SLAC-PUB-5982
UH-511-757-92
November 1992
(E/I)

Report on the UNIX "Tupleviewer" Challenge [1]

Frederick A. Harris

University of Hawaii, Honolulu, Hawaii 96822

Anthony S. Johnson

Boston University, Boston, Massachusetts 02215

Jason Hollinger

California Institute of Technology, Pasadena, California 91125

Tom Glanzman, Paul F. Kunz, Tom Pavel, & Paul E. Rensing

Stanford Linear Accelerator Center, Stanford, California 94309

Dick Damian

IBM Palo Alto Scientific Center, Palo Alto, California 94304

One result of the 1991 B Factory Workshop at SLAC was the definition of a software benchmark project: a user-friendly data browsing tool called a "tupleviewer". The tupleviewer is a program which interactively displays 1-D and 2-D graphical plots from data stored as n-tuples. A set of rules define the minimum requirements for the tupleviewer, but the choices for platform, programming language, window system, graphics package and GUI development system are left open. The purpose of this project is to provide an arena in which to compare these open choices, as well as to provide a training exercise. The results of these efforts, including the experiences of the developers and comparisons between the projects are reported. In particular, comparisons are reported between the various GUI tool kits used.

Introduction

One of the working groups that was formed during a series of B Factory Workshops at SLAC in 1990[1] and 1991 was the Computing Subgroup. This group quickly came to the conclusion that in order to handle the large amount of data that will be produced at a B Factory, a farm of RISC workstations would be required. Since UNIX has been the standard operating system on these workstations, high energy physicists would have

[1]Work supported in part by the Department of Energy, contracts DE-AC03-76SF00515, DE-FG03-92-ER40701, and DE-AC03-83ER40103.

Presented at the Conference on Computing in High Energy Physics, Annecy, France, September 21-25, 1992.

to learn a new operating system and become familiar with new tools, which are quite different from those of their traditional computing environment. In the 1991 Workshop, Paul Kunz suggested that the best way to learn about the myriad of computing issues was to do R & D projects and proposed that to learn about graphics and Graphical User Interfaces, in particular, people might participate in the "tupleviewer" benchmark project.

The idea is for people to implement essentially the same program, then make comparisons of the ease of implementation. The program must be able to display 1D and 2D histograms of data collected as n-tuples, be able to request a "file browser" to pick the n-tuple to be read in, be able to select the variable to be plotted using a "tuple browser", and be able to point and click to change display options. To display a 2D plot, the user should be able to pick two variables in the same manner as for the 1D plot. Plots should be autoscaled so that all data is visible. Then the user should be able to change the low and high range that is displayed and the number of bins by using sliders or by typing in numbers. The displays must change interactively as the sliders are dragged. The HippoDraw program, written by Paul Kunz and colleagues on the NeXT, had already greatly surpassed the benchmark.

Graphical User Interfaces

The way in which people interact with computers is shifting from command line interfaces to point and click style graphical user interfaces (GUI), such as those found on Macintosh computers. An important advantage of these interfaces is a common "look and feel" across applications that makes them easier for the user. Unfortunately, it has been estimated that creating a good GUI interface can take as much as 75 % of the program development time[2]. Fortunately programming GUI's is becoming easier as libraries and tools are becoming better. GUI builders are becoming available which allow the visual aspects of the GUI to be designed with WYSIWYG layout tools.

NeXTSTEP on the NeXT was designed specifically for GUI programming, and it is possible to develop interfaces with ease. Unfortunately this system is not available on other UNIX workstations. The rest of the UNIX world has standardized on X windows. However the X specification extends only up to the windowing and drawing level. It does not include the top level, which gives the "look and feel" of the interface. The two main contenders for the top level are Motif and Open Look. In addition, there are public domain interfaces available, including InterViews from Stanford. Motif and Open Look include objects, called widgets, that are the components of a GUI. Typical widgets include push buttons, sliders, menus, as well as more specialized widgets such as file selectors. The interfaces also include style guides and a window manager. One of the main purposes of the tupleviewer project was to determine the relative ease-of-use of the various interface developers.

Projects and GUI Developers

A wide variety of systems and software were used in the Tupleviewer project, as indicated in Table 1. Note that many of the projects run on more than one system. By "Graphics", we are referring to the software that draws the histograms and scatterplots

Table 1: Tuple Viewer Summary.

Project	Systems	GUI builder	Window/ widgets	Lang.	Histogram/ graphics	Time	Size (bytes)
F. Harris	DEC5000 RS6000	VUIT	X/Motif	C/F77	hbook/hplot higz/Xlib	1 yr 20%	3.0 M
xtc T. Johnson	VAX,SGI RS6000	VUIT Midas	X/Motif	C	hippo	6 mo[a] 50 %	65 K
D. Damian	RS6000	AIC	X/Motif	C	hippo/phigs	3 wk 100 %	1 M
J. Hollinger	RS6000	AIC	X/Motif	C	hippo	1 mo 100%	630 K
P. Rensing	RS6000 SPARC	ibuild	X/ InterViews	C/C++	InterViews hippo/idraw	5 wk 25 %	2.0 M
hiv T. Glanzman	Sun SPARC	TAE+	X/Motif	C	hippo	1 mo 60 %	2.0 M
hip T. Pavel	Sun SPARC	none	X/ InterViews	C/C++	InterViews/ hippo	1 mo 50 %	1.3 M
HippoDraw P. Kunz	NeXT	Interface Builder	NeXTSTEP	C/ Obj-C	hippo/draw/ DPS		0.6 M

[a]This includes the time to develop the Midas interpreter.

in the drawing window. The *hippo* package, which manages ntuples and produces histograms and scatterplots using underlying graphics libraries, such as X graphics, PHIGS, or InterViews, was written at SLAC. *Idraw* is a full drawing program and allows adding labels, rotation, stretching, etc. to the plots being drawn. Also given in the table is the development time and estimated fraction of the time spent on this project. For many of the people involved, this project was their first GUI based programming exercise. Thus the time estimates are not necessarily comparable since the development time included the startup time of learning X windows, Motif, the GUI builder and sometimes even the C programming language. Also some projects went considerably beyond the benchmark guidelines by including multiple windows, cuts, postscript printing, etc., while others fell a little short of meeting the guidelines. However the numbers given are useful as an upper limit on the amount of time to develop a simple but useful GUI application. Also given are rough estimates of the program sizes.

Details of the GUI builders that were used are presented in Table 2. All allow WYSIWYG development of the GUI, greatly reducing the amount of time required. Some of the GUI developers are available on many systems. Even those that are not so widely available, however, generally output some standard form of code, such as C source code or UIL (User Interface Language), thus enabling one's application to be built on many systems.

Comparison

Although there were variations in the GUI design and ease of use, all of the projects produced a GUI tupleviewer program passing or nearly passing the benchmark guidelines,

Table 2: GUI Builder Summary

Product	Vendor	Supported Platforms	Code Generation	Proprietary Runtime	WYSIWYG
TAE+ 5.1	NASA/ Cosmic	Sun, DEC, Mac, SGI, HP, 386	C,Ada, TCL	yes (free)	yes
VUIT 2.0	DEC	Sun,VMS, ULTRIX	C, UIL	no	yes
AIC 1.0	IBM	IBM	C, UIL	no	yes
InterViews 3.0	Stanford Univ.	any with X11 & C++	C++	no	yes
Interface Builder 2.0	NeXT	NeXT	no	yes	yes

and all produced very pretty interfaces. The ability to interactively control the variables to be plotted and the type of plot to be plotted makes exploration of the ntuple data easy, as well as fun. It should be emphasized that that the majority of the people involved in this project had no previous experience with X windows, Motif, or the GUI builder that was used. Two of the GUI interfaces are shown in Figs. 1 and 2.

The use of the GUI WYSIWYG developers was found to be essential. Now only a small fraction of the time is needed to develop the visual aspects of the GUI. Time is still required to write the control program which must carry out the action specified by the user, for instance, what actually happens when a particular button is pushed. However not all authors were completely happy with the speed of the GUI developers used. One (Hollinger) found that it was faster to edit the .i output files of the IBM AIC, than to do everything with the AIC developer itself. The other (Johnson) found that VUIT was a very convenient and powerful method of building the initial configuration of the the GUI interface but that the compiling/linking/running cycle needed to test the many C routines needed in the application led to a very slow and painful development. To overcome this, he developed a "homegrown" Motif interface interpreter, Midas (Motif Interactive Data Analysis Shell), which aims to provide a rapid interface development tool, using an extensible interpretive language.

Although the GUI builders help in the development process, there are still problems:

1. Widget libraries are built on top of a multi-layered X toolkit system. Developers must be familiar with many routines at each level, a total of over 1000 routines.
2. The OSF/Motif documentation[3] is voluminous but not necessarily understandable. There are few examples, and much trial and error development is necessary. Although the TAE+ documentation is also voluminous, it offers some examples.
3. Even with the great complexity, the Motif widget set is incomplete. For instance, there is no list widget which includes an *ok* and a *cancel* button to allow the user to signal completion of a multiple selection. However, Motif is extensible. Also missing are simple output widgets, like movers, rotators, and strip-chart recorders, which are necessary for control-panel applications. TAE+ does provide such widgets.
4. Some Motif widgets are buggy. The program crashes when doing things that are

correct according to the documentation. By using alternative approaches, one can avoid the crash.

5. The Motif GUI developer documentation assumes knowledge of the Motif widget set. This is not very helpful for the beginner.

Even with better documentation and with the bugs removed, a library of working examples is needed to help the new GUI developer. One of the benefits of the tupleviewer project is that it provides a collection of working examples for such a library.

So far, we have primarily discussed the C development systems. Our experiences seem to indicate that Object-Oriented (C++ and objective C) toolkits represent a big improvement in ease of coding GUI's. For example, the benchmark required that histogram limits and the number of bins must be controlled by both slider widgets and text widgets. In the straight C implementations, if either the slider or text widget changed, then the callback routines had to make XtSetValue calls to set the other widget to the correct value. This required many lines of code and the usual debugging. On the NeXT, the information concerning the change could be transmitted between the slider and the text widget by drawing a line between them in the Interface Builder.

Unfortunately, the availability of object-oriented toolkits is limited at this time. NeXTSTEP is by far the most carefully engineered and easiest to use toolkit, but its limited availability is a problem in a multi-vendor, open environment. InterViews is a high quality C++ toolkit for X but has its own steep learning curve and lacks some of the 3D aesthetics of Motif. The interface builder (ibuild) is still undergoing development and should be much improved in the future.

Summary

It is too early to choose any one of the development systems used for the tuple viewer project over the others. Software and hardware are changing very rapidly so that the advantage of one system today may be available on the others tomorrow. However we have collected data and gained experience which will help in making the necessary decisions in the future.

We have also learned that developing a modest GUI interface will take on the order of a few weeks for someone who is experienced. Using GUI builders, constructing the visual aspects of the GUI is now a small share of the total task, and developing GUI's should be easier in the future as more working examples become available.

References

[1] *Proceedings of the Workshop on Physics and Detector Issues for a High-Luminosity Asymetric B Factory at SLAC*, SLAC-373 (1991).

[2] M. Edel, *Graphical User Interfaces*, Fermilab Comp. Note EN0039 (1991).

[3] Open Software Foundation, Prentice Hall.

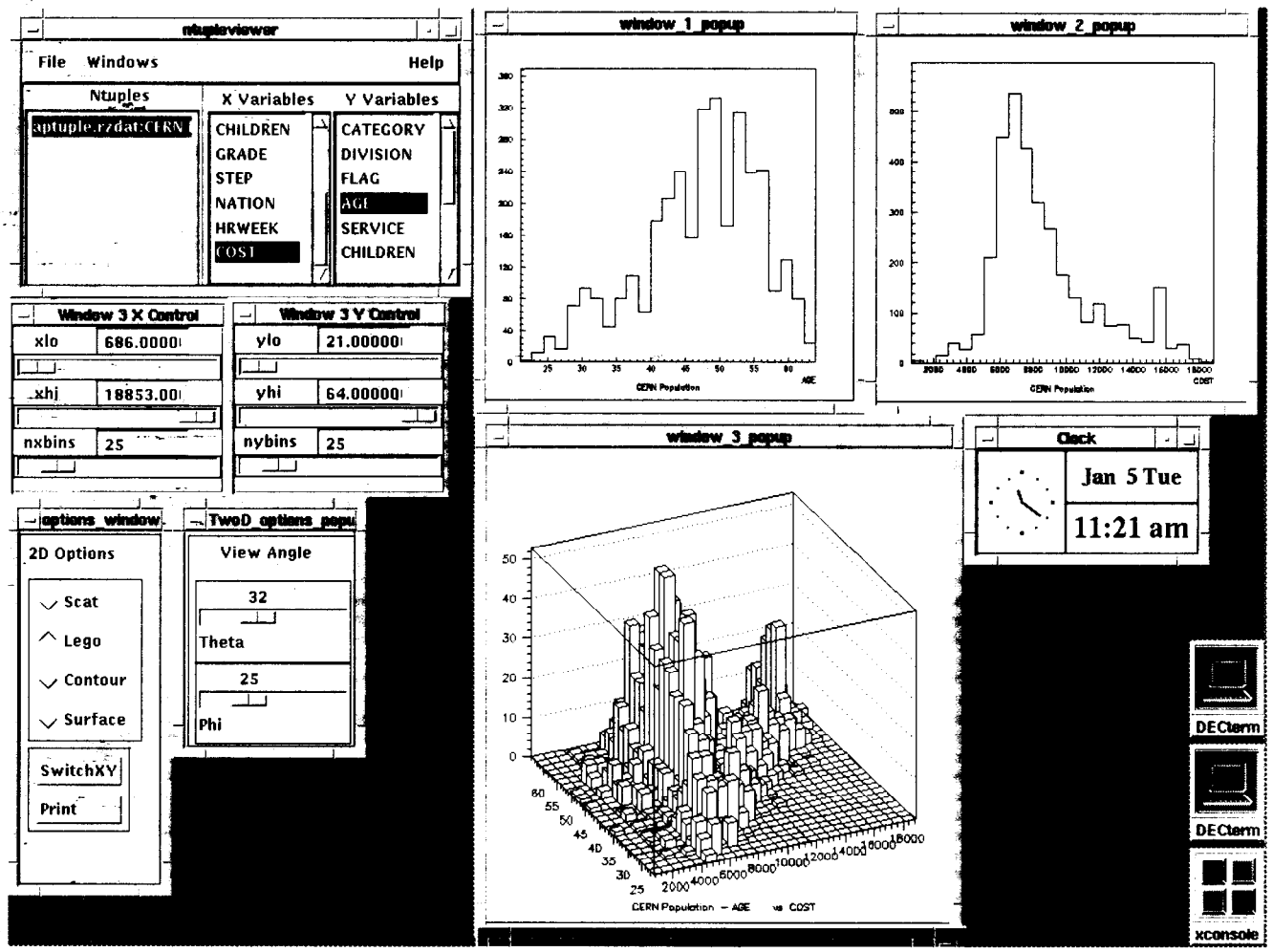

Figure 1: Ntupleviewer on DECStation 5000

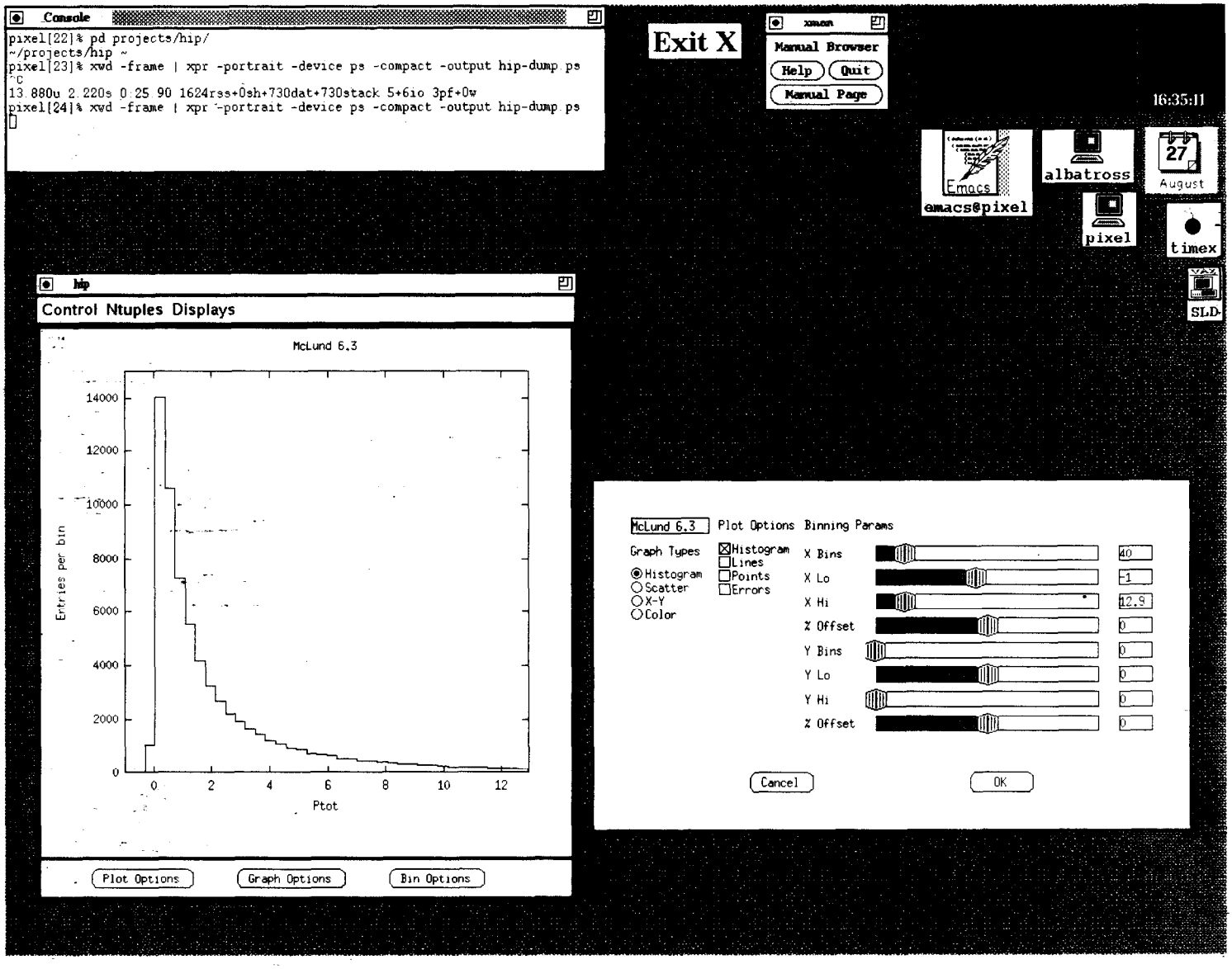

Figure 2: Ntupleviewer (hip) on Sun