IBM

Interdivisional Technical Liaison Conference on Image Processing and Visualization

Milan, September 15th - 17th, 1992

Conference Organizers: B. Collins and A. Stein

Local Organizers: P. Corsi, A. Bondi and L. Cicero

MigWob: a coherent C++ interface to mixed X-Windows and graPHIGS programming

G.Chiozzi - G.Ghezzi - M.Tucci

IBM SEMEA s.p.a.
Centro Ricerche e Soluzioni Tecnico-Scientifiche
Segrate - Roma

S. Pantarotto

Universita' degli Studi
Dipartimento di Scienze dell'Informazione
Milano

September 1992

Abstract

MigWob is a uniform and coherent programming environment for developers who need to get the best from both X-Windows and graPHIGS(*) under the AIX(*) operating system.

X-Windows power stands in its toolkits, considered as tools to develop Graphical User Interfaces (GUI hereafter) with a standard look and feel, but it lacks in pure graphical functions. On the other hand, graPHIGS is a powerful language for 2 or 3 dimensional graphic programming, but it does not make high level primitives for the user interface (buttons, menus...) readily available.

X-Windows and graPHIGS are difficult to be mastered and are too different from each other to be integrated in a single application. MigWob integrates X-Windows and graPHIGS in a unique framework, via a C++ hierarchy of classes.

The main benefits gained by the use of MigWob are the following:

- a cut off in the learning curve, thanks to the formal neatness of the library;
- a cut off in the development time;
- a significant reduction of the code size due to the reduced number of instructions;
- the resulting code is easy to read and to maintain;
- the code is also easily extensible and reusable, thanks to the characteristics of the Object Oriented Language used.

Currently, a working prototype of the library is available; it covers all the needs of 2D graphic programming (the area of our interest). It is a tool in use by groups of developers in Milan and Rome. Extensions to MigWob will be driven by the feedbacks coming from this on field usage.

At the same time, an Interactive User Interface Builder, called MakeWob, is under development. With this last tool, programmers will be able to "write" MigWob code using mainly the mouse.

Introduction

MigWob is an object-oriented library for two dimensional graphics. It consists of two blocks: LibGm and LibWob. They are built over X-Windows and graPHIGS respectively.

MigWob was born within a wider project in the Italian Scientific and Technical Solution Center (STCS hereafter), where the image processing group works on automatic recognition of maps and technical drawings. These activities are performed in a Risc System/6000(*) environment, under the AIX operating system. During the interpretation processing of a map, some automatic steps need a final check, to confirm the results computed by the system. Therefore, user interfaces providing advanced graphic capabilities are to be built. Moreover, the only way to test and debug algorithms for the automatic interpretation process is to display their results on a screen.

A basic design issue of MigWob was to provide a framework both to build 2D graphical user interfaces and to serve as a development and debugging tool for people involved in the automatic recognition of maps.

On the one hand, we needed a tool oriented to free application developers of the burden of implementing the user interface. On the other, we wanted to free them of the burden of learning and using a library providing advanced graphics (e.g. graPHIGS, GL, etc.). MigWob offers a set of high-level tools to the programmer skilled in building GUIs: new objects can be derived in an efficient and powerful way. To the application developer who needs to realize automatic programs, MigWob offers a quick way to build a simple graphic interface, while it provides functions and methods for the output visualization. These functions and methods hide graPHIGS and X-Windows, so that a deep knowledge of these is not required.

Events

X-Windows is event driven. The application is embedded in the "main loop". The event queue is constantly fed by the events occurring in the X-Windows environment, both specific or stranger to the application. X-Windows manages the queue and dispatches the proper messages to the application. The application simply waits for input events. Then it answers through the callback mechanism[1]. In MigWob, all the events are driven in a single queue and then dispatched to the relative graphic object. There are no functional distinctions between X and graPHIGS events: they are treated in a uniform way.

MigWob can require a shift in programming philosophy to the programmers used to graPHIGS. In graPHIGS, most programmers design their applications constantly checking the input that the workstation receives (Request or Sample modes), in order to answer with the proper actions. MigWob requires the programmer to adhere to event-driven programming[2].

Objects

MigWob is written in C++. The possibilities of this language are exploited in two ways. Firstly, we take advantage of the possibility of overloading functions and setting defaults. In this way, we get rid of hundreds of parameters and take care of all the settings that the programmer does not want to specify. The application specifies only the things relevant to the program. All the others are dealt with by MigWob. Secondly, we heavily use an object-oriented approach. All the elements of LibWob and most of the elements of LibGm are objects.

We built the library so that the programmer not used to Object Oriented programming does not have to build new objects or define classes and methods: there is no need to learn C++, and it is possible to go on writing programs in C, or even in Fortran. It is though true that the programmer, in order to use MigWob, must gain some knowledge of object-oriented programming. It is necessary to understand the change from a function-based language to a data-based one. A little syntax to invoke objects' methods must be learnt. Of course, the programmer must get used to overloading.

Programming GUIs

Nowadays, it is widely recognized that the application and the GUI should be sharply separated. Not only the two tasks should interfere as little as possible, but they should be accomplished by different people[1]. Unfortunately, this is often wishful thinking. Most of the time it simply happens that there are not enough human resources to separate the tasks, and the application developer must take care of the GUI too. MigWob provides a simple tool to achieve this task with the least effort. It is worth reminding that in graPHIGS it is impossible to distinguish between the application and the graphic interface.

What MigWob is and how it works

We try to give here a very general outline of the architecture of MigWob. As we mentioned above, MigWob consists of two blocks: LibWob and LibGm. The logical trait-d'-union between the two is given by the object "gPWorkstation" described below. Layers of higher-level functions can be built above MigWob, using the tools it offers. We built one for our purposes, i.e. a map editor.

LibWob

The main feature of X-Windows is given by the XToolkits. They are excellent tools to build graphical user interfaces, and nowadays we are all used to see interfaces with a standard look and feel: we accept rules on where certain buttons must be put; we expect the menubar to show popup menus when we click on one of its items; we want message boxes to appear in order to warn or advice us. XToolkit and Motif widgets perform this job in an excellent way. It is true, though, that X-Windows has its drawbacks: firstly, it is too rich. There are hundreds of functions and parameters to learn; and it requires an event-driven approach which could be unfamiliar to the application developers. It is unwise to require application developers to learn X-Windows in order to use the few toolkits they need to build an interface.

LibWob has been our solution. LibWob is a library of objects built over XToolkit and Motif.
The main interface objects are represented in the following trees:

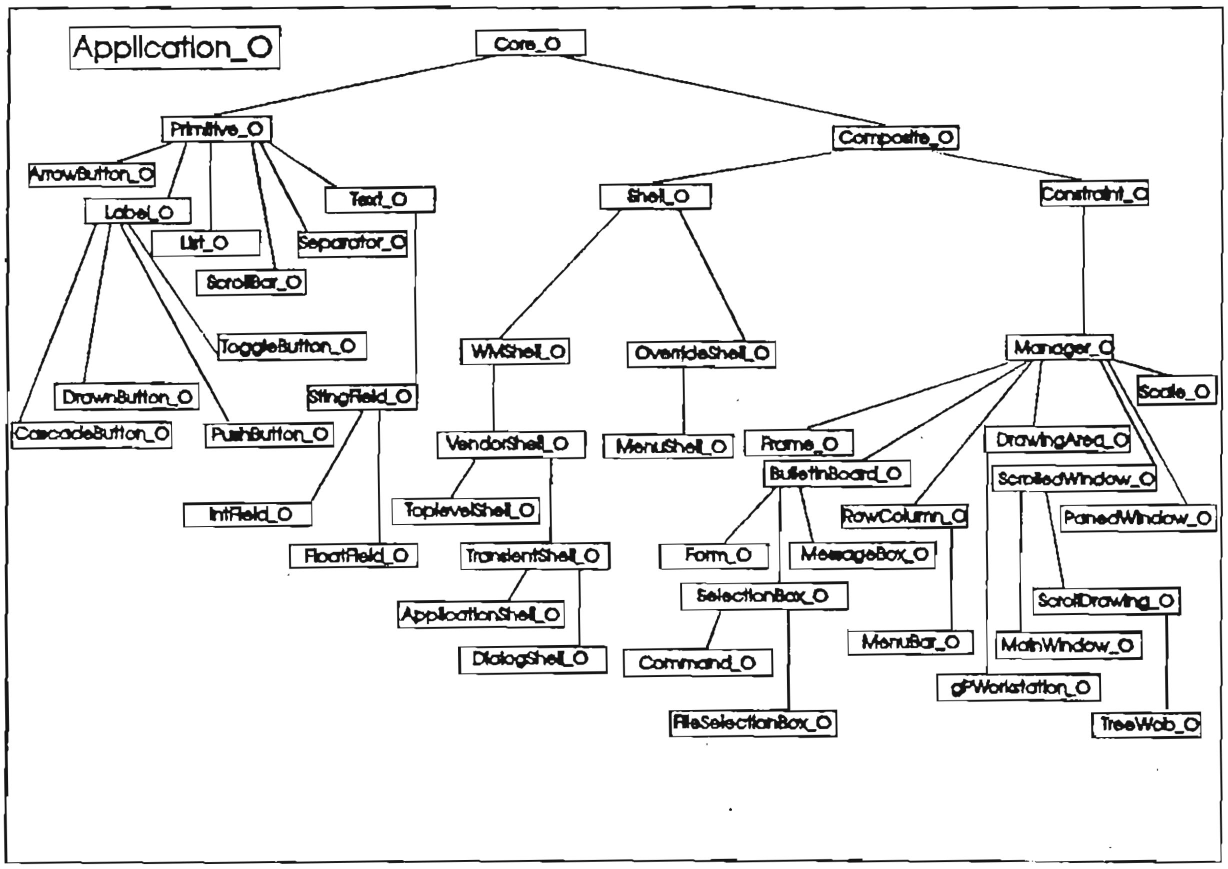

Figure 1. Main interface objects in LibWob

Every user interface built with this library is dealt through an instance of class Application_O. This object takes care of every initialization and of the display handling.
LibWob interface elements are a set of classes which correspond to the Motif widgets. Mirroring the Motif architecture, we built two main meta-classes of objects:

Primitive_O	The instances of classes derived from Primitive_O cannot include other objects: labels, buttons, scroll-bars are of Primitive type.	
Composite_O	Composite_O is the meta-class whose derived classes can contain one or more objects. Two more main classes are derived from Composite_O:	
	Shell_O	This class, and those derived from it, can have a single child object.
	Constraint_O	Again a meta-class, whose classes can include more than a single object. There are constraints between these objects.

Another important feature of the library is the error handler:

Error	Error handler. It intercepts MigWob errors and sends the appropriate messages. The messages are dealt with by the use of catalogs (the standard AIX message facility) which can be easily updated and modified by the user.

LibWob is easily expandible, thanks to its structure. If the need arises, the user can define a new class. A template file is supplied with the skeleton of every class and includes of the library. The new class can be easily fitted in the LibWob hierarchy, thus inheriting all the methods of the parent class. It is then sufficient to update the supplied makefile and compile the library. Building a new class is much easier than building a new widget. LibWob includes also higher level objects to ease frequent use operations: dialog and message boxes are typical of this set:

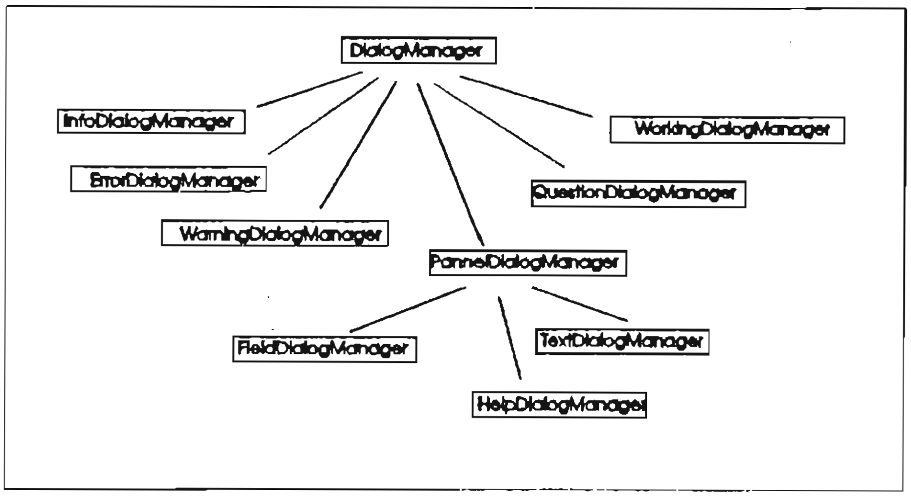

Figure 2. Simple dialogs

LibGm

GraPHIGS is a very powerful graphic language. It is very versatile and it virtually answers all the needs of a graphics programmer. It is a difficult language to learn, though, and it has a long start up time. The concepts and hierarchies related to the workstations, viewports, windows, views and structures are not readily mastered. The number of routines and parameters is very high.

Most applications actually use only a subset of graPHIGS routines and functionalities. We decided to choose the most suitable subset for our purposes (2 dimensional primitives and their attributes). Also, we took advantage of the fact that the graPHIGS architecture is suited for an object-oriented implementation. LibGm makes some choices, concerning for example the type of connection to the workstation, or which graPHIGS routine is best suited for a certain task, on behalf of the programmer. LibGm supplies the programmer with a set of primitives and methods easy to use. Our aim has been that of freeing the application developer from worrying about anything but the task the application must achieve. The "distracting" but necessary parts have been loaded on, and hidden in, the library. Besides, LibGm provides a set of high-level functions, which would require many graPHIGS instructions.

LibGm is built over graPHIGS. Its main objects are the following:

gPWorkstation_O	This object links LibWob and LibGm. It actually is an object of the Constraint_O class, but of course cannot contain other objects. Its methods create and connect a workstation, associate views to the workstation, and deal with change of coordinates.
Input	This is a meta-class whose derived classes define the input devices.
gPView	It creates the object view. The user manages the view by name. The methods associated with this object deal with the windowing, association of graphic structures to the view, panning, zooming in and out, etc.
Struct	It creates a graphic structure. Its methods include all the bidimensional graphic primitives and methods to edit a structure.

Most of the parameters concerning workstations, windows, and views are hidden to the programmer, as well as the creation of the workstation and of the structure store. In order to maintain a "soft" approach to the library for those programmers not familiar with object-oriented programming, all the graphic primitives, beside being methods of the Structure object, are duplicated and can be used as normal overloaded functions.

Other products.

There are other products on the market which accomplish some of the tasks achieved by MigWob. Most of them simply put a graPHIGS workstation in a window. These do not actually realize an integration between X-Windows and graPHIGS.

Some libraries use an object-oriented approach. These too, though, are based EITHER on X-Windows[3], OR on graPHIGS[4], and do not present an integrated environment. Moreover, many of the Object Oriented libraries built over X-Windows OR on graPHIGS[4], are actually based on Xlib only[4]. XLib does not include widgets. This means that the experienced Motif programmer is set back by the need to approach the new library in a very different way. Besides, the library developers must accomplish again many of the tasks which Motif already accomplishes. This was beyond our scope. Often, these Object Oriented libraries have a "look and feel" which is usually very different from the Motif one. But Motif "look and feel" and the use of widgets have become a de facto standard.

MigWob drawbacks

Both LibWob and LibGm cover a subset only of X-Windows and graPHIGS respectively. This obviously means that MigWob is less versatile of X-Windows or graPHIGS. Fewer things can be done. On the other hand, it is much easier to learn and to use. MigWob is a two dimensional graphic library. At the moment, no three-dimensional functionalities have been implemented. However, the library is easily extensible, and compatibility towards X-Windows and graPHIGS has been maintained. This means that, should the need arise, direct calls to graPHIGS and X-Windows can be used. We strongly discourage MigWob users from the use of this work around. Besides, the library is available under the AIX operating System only, while it could be ported under VM.

Learning MigWob

When we began to work on MigWob, we decided a "soft" approach to C+ + and Object Oriented programming, fearing that the change of philosophy would have been too bold for those colleagues used to third generation languages. So, in the beginning, we decided to exploit mainly the possibility of overloading, using, especially for the graphic routines, as few objects as possible. We were then both pleased and surprised when those same colleagues, when they began to test the library, asked us to implement more and more object-oriented functionalities.

MigWob is currently in use both in the Italian STSC and in other IBM SEMEA(*) departments. The results have been rather encouraging.

MigWob is easy to learn. It has been used by a wide variety of people: graPHIGS users, X-Windows users, programmers not acquainted with graphics. They all shifted quite easily to MigWob and appeared to enjoy the possibility of an integrated environment very much. They put the library under test, which helped us in the debugging phase, giving us useful suggestions at the same time. However, we have maintained the possibility of a "soft" approach to the library: the programmer who does not want to make an abrupt shift to object oriented programming can simply exploit the overloading of functions. For this purpose, all the methods associated with the Structure object in LibGm have been duplicated and can be used as plain functions. Besides, in projects involving the automatic recognition of maps, graphics is the most immediate debugging tool; MigWob is very handy for this practical purpose.

Bibliography

[1] Young D. - *The X-Window System: programming and applications with Xt*, 1990, Prentice Hall

[2] Foley J.D., vam Dam A., Feiner S.K., Hughes J.F. - *Computer Graphics, principles and practice*, 1990, Addison-Wesley

[3] Fekete J.D. - *WWL, a Widget Wrapper Library for C+ +*, 1990, Laboratoire de Recherce en Informatique, Faculté d'Orsay, Orsay Cedex (France)

[4] Wampler S. - *Development of a graPHIGS Based Object-Oriented Graphics System*, 1991, graPHIGS User's Group Meeting, Blacksbourg, Virginia (USA)

[5] Calder P.R., Linton M., Vlissides J. - *InterViews: A C+ + Graphical Interface Toolkit*, 1988, Computer System Laboratory, Stanford University

(*) graPHIGS, AIX, Risc System/6000 and IBM SEMEA are trademarks of the IBM Corporation.