

X/MOTIF

Amarra, Christine
Dueñas, Miko
Lucena, Josh
Nieva, Jeanie
Sevilla, Mae
Vy, Sally

What is X/Motif?

- First graphical user interface offering user-oriented PC-style behavior and screen appearance for applications running on systems that support the X Window System.
- It is also the base graphical user interface toolkit for CDE (Common Desktop Environment)
- It enables the development of cross-platform and network-based applications.

History

- Early windows systems ran only on the developers' OS, thus the X Window system was designed.
- Two interface toolkits were created for X Window systems—Motif and Open Look/Open Windows.
- Motif is a product of the Open Software Foundation (OSF), which originally included DEC, IBM and HP.
- Motif eventually became the standard GUI toolkit for UNIX.

Initializing the Motif application

- `#include <Xm/Xm.h>`
- Calling `XtVaOpenApplication()`
 - The application is connected to the X display.
 - The application is parsed for the standard X command-line arguments.
 - Resources are set up.
 - A top level window / shell, which handles the application's interaction with the window manager, is created

Initializing the Motif application (con't)

```
Widget toplevel;  
XtAppContext app;  
toplevel = XtVaOpenApplication  
    (&app, "name", NULL, 0, &argc,  
     argv, NULL,  
     sessionShellWidgetClass, NULL);
```

Widgets

- The basic building block for the GUI.
- Some widgets in Motif are:
 - Text/picture labels
 - Push button
 - Toggle button
 - Text Area
 - Scroll Bar
 - Textfield
 - Menus
 - Menu bar
 - List
 - Layout widgets
 - etc.

Gadgets

- Windowless widgets, thus require lesser resources than widgets.
- Behavior is identical to that of the corresponding widget but the control of the gadget is the responsibility of the parent widget.

Resources

- Resources specific to the Main Window and its sub-elements can be useful when configuring the default appearance of an application.
- When these resources are set in an *appdefaults* file, the specifications can also provide a framework for users to follow when they want to set their own configuration parameters.
- The first step in specifying resources in an *appdefaults* file is to determine exactly which aspects of the program you want to be configurable.

Resources(cont.)

- Here is how to specify some resources:

```
Arg args[2];
```

```
int n = 0;
```

```
XtSetArg(args[n],
```

```
    XmNlabelString, label); n++;
```

```
label = XmCreateLabel (toplevel, "label",  
    args, n);
```

```
XtManageChild (label);
```

PushButton Widget

- Requires the `<Xm/PushB.h>` header file

```
Widget pushb_w = XtVaCreateWidget  
    ("name", xmPushButtonWidgetClass,  
    parent, resource-value-list,  
    NULL);
```

```
Widget pushb_w = XmCreatePushButton  
    (parent, "name", resource-value-  
    array, resource-value-count);
```

PushButton Gadget

- Requires the `<Xm/PushBG.h>` header file

```
Widget pushb_g = XtVaCreateWidget  
    ("name", xmPushButtonGadgetClass,  
    parent, resource-value-list,  
    NULL);
```

```
Widget pushb_g =  
    XmCreatePushButtonGadget (parent,  
    "name", resource-value-array,  
    resource-value-count);
```

Label Widget

- `#include <Xm/Label.h>`

```
Widget label;
```

```
Arg args[...];
```

```
int n= 0;
```

```
XmString str = XmStringCreateLocalized  
    ("A Label");
```

```
XtSetArg (args[n], XmNlabelString, str);  
    n++;
```

```
label = XmCreateLabel (parent, "label",  
    args, n);
```

```
XmStringFree (str);
```

TextField Widget

- `#include <Xm/TextF.h>`

```
Widget textfield_w =  
    XtVaCreateWidget ("name",  
        xmTextFieldWidgetClass, parent,  
        resource-value-list, NULL);
```

```
Widget textfield_w =  
    XmCreateTextField (parent, "name",  
        resource-value-array, resource-  
        value-count);
```

RowColumn Widget

- `#include <Xm/RowColumn.h>`

```
Widget rowcol;
```

```
rowcol = XmCreateRowColumn  
    (toplevel, "rowcolumn", NULL, 0);
```

Form Layout Widget

- `#include <Xm/Form.h>.`

```
Widget form = XtVaCreateWidget  
("name", xmFormWidgetClass,  
parent, resourcevalue-list,  
NULL);
```

Event Handling

- The essence of X Programming is the handling of asynchronous events.
- Translation tables define how widgets respond to particular events; contain the pre-defined callback resources.

Event Handling: Translation Tables

<Btn1Down>:	Arm()
<Btn1Down>, <Btn1Up>:	Activate() Disarm()
<Key> osfSelect:	ArmAndActivate()
<Key> osfActivate;	PrimitiveParentActivate()
<EnterWindow>:	Enter()
<LeaveWindow>:	Leave()

Translation Table with Virtual Bindings

BSelect Press:	Arm()
BSelect Click:	Activate()Disarm()
KSelect:	ArmAndActivate()
KHelp:	Help();

Callback Resources

- Callback resources are the hooks on which an application can hang its function; a widget that expects to call application functions defines one or more of these.

XmNarmCallback()

arm()

XmNactivateCallback()

activate()

XmNdisarmCallback()

disarm()

Callback Resources (cont)

- Adding callbacks:

```
XtAddCallback (Widget w, callback  
resource, pointer to the function,  
client data to be passed);
```

- Callback functions:

```
void methodname (Widget w, XtPointer  
client_data, XmWidgetCallbackStruct  
*cbs);
```

Event Handling (con't)

- Displaying the widgets:
 - **XtRealizeWidget** (*Widget shell*);
- Event loops:
 - **XtAppMainLoop** (*app*);
 - Invokes the event handling.