Visual Motif Code Generator
by Xiao Pu

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
VMCG is the Visual Motif Code Generator, which is used to generate OSF/Motif C source code visually. Here VMCG uses a subset of OSF/Motif widgets, including representative Shell, Manager and Primitive widgets. VMCG also supplies widget hierarchy display, widget geometry management over its parent manager widget, widget resource editor and widget callback editor. Future directions for VMCG are discussed.

VISUAL MOTIF CODE GENERATOR

by

Xiao Pu

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

May, 1995

APPROVAL

of a thesis submitted by

Xiao Pu

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

4/20/95 — Date

J Dunbigh Stanley — Chairperson, Graduate Committee

Approved for the Major Department

4/20/95 — Date

J Dunbigh Stanley — Head, Major Department

Approved for the College of Graduate Studies

5/3/95 — Date

R L Bowen — Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature Xiao Pu

Date 4/20/95

TABLE OF CONTENTS

TABLE OF CONTENTS - Continued

List of Figures

ABSTRACT

VMCG is the Visual Motif Code Generator, which is used to generate OSF/Motif C source code visually. Here VMCG uses a subset of OSF/Motif widgets, including representative Shell, Manager and Primitive widgets. VMCG also supplies widget hierarchy display, widget geometry management over its parent manager widget, widget resource editor and widget callback editor. Future directions for VMCG are discussed.

INTRODUCTION

X/Motif

The X Window System (or simply X) is a hardware- and operating system-independent windowing system. It was developed jointly by MIT and Digital Equipment Corporation, and has been adopted by the computer industry as a standard for graphics applications.

X provides a root window within which you can display smaller windows. You can run a number of applications simultaneously and each application may have any number of windows.

Figure 1 shows a workstation screen with a typical assortment of windows. The root window is the large window that contains all other windows.

X is composed of, among other things, a set of library functions. These functions are normally referred to collectively as Xlib. Xlib is the heart of X and it consists of a large number of C library functions. Unfortunately, programming using only Xlib can be tedious, so the X designers created a second set of functions that are collectively referred to as the Xt Intrinsics or the X Toolkit. The Xt Intrinsics provide a higher-level set of functions that make programming easier.

Although easier to use than Xlib, the Xt Intrinsics can also be tedious and cumbersome to use, so widgets and gadgets were designed to utilize both Xlib and Xt Intrinsics functions and relieve the programmer of much of the dirty work. A widget is a user interface component composed of data structures and procedures, which usually has its own window. A gadget is a widget that does not have a window.

Figure 2 shows the relationship among OSF/Motif, Xt Intrinsics, Xlib and operating system.

The Motif widgets and gadgets were designed under the principles of object-oriented programming. This means that they are organized in sets of different classes. A class is a set of objects with similar characteristics. One of the most important aspects of object-oriented programming is that of inheritance. Any class can inherit characteristics from higher class. For example, the PushButton widget can inherit characteristics of the Label widget, the Primitive widget, and the Core widget. Primitive and Core widgets are considered superclass widgets.

A gadget does not have a window of its own, but must display its contents in its parent's window. Gadgets are, therefore, more efficient.

Motif's object-oriented architecture groups widgets and gadgets into different classes. Each class has data structures and procedures that operate on the data. The classes are structured in a hierarchy, with the superclasses toward the top of the hierarchy.

Figure 3 shows the Motif and Xt widgets Class Hierarchy. Subclasses, to the right, inherit features from their superclasses, to the left.

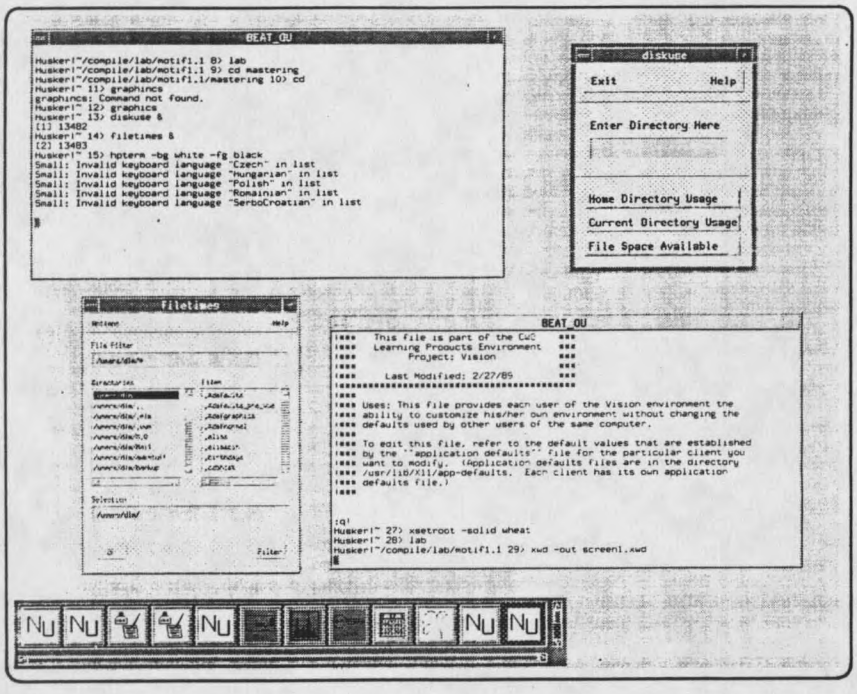

Figure 1: A Typical X Windows System Screen

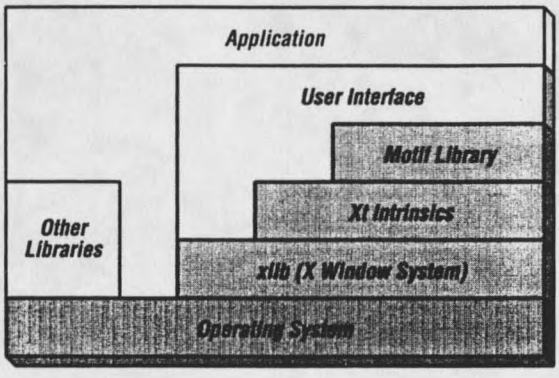

Figure 2: X Windows System Hierarchy

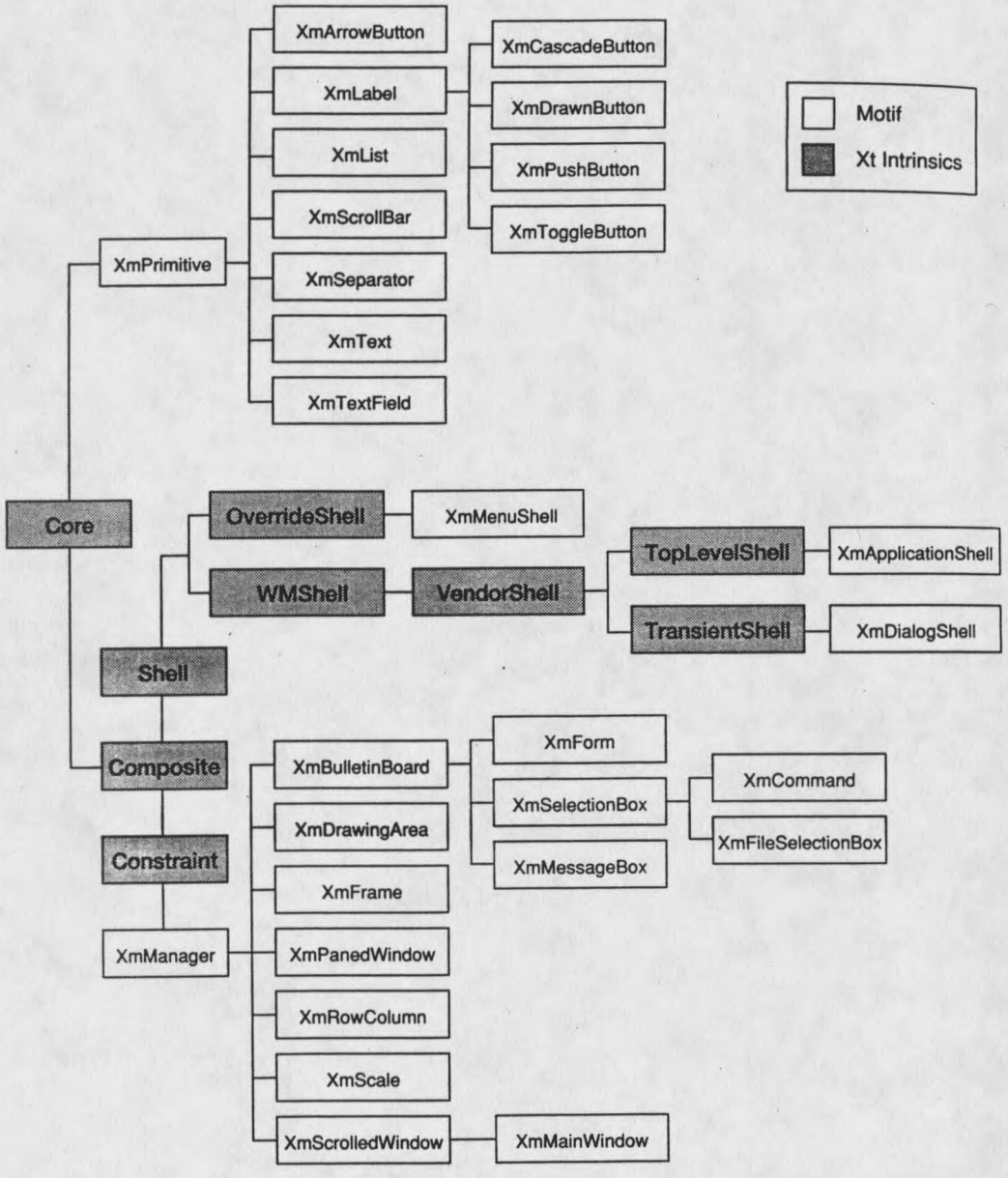

Figure 3: Motif Class Hierarchy

VISUAL MOTIF CODE GENERATOR

Developers have long known that the sophistication and power of their X/Motif GUI designs are restricted by the complexity of X/Motif unless they make available the many hours that it takes to create widgets. Here, we develop a Visual Motif Code Generator (VMCG) to give an easy and intuitive way to design X/Motif GUIs.

When we develop our X/Motif application, we need to know the hierarchy of the widgets used in our development intuitively. The best way is to supply a tree hierarchy for it; the second important thing is the geometry management for the child widget over its parent widget related to the issue of position and size; the third thing is to edit resource for the widgets; the fourth is to give C code for developers to enhance it.

In VMCG, we give a solution for the above issue by using Application Shell widgets, Form widgets, PushButton widgets, Label widgets and DrawingArea widgets, which represent the Shell, Manager (also known as the Container) and Primitive widgets in X/Motif widgets.

The Label Widget provides a visual label as text (in any font or using multiple fonts) or as an image (in the form of a Pixmap). Its text is a compound string and can be oriented from left-to-right or right-to-left. Compound strings can also be multilined, multifont or any combination of these. Basically, the Label widget is only intended to be used to display labels or other visual aids. Figure 4 shows a Label widget with three different fonts and lines in a compound string.

The PushButton widget supports the same visual display capabilities as Labels, since it is subclassed from them. However, PushButton also provides resources for the programmer to install callback routines that will be called when the user "activates" the button (clicks on it). Additionally, the PushButton displays a shadow border that changes in appearance to indicate when the pointer is in the widget, and when it has been activated. When

the PushButton is not selected, it appears to project out towards the user. The button's border is highlighted when the pointer moves into the button; in this state, the button is said to be "armed". When the user actually selects the button by pressing the pointer on the armed button, it appears to be "pushed in". The user actually "activates" a PushButton by releasing the mouse button while the button is in a "selected" state. Figure 5 shows examples of PushButton widgets.

The Form widget is one of the Manager widgets which are used to handle the placement of multiple widgets in a single window. Form supports the capabilities of the class by introducing a sophisticated geometry management policy that involves both absolute and relative positioning and sizing of its children. For example, Forms may lay out their children in a grid-like manner, anchoring edges of each child to specific positions on the grid, or they may attach them to one another in a chain-like fashion. Figure 5 also shows a form widget with three PushButton widgets on it.

The DrawingArea widget is a widget that behaves like a drawing surface for X. Once you have established a DrawingArea, you can use any of the X Drawing functions to draw it. The X Drawing functions allow you to draw points, lines, rectangles, arcs, and so on. As with any other widgets, you can attach a DrawingArea widget to a form, make it sensitive and insensitive, resize it, and so on. Figure 6 shows a DrawingArea widget with a mouse signature on it.

The ApplicationShell widget is used to create an entirely new Shell from within an application. That is, you might want to call some code and have it create an entirely new and separate Shell into which you can add widgets. This new Shell has all of the attributes of the toplevel Shell and fullfills the same function. It can contain anything you would normally put in a toplevel Shell: any widget, form, and so on. The new Shell is a complete window, and you can resize it, minimize it, and maximize it just like any other window.

In Figure 8 the Tree Hierarchy Interface is a ApplicationShell from VMCG MainMenu's toplevel Shell.

Using VMCG, we can develop our GUI designs visually and intuitively. We just select the widget type we need and add it to the application hierarchy, also we can delete some unneeded widgets from the hierarchy, design their geometry management by mouse movement, edit their resources, add some callback routines, and finally we can get the needed C source code. For example, Figure 7 shows an MSU student ID form GUI developed by VMCG.

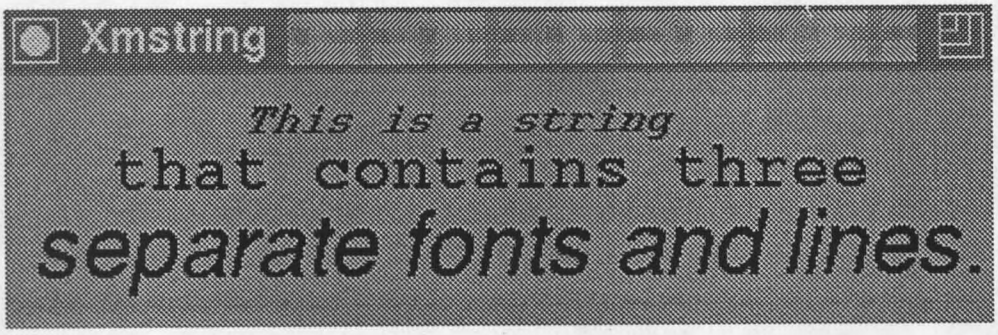

Figure 4: A Label Widget

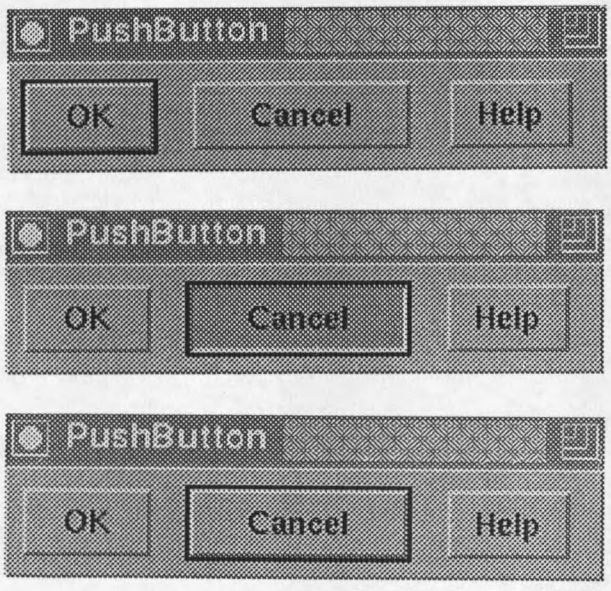

Figure 5: Examples of PushButton Widgets