Xwin--A Graphical User Interface Developing Toolkit on Workstations

Ge Lei, Jijiu Zhao
Institute of High Energy Physics, Chinese Academy of Sciences
Beijing 100039, P. R. China

Abstract

Graphical User Interface is an essential part of accelerator control applications. A software toolkit called Xwin has been developed by the authors of this paper for easily constructing GUI applications of accelerator control systems. This paper mainly introduces the functions and features of Xwin.

1. Introduction

X Window System and Motif are the most popular GUI toolkits and de facto industry standards on workstations. Most control applications of accelerators in the world were developed based on X Window System and Motif.

X Window System is powerful but complicated. Motif has many pre-defined widgets for constructing GUIs. However, it lacks functions specially used for control systems [1].

Since the research of experimental physics becomes more and more dependent on computer aids, physicists have to develop programs themselves to help their research. For them it is rather difficult to learn and use the functions of X Window System and Motif, especially with so many parameters and attributes to be understood.

We developed a software package called Xwin to easily create GUI applications especially for accelerator control systems. Based on X Window System and OSF/Motif (Fig. 1), Xwin is a sharable library providing a set of functions to construct graphical user interfaces.

Xwin on the whole provides a simple way toward GUI programming. And many functions in it are powerful and convenient to use even for the experienced programmers of X Window System and Motif.

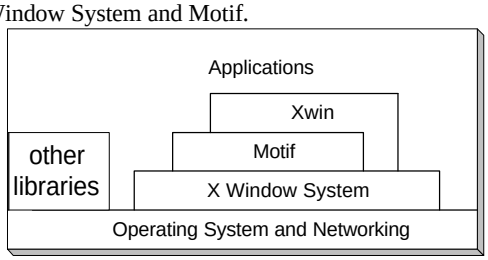

Fig. 1. Relation between Xwin, X Window System and Motif

2. The design of Xwin

First of all we designed a subset of GUI components. Some of the widgets are just adopted from Motif, some are inherited from Motif by adding special attributes, and some are integrating several widgets of Motif to form a new feature.

The widgets which can be created in Xwin are divided into three levels, as shown in figure 2.

- Shells which envelope other widgets and communicate with the GUI server. There are two kinds of shells – the application shell that is independent from other shells, and the transient dialog shell that is adhered to the top of its parent.
- Containers which manage the components in them. Xwin can create two kinds of containers, to treat the geometrical parameters of its children as relative values and absolute values separately.
- Simple and complex components, they are button, label, separator, frame, menu bar, pulldown menu, popup menu, canvas, axis plane, text area, list, knob, etc.

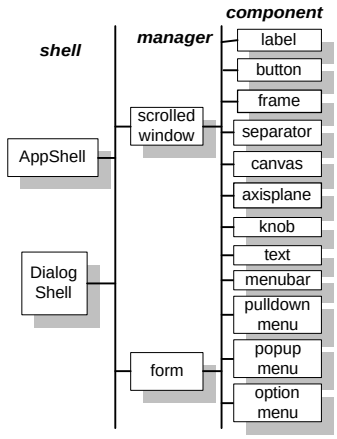

Fig. 2. Widgets and their levels in Xwin

Then we added what the physicists and control application programmers asked for - displaying scientific data in axis planes or in graphics, providing predefined elements to be easily used to control physical processes, etc. We ourselves faced the demand to improve the readability of the applications, to help with maintenance or development. We decided to construct a library to solve these problems.

3. Functions of Xwin

Xwin provides all together 109 public functions and several macros.

3.1 The structure of Xwin programs

- WnInit(char *title, int argc, char **argv, XtAppContext *context) - to initialize the GUI environment for the application. After calling this function, the programmer can create windows, buttons, menus, etc, to construct interfaces as needed and to set event processing procedures. Then, using
- WnRealize(top_widget) - to generate the widgets and to visualize them on the screen. It is a macro.
- WnMainLoop(context) - to start the loop for waiting and handling events. It is a macro.

This is the main sequence of Xwin programming.

3.2 Creating widgets

For example, WnDialog is to create a dialog shell. WnScroll is to create a scrolled window with the scroll bars appearing automatically when needed. WnPulldownMenu is to create a pull-down menu and the buttons on the menu, to set accelerator keys for the buttons and to add callback procedures. WnLabelP is to read a bitmap file and create a label for that bitmap picture. WnKnob is to create a knob for physical control.

3.3 Getting and setting attributes of widgets

For all widgets in Xwin, there are functions to set or get the height, width, location, foreground and background colors. Besides, for some special widgets, Xwin has special functions to set and get attributes. For example, there are functions to return or set the displayed string of a label. There are functions to read strings from a file to a text-area or write strings of a text-area to a file, etc. The current value of a knob can be set and got. Programmers can also set a widget to blink or stop the blinking.

The width and height of the screen can be visited, too. There are functions to add activated-callback procedures for buttons and destroyed-callback procedures for shells.

3.4 Man-machine interacting functions

We integrated some interacting procedures into single functions. For example,

*char *WnFileSelect(char *prompt).*

This function pops up a file selection box, prompts the user to view and select a file. Then waits for the user's reaction. When the “OK” or “CANCEL” button is pressed, the function is ended and the selected file name or NULL is returned to the program. The programmer can specify the prompting string through the parameter "prompt". In figure 3, the right part is a file selection box.

The functions WnGetStr, WnGetInt and WnGetFloat allow the user to input a string, an integer or a floating number respectively. WnConfirm prompts the user to confirm requested operations.

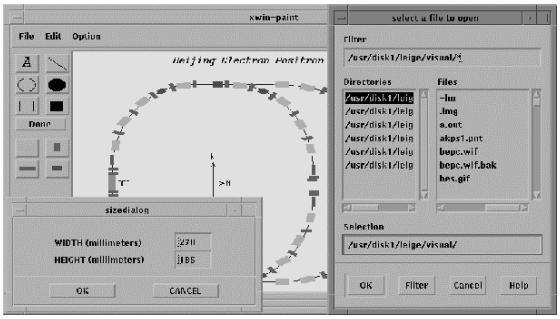

Fig. 3. File-selection Box

3.5 Drawing graphics

Xwin provides functions to draw and fill graphical elements such as a point, a line, an arc, an ellipse and a rectangle, a curve, etc.

3.6 Creating axis plane and displaying data

Also Xwin provides functions to create axes to form a coordinate plane and drawing mathematics functions and other graphical elements in the coordinate plane.

The upper left part of figure 4 is an example; only three functions are called to draw the data in the axis plane:

axis = WnAxisPlane(Widget parent, WnAxisData axisdata);

WnReadFile(char *file-name, char *curve-data);

WnAxisCurves(Widget axis, char *curve-data);

The above three functions are used to create an axis plane, to read data from a file, and to draw curves in the axis plane respectively. WnAxis Data is a data structure provided by Xwin to define an axis plane.

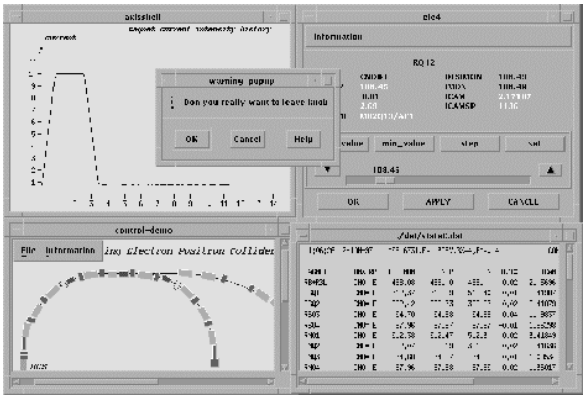

Fig. 4. View of a control panel

3.7 Common facilitating functions

Xwin integrates some commonly used procedures to facilitate C/C++ programming, such as converting data, visiting files, displaying pictures and photographs.

4. The features and techniques

Xwin is simple and practical. It provides an easy way to create window, to draw graphics and to generate complete man-machine interacting.

4.1 Simplifying drawing

X Window System and Motif are the basic environment of Xwin, which includes three libraries altogether: X Lib, Xt and Xm. Among the three libraries only X Lib has the functions of drawing graphical elements. However, X Lib is the lowest library of the three, and is also the most complicated and tedious one to use. Programmers have to query and set up many environment variables, and have to use several data structures that are necessare in X Lib drawing [2].

We designed a private data structure named __WnDrawareaRecord, to record for every canvas some key data, such as the graphical context and the pointer to the screen buffer. This record is linked to an extendable attribute of the drawing area widget of Xt Intrinsics. The necessary and tedious drawing procedures of X Lib are done inside every drawing function of Xwin. By this way, the complicated drawing processes using X Window System are transparent to the programmers using Xwin.

And the parameters of drawing functions are simpler than those of X Lib, using only the target canvas and the necessary geometric describing parameters.

For example, using Xwin to draw a line simply needs *WnLine(Widget w, int x1, int y1, int x2, int y2).*

The parameter w is the canvas. (x1, y1) and (x2, y2) are the coordinates of the starting and the terminal point of the line respectively.

4.2 Handling expose event

Redrawing all the contents of a drawing area when the expose event occurs is a fact that has to be faced and must be handled by all programmers of X Window System and Motif.

There are two basic methods to handle the expose event. One is to record what has been drawn and to redraw them when the expose event occurs. The other way is to provide a buffer and make the copy of the drawing area. When the expose event occurs, just copy the content of the buffer to the screen. Through our experience we found that the first method is slower than the second one, especially when many graphical elements are to be drawn. So we adopted the second method. The pointer to the buffer of the drawing area widget is kept in a record data of the type __WnDrawareaRecord. Private procedures are set to handle expose event. Therefore the programmers using Xwin need not know about expose event at all.

4.3 Integrating interactive functions

Some frequently used man-machine interacting procedures were integrated into single functions in Xwin, in order to not only simplify the GUI programming, but also to improve the readability of programs.

After analyzing the secret and principles of the Xt Intrinsics waiting event and dispatching event [3], we decided to create the following functions to conveniently complete the interactive processes: WnConfirm, WnFileSelect, WnGetFloat, WnGetInt, WnGetString.

Besides, we have set some protocol atoms for the dialog shells and added activated callback procedures for the buttons in the dialog windows. These two methods are to make sure that every interactive function can end itself and return an effective result, to keep the whole program running correctly no matter whether the dialog window is exited normally or destroyed abnormally.

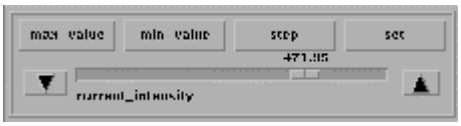

Fig 5. The outlook of Knob

4.4 Designing GUI elements for control

In accelerator control systems, some physical variables such as voltage and current values are adjusted frequently [4] [5]. We designed an integrated element called Knob, to help to handle the process. The outlook of Knob is shown in figure 5. A knob is the integration of a frame, a form, four push buttons (to set maximum, minimum, step and exact values), two arrow buttons and a scale.

We also provide functions for creating a Knob, for setting and getting the current value which is presented and controlled by the knob. A double precision floating number is used to present the variable controlle by the knob.

4.5 Displaying pixmap

In order to generate pixmaps, the object-oriented technique is used in the function *WnGIF(char *file-name)* to process the graphical files in GIF format. A pixmap can be used on the surface of a label or a button, to make the applications lively.

4.6 Error checking

At the beginning of every function of Xwin the effective of the received parameters is examined. Besides, in Xwin, before the functions of X Window System and Motif are called, the check on the effective of the key parameters is made, too. These ensure as much as possible that the programmers of Xwin get the information they can understand if an error occurs, other than leaving the problem to X11, Xt and Xm libraries.

5. Conclusion

To be simple and practical is the principle of Xwin. It provides a whole concept to construct GUI applications, simplifies graphical drawing and scientific data displaying, integrates some commonly used man-machine interacting procedures into single function calls, in a sense to improve the readability of programs. It realized integrated elements that are used very frequently in control applications, provides functions to connect text-area and file reading and writing. It can be used in C, C++ programming languages, and can be called combined with the functions of X Lib, Xt Intrinsics and Motif.

Xwin was first realized on Alpha workstation running Digital UNIX. Then it was transplanted to DEC workstation running VMS operating system. We have developed several applications using Xwin. Figure3 is the interface of a drawing facility and figure 4 demonstrates the BEPC control panel.

6. Acknowledgment

We would like to thank the colleagues of Controls Division of IHEP.

7. Reference

[1] OSF/Motif Programmer's Guide, Digital Press
[2] X Window System, Digital Press
[3] X Window System Toolkit, Digital Press
[4] J.J. Zhao et al, New Man-Machine Interface at the BEPC, ICALEPCS'95
[5] J.J. Zhao et al, Status of the BEPC Control System, ICALEPCS'93