NAS User Interfaces

Dan Wallach and Tom Woodrow

NAS Systems Development Branch
NAS Systems Division
NASA Ames Research Center
Mail Stop N258-5
Moffett Field, CA 94035

Abstract

Graphical User Interfaces at NAS have been developed over the past five years using the proprietary Silicon Graphics Inc. (SGI) Graphics Library (GL) as well as the locally developed Panel Library. The Panel Library was built to offer a higher level library to build user interfaces . Panel Library has been highly successful, but suffers from several problems. With the forthcoming upgrade of the Silicon Graphics Operating System IRIX 4.0 there are several new alternatives with respect to user interface development . This paper describes the alternatives, recommends a transition to an X/Motif development environment and defines an implementation plan.

1. Current Development Environment

Currently, the production SGI workstations at NAS are running the IRIX Operating System, version 3.3. In the next Operating System release, the Window System will transition from a Sun Microsystems NeWS-based implementation to an MIT X-based implementation. This should not cause significant problems for the majority of existing graphics applications. However, those applications which use specific and even undocumented features of the current window system will require software modifications. We have had development versions of the Operating System in-house for some time and have verified this.

Two of the major GL based tools developed at NAS are the Panel Library and Panel Editor. These were developed by David Tristram and Eric Raible as an alternative to writing user interfaces directly in GL. The Panel Library defines high level graphical objects which are composed of lower level GL primitives. These graphic objects can be used as components in a graphical user interface. At the time the Panel Library was developed, there were no other high level user interface li-

braries available on the SGI platform. Currently the Panel Library is used in a number of applications, both in-house (most notably FAST) and external. It is available through COSMIC. There are currently several problems with Panel Library and the Panel Editor:

1) We have lost most of our Panel Library development expertise (both the author and the major support developer have left NASA or moved on to new jobs within the organization).
2) There is an inadequate support organization (in fact, bugs are being fixed by users rather than NAS).
3) The Panel Editor makes extensive use of undocumented and no longer supported calls. It will require a major rewrite to function in IRIX 4.0

2. Desired Development Environment

We need to continue some level of support for the Panel Library, if only for those applications which depend on it. However, an alternative which provides equivalent capability and is vendor supported would be preferable for new development. Additionally, we should take advantage of any emerging standards. It is required that a graphical layout tool accompany any new user interface library. Although these tools are typically used only early in the user interface design process, they can greatly simplify user interface design. This is especially true for users who are unfamiliar with or frightened by the window system.

In summary, the desired User Interface development environment will include the following:

1) continued support for the Panel Library as is (primarily for existing applications)
2) a new user interface library which:
   a) includes at a minimum the set of graphical objects defined in the Panel Library
   b) works under IRIX 4.0 (i.e., is either GL or X based)
   c) offers external support (vendor or 3rd party)
3) a graphical layout tool which can be used to create interfaces composed of any of the graphical objects in the user interface library

2.1 X based Alternatives

Over the past three years the X Window System has emerged as a de facto windowing standard. There has been a large volume of code developed by many vendors and users. One of the strengths of the X Window System is the extensive range of options in user interface toolkits. Of these toolkits, there are 2 which appear to be more robust than the others and have gained the largest user base. These are Motif, from the Open Software Foundation, and XView (OpenLook), from Sun Microsystems.

With IRIX 4.0, NAS software developers will have access to these robust user interface toolkits. Most computer vendors support the X Window System and Motif. Many have additional toolkits available from third parties. Below is a chart of the current NAS production hardware vendors and support options available for X, Motif and OpenLook

Vendor	OS version	X	Motif	OpenLook
SGI	IRIX 4.0	SGI	SGI	SGI
Cray	UNICOS 6.1	CRI	CRI	3rd party
Amdahl	UTS 2.1[1]	Amdahl	Amdahl	none
Sun	SunOS 4.1	Sun	3rd party	Sun
Convex	ConvexOS 10.0	Convex	Convex	Convex

Almost all computer vendors now have a plan for X Window System support.

Graphical layout tools are very useful early in the development of the user interface. There are a wealth of choices available in the X world and we have evaluated several. These are UIM/X, from Visual Edge, Builder Xcessory, from Interactive Computing Solutions, and TAE Plus, from NASA Goddard. All run on SGI workstations under IRIX 3.3 and 4.0 and Sun workstations under SunOS 4.1. Their relative merits will be discussed below.

While it is important for the libraries to be available across all NAS systems,

---

1. Currently, the Amdahl is running UTS 1.2. Changeover to UTS 2.1 should occur with the next 3 - 6 months. Amdahl says it will begin shipping X and Motif with the operating system by 1Q92.

the layout tool need only execute on NAS workstations or the Support Processing Systems. This is because you can do your layout on the workstation and compile the code anywhere you have the libraries.

The X Window System is implemented in a client/server model. There are client processes which solve specific problems, like creating a virtual terminal window; and there is a generic server which handles all graphical output. The X server executes on a user workstation and controls the screen. The X client can execute anywhere on the network. Clients communicate with the X server using TCP/IP sockets. For this reason, the X Window System is called network transparent; i.e., the clients don't care whether the server is local or remote. Because of this, you can run X clients on a centralized machine and direct your graphics to any workstation you like. For brevity, we will ignore issues of security.

Because X graphics can be distributed to any X server over the network, running the layout tools on the Support Processing System (SPS) and directing the graphics to the workstations has the appearance of running directly on the workstation. Several existing applications on the SPS already do this (Mathematica, WingZ and FrameMaker). For these reasons, we probably only need to provide the layout tools on the SPS or (optionally) workstation fileservers.

2.2 Other GL based Alternatives

Although the X Window System offers many new alternatives for user interface development environments, there is at least one other recent alternative. This is the Forms Library and layout tool, designed by Dr. Mark Overmars of Utrecht University, in the Netherlands. Forms is funded by a grant from the Netherlands Organization for Scientific Research.

The Forms package is built on top of the GL library from SGI. It is available in the public domain.

Although Forms is available freely, it suffers from some of the same problems as the Panel Library:

1) There is no available support organization (Overmars currently is supporting the software himself).
2) The user base for this product is quite limited.
3) There is development required to replace the Panel Library specific actuators

(these will be discussed in more detail in the next section).

Due to these problems, the Forms alternative should not receive serious consideration at this time. Individual users may choose to use it on their own. Forms will be installed in the unsupported area on the Support Processing System.

3. Detailed Library Comparisons

The requirements for a NAS user interface library are based initially on Panel Library functionality. This is in order to assure a migration path for existing software.

The Panel library contains a large number of graphic objects, called actuators. Many of these do not appear in any other toolkits available in the X environment. The following table shows a comparison of Panel and Motif.

Feature	Panel	Motif	XView
Misc.:			
scripting	YES	no	no
3-D look	somewhat[1]	YES	YES
Dials	YES	no	no
Context sensitive help	no	no	YES
Clocks	no	no[2]	no[2]
Scrolling regions	YES	YES	YES
Pop-up requesters	no[3]	YES	YES
File selector	no	YES	YES
Meters	YES	YES[4]	YES[4]
Strip charts	YES	no[5]	no[5]
Iconified buttons[6]	YES	no	no
Color choosers (pallette)	YES	no	no
Text:			
Input: one line only	YES	YES	YES

Output: one line only	YES	YES	YES
Input: multi-line (editor)	no	YES	YES
Output: multi line (scrolling)	YES	YES	YES
Buttons:			
keyboard equiv. [7]	no	YES	YES
radio	YES	YES	YES
cycling	YES	YES	YES
arrows	YES	YES	YES
Scroll Bars:			
2-D positioners (puck)	YES	no	no
value-attached	YES	YES	YES
handles change their length	no	YES	YES
multi handles (multislider)	YES	no	no
differential[8]	YES	no	no
slideroids[8]	YES	no	no
"fine" control	YES	YES	YES
Menus:			
sub-menus	no	YES	YES

[1] Panel's 3-D look is supported only by some buttons, and is not consistent.
[2] The Athena widgets, an example widget set which comes with the standard MIT X distribution, support clocks (example: xclock), which could possibly co-exist with Motif widgets.
[3] Panel supports pop-ups insofar as you could open another panel, but the other systems actually allow a popup to temporarily disable all the other buttons.
[4] Motif could support digital meters. Panel supports a specific output-only analog meter. Forms could emulate such a meter with a Dial.
[5] The Athena widgets support strip charts (example: xload) which could possibly coexist with Motif widgets.
[6] Panel allows a number of buttons to simply disappear into an icon. Other packages easily allow a button to cause a pop-up to appear, duplicating the functionality.
[7] Motif allows using the arrow keys to traverse menus, as well as using the return

key to indicate a "default" action.
[8] Differential sliders move up or down with a velocity proportional to how far the mouse is from when the select button went down. Move the mouse farther, the slider moves faster. Slideroids are like differential sliders, but there is not a scroll bar displayed, just a number indicating the current value.

3.1 General Impressions

Below are some general impressions of the libraries:

Panel

Panel is relatively nice. It contains a large number of strange and different widgets which are not available anywhere else, like multi-sliders, pucks, and palettes. Because it was developed at NAS, there are no licensing problems when programs are sent to COSMIC.

We would probably keep Panel forever if it were not for the glaring problems which were stated in section 1, above. The most serious problems are lack of support and expected incompatibility with IRIX/GL 5.0.

Motif

OSF/Motif is available from a number of vendors, and supports a consistent look-and-feel across these platforms. To use it on the IRIS requires knowing the X toolkit (Xt). Motif will be around for a very long time as many companies have adopted it.

For this reason, Motif is a good system to write future programs in, except:

- •GL-style event handling is not supported (qdevice(), qread(), etc.).
- •IRIS programs will not run on pre-IRIX 4.0 operating systems.
- •Motif can add a thousand lines of code to a program, and as much as a megabyte of library linked to it with non-shared libraries.

Many commercial products improve the programming environment and make Motif something we can recommend to use. Motif, and most X toolkits, without a layout tool are very tedious during the layout phase.

3.2. User Subjective Feel

The most important thing about a user interface is how the users react. The buttons in Motif and Forms have adequate appearance, but the on-or-off state of Motif toggle-buttons is not readily apparent given default settings. Forms push-buttons and

light-buttons are easy to understand, as are all the Panel actuators except the typeins.

Of the three, Motif seems to run hands-down fastest. It is just smoother. Part of the blame lies in Forms or Panel using GL calls which really make X calls and translate them around. Much of this performance problem is supposed to improve, but we will see.

Panel

Panel is functional, but ugly. All panels are the same ugly blue color. Sometimes, a panel can take an awfully long time to appear on the screen. Slider response is satisfactory, though still not as crisp as Motif. Menus are not wonderful, but they work. Panel has tons of buttons and output devices that the others lack, although most of these could be implemented in the other widget systems without too much work.

Motif

Motif looks as good as a user's resources. You can reconfigure an X/Motif program by defining resource values in your own X resource database. This is nice because if you do not like the default look, you can fix it, without access to source code. Motif windows appear fairly quickly and sliders respond amazingly well. Programs feel faster.

Motif provides some fairly obscure buttons whose utility seems limited.

5. Graphical Layout Tools

The Panel Editor (pe)

For Panel, Eric Raible wrote 'pe'—the Panel Editor. pe lets you lay out your various panels, and will generate a LISP-like output for itself, along with normal C output for your program. The LISP output is very nice because it is human editable, especially with an editor like Emacs which is very good at Lisp mode. To sum it up:

Good features of the panel editor:

- human editable output (LISP syntax)
- context-sensitive help

Problems with the panel editor:

- •does not work under IRIX 4.0
- •uses an undocumented call of 3.3.x—Raible is not interested in fixing this.
- •slow on a PI, reasonable on a 320VGX
- •does not support many widgets that Panel supplies (multisliders, slideroids, view/graphframes, palettes)—this often means using pe to generate the original interface and then modifying the C code by hand. FAST does not even bother with pe for this reason.
- •very little documentation (but not really necessary)

UIM/X

UIM/X is a commercial product. It runs under IRIX 3.3.2 or 4.0 without problems. As UIM/X is a commercial product, it should also be available for Sun or other platforms. UIM/X has several interesting features pe and fdesign lack. Foremost, UIM/X contains a complete C interpreter. Currently, it does not take much effort to crash UIM/X with bad C code, but at least they go to great effort to save your work before the program dies. The C interpreter is not really meant as a general development environment (like Saber-C, for example). Rather, it allows button actions to be specified while they are laid out. If the user wants a button to cause a file requester to come up, the callback will popup the file requester, all within UIM/X. This is a very nice feature. In fdesign or pe, the user would only be able to specify the callback's name, and the user would not be able to test the interaction until later.

UIM/X's methodology for laying out widgets is slow and painful. You must select the desired widget from a nested menu each time.

UIM/X's native save format is similar to X resources—it is very human readable, although it is strongly recommended that you do not edit it. UIM/X's C output uses a proprietary library of calls, not normal Motif code. This brings up UIM/X's biggest flaw: UIM/X is a commercial product. We would buy it from Silicon Graphics on the same basis we buy other commercial software. With the standard license, we pay per workstation which could grow expensive. Also, with the default distribution, we receive a binary libuimx.a, but no source. In the next version of UIM/X, tentatively available in the next several months, straight Motif code will be output, eliminating the need for libuimx.a completely.

Good features:

- • Runs on IRIX 3.3.2 and 4.0.

- Supports everything in Motif.
- (theoretically possible to) add your own widgets.
- Built-in C interpreter.
- Extensive manual.

Problems:

- per workstation licensing costs
- requires at least 16 MB of RAM to run, and 32 MB would be better. The binary image alone is 4.5 Megs. It seems to run better remotely from wilbur, with its 128 Megs.
- needs more on-line help than it has—very weak right now
- still has strange bugs in places—will crash when you do something weird in your C code.
- some common interactions (like, changing the label name on a widget) should be easier to do than they are
- has no support (currently) for GL, GLX widgets, and other IRIS-specifics.

Builder Xcessory

Builder Xcessory is a commercial product from Interactive Computing Solutions. The product is in version 2.0 and uses Motif version 1.1. Below are some thoughts about the product:

Good Features:

- Very intuitive interface (significantly better than UIM/X).
- Very fast (lots faster than UIM/X).
- All resources are available as selections from a list (rather than entering the resource as a text string in UIM/X).
- Good on-line help.
- The generated C output code is human readable and usable. Five different files are generated: callbacks-c.c, creation-c.c, main-c.c, makefile-c, and app-defaults. A real user will only need to modify callbacks-c.c and can use the other c files to generate quick test applications.

Problems:

- Per workstation licensing costs (same as UIM/X).

TAE Plus

TAE Plus is a product developed at NASA Goddard. It is available to NASA centers at no cost from COSMIC. The tool is due to be released for SGI by early 1992. We evaluated version 5.1 for the SGI platform. It is not as strong a candidate as either of the commercial products mentioned but may be installed on the system because of its cost. The determination whether to install this on the NPSN will be made based on user feedback from several test users.

Good features:

- Source code included.
- It is free to any NASA user.
- Limited free support is provided by the developer.
- A significant amount of documentation is provided.

Problems:

- Generated code uses a higher level library on top of X/Motif called Wpt.
- Use of TAE is less intuitive than either UIM/X or Builder Xcessory.

6. Recommendation

Motif and XView will be available on the SGI platform, bundled with the 4.0 Operating System. We currently have it in-house for evaluation with IRIX 4.0. However, we will require source code to the Motif libraries to use as examples if we decide to make any custom graphical objects. XView source code is available via anonymous ftp from many sites. We recommend the purchase of a Motif source license for one Silicon Graphics machine. A Motif source license has a one time fee of $1,000.00.

We further recommend the purchase of Builder Xcessory for all 4 SPS machines. Builder Xcessory is priced on a per CPU basis at approximately $2500 per copy. Providing the layout tool on the SPS has the following benefits:

- Cost is limited to 4 copies (approx $10K)
- People at remote sites can use it as well as users located at Ames
- If we decide in the future to change to a new layout tool, we are not heavily invested with this one.

Based on user experience with Builder Xcessory, we may decide to buy more copies in the future or purchase another GUI tool.

7. Plan

1) Acquire Motif source license to aid in widget development.
   This is currently in progress. A Purchase Request was initiated in 10/91.

2) Develop Panel specific widgets in a Motif style to augment the Motif library (puck, multislider, strip chart, dial, meter).
   These are currently in progress. Development of puck, dial and meter is nearly complete. Multislider and strip chart should be completed by 4/92.

3) Acquire graphical layout tool.
   We plan to initiate a purchase for a graphical layout tool in 1/92.

8. Conclusion

With the transition to IRIX 4.0 some user applications will require modification due to the underlying window system change. We hope to capitalize on the changeover by providing a better user interface development environment. This will be based on the emerging standards of the X Window System and Motif. Existing applications which use the Panel Library will continue to function. We will offer an alternative for new development and a transition path for existing applications.

Appendix

Included are several hardcopy examples of Panel Library, Motif and XView. Each shows a specific layout tool and a single example program to give you a feel for the differences in look and feel between the three.