pointint the mouse in the area of the application, and click the right button (or third button). It can be closed or dismissed by select "*Dismiss Callback Code Editor*" in either the main menu item "*Window*" or in the callback code editor's menu item "*File*".

5.2. Add event--"push" action code for the exit button

First you need select an object to edit from the top field "*Select Object*", --in this case choose object "*Pushbutton1*"--scroll back and forth the list and click on the object that you want to edit. This will set the callback code editor to bring out the related events which will be listed on the field "*Select Callback Type*".

Here we select the event "*activate*" which means when the button is being pushed. Now, the callback code template is showed at the bottom edit field. We can move mouse to that field and input the code we want to respond to the "*push*" event. Since it is an exit button, we put a print code to say "*Goodbye!*" and terminate the program by using the *exit()* function.

The callback code editor will automatically update the code to the object. If you want to make sure you have the code saved to the object, use the item "*Save to object*" in the menu item "*File*".

Now, you have finished develop the application "hello". All you have to do is just to ask the VXP system to automatically generate all the files related to the application, like C source code, makefile, Imakefile, and of course executable text file.

6. Generate application related files

6.1. Generate *.vxp* file, save the application

Use "*Save*" or "*Save as ...*" in main menu item "*File*" to save the application in VXP's internal

format to disk. *Note:* the filename must use extension "*.vxp*". In this case, we save it as "*hello.vxp*". After saving the application, the system will pop up a message dialog tell you that the application has been saved.

6.2. Generate C source code

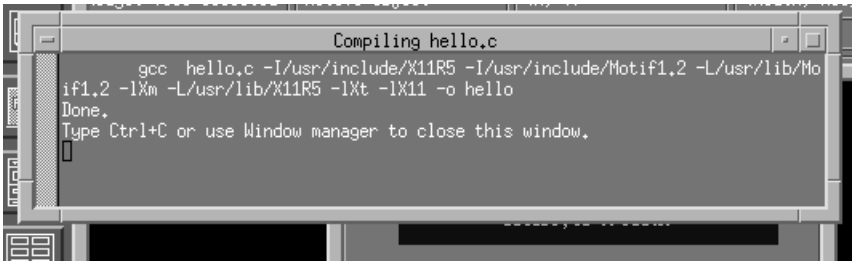
Use "*Generate C source code*" in main menu item "*Application*" to generate and save the C source code for the application.

6.3. Generate makefile and Imakefile

Use "*Generate makefile*" and "*Generate Imakefile*" in main menu item "*Application*" to generate and save makefile and Imakefile for the application respectively.

6.4. Generate the executable text file

In other word, this is to compile and link the application into executable text image. Use "*via makefile*" or "*via Imakefile*" in main menu item "*Compile*" to generate the executable text file for the application. Type Ctrl+C or use window manager to kill that window.



6.5. Run the application

The VXP system supports to execute the application without leaving VXP system. Once you have generate the executable text file for the application, you can select "*Execute the application*" from main menu "*Run*" to try the application within the VXP system.

OK, these are all the steps required to build a X windows application using VXP system. You can iterate the whole process to modify or maintain your application. Any of the steps is independent from each others, so you can step into any specific step as you wish.

This finishes the tutorial for VXP -- *Visual X windows Programming Interface*

Appendix. Listings

A-1 *"hello.vxp"* file

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

Application:      hello
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 : hello ( 0 )
( 214 , 258 )      ( 297 , 146 )     hello
!VXP! end-object

!VXP! begin-object
BulletinBoard : BulletinBoard1 ( 1 )
( 0 , 0 )    ( 297 , 146 )     hello
!VXP! end-object

!VXP! begin-object
PushButton : PushButton1 ( 1 )
( 110 , 73 )       ( 80 , 62 )       BulletinBoard1
!VXP! begin-resources
labelPixmap : xlogo32
labelType : XmPIXMAP
!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("Goodbye!\n");
      exit(0);
}
!VXP! end-callback
!VXP! end-object

!VXP! begin-object
Separator : Separator1 ( 1 )
( 10 , 57 )        ( 277 , 9 )       BulletinBoard1
!VXP! end-object

!VXP! begin-object
Label : Label1 ( 1 )
( 34 , 15 )        ( 237 , 32 )      BulletinBoard1
!VXP! begin-resources
fontList : -adobe-new century schoolbook-bold-r-normal--12-120-75-75-p-
77-iso8859-1
background : navy
foreground : cyan
labelString : Hello, X World!
!VXP! end-resources
!VXP! end-object
```

A-2 "*hello.c*" file

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

/* application C source code */
```

```
/* include part */

/* X11 and Motif includes part */
#include <stdio.h>
#include <Xm/Xm.h>
#include <Xm/Label.h>
#include <Xm/Separator.h>
#include <Xm/PushB.h>
#include <Xm/BulletinB.h>

/* application's include part */


/* global definition part */

/* Defines object's Global Access Array */
typedef struct {
      Widget hello;
      Widget BulletinBoard1;
      Widget PushButton1;
      Widget Separator1;
      Widget Label1;
} _GAArray;
static _GAArray vxpOBJ;

/* application's global definitions */


/*
 *main()
 */
void
main(int argc, char **argv)
{
      /* variable & prototype declaration */
      XtAppContext app_context;
      void vxpInitial();
      Widget hello;
      Widget BulletinBoard1;
      Widget PushButton1;
      void PushButton1_activate_cb();
      Widget Separator1;
      Widget Label1;

      /* create topLevel widget, initialize application */
      hello = XtVaAppInitialize(&app_context, "Hello",
            NULL, 0, &argc, argv, NULL, NULL);
      vxpOBJ.hello = hello;

      /* widgets creation */

      BulletinBoard1 = XtVaCreateWidget("BulletinBoard1",
            xmBulletinBoardWidgetClass,
            hello, NULL);
      XtVaSetValues(BulletinBoard1, XmNx, 0, XmNy, 0,
            XmNwidth, 297, XmNheight, 146,
            NULL );
      vxpOBJ.BulletinBoard1 = BulletinBoard1;

      PushButton1 = XtVaCreateWidget("PushButton1",
            xmPushButtonWidgetClass,
            BulletinBoard1, NULL);
      XtVaSetValues(PushButton1, XmNx, 110, XmNy, 73,
```

```
            XmNwidth, 80, XmNheight, 62,
            XtVaTypedArg, XmNlabelPixmap, XmRString,
            "xlogo32", 8,
            XtVaTypedArg, XmNlabelType, XmRString,
            "XmPIXMAP", 9,
            NULL );
      XtAddCallback(PushButton1, XmNactivateCallback,
            PushButton1_activate_cb, (XtPointer) NULL);
      XtManageChild(PushButton1);
      vxpOBJ.PushButton1 = PushButton1;

      Separator1 = XtVaCreateWidget("Separator1",
            xmSeparatorWidgetClass,
            BulletinBoard1, NULL);
      XtVaSetValues(Separator1, XmNx, 10, XmNy, 57,
            XmNwidth, 277, XmNheight, 9,
            NULL );
      XtManageChild(Separator1);
      vxpOBJ.Separator1 = Separator1;

      Label1 = XtVaCreateWidget("Label1",
            xmLabelWidgetClass,
            BulletinBoard1, NULL);
      XtVaSetValues(Label1, XmNx, 34, XmNy, 15,
            XmNwidth, 237, XmNheight, 32,
            XtVaTypedArg, XmNfontList, XmRString,
            "-adobe-new century schoolbook-bold-r-normal--12-120-75-75-
p-77-iso8859-1", 73,
            XtVaTypedArg, XmNbackground, XmRString,
            "navy", 5,
            XtVaTypedArg, XmNforeground, XmRString,
            "cyan", 5,
            XtVaTypedArg, XmNlabelString, XmRString,
            "Hello, X World!", 16,
            NULL );
      XtManageChild(Label1);
      vxpOBJ.Label1 = Label1;

      XtManageChild(BulletinBoard1);

      /* Create windows for widgets and map them. */
      XtRealizeWidget(hello);
      vxpInitial();
      /* Loop for events. */
      XtAppMainLoop(app_context);
}


void PushButton1_activate_cb
(Widget w, XtPointer client_data, XmPushButtonCallbackStruct *cbs)
{
      void exit();

      printf("Goodbye!\n");
      exit(0);
}


/* application's initial routine */
void vxpInitial()
{

}
```

```
/* application's functions */
```

A-3 makefile for hello.c

```
PROGRAMS = hello
CFLAGS =
LIBS = -L/usr/lib/Motif1.2 -lXm -L/usr/lib/X11R5 -lXt -lX11
IFLAGS = -I/usr/include/X11R5 -I/usr/include/Motif1.2
CC = gcc

all: $(PROGRAMS)

.c: ; $(CC) $(CFLAGS) $@.c $(IFLAGS) $(LIBS) -o $@
```

A-4 Imakefile for hello.c

```
#define PassCDebugFlags 'CDEBUGFLAGS=$(CDEBUGFLAGS)'
CC =  cc

SRCS = hello.c
OBJS = hello.o

PROGRAMS = hello

all:: $(PROGRAMS)

LOCAL_LIBRARIES = $(XMLIB) $(XTOOLLIB) $(XLIB)

NormalProgramTarget(try, $(OBJS),,$(LOCAL_LIBRARIES),)
```