An Experiment with Electronic Logs

Alex M. Waller
Fermilab
Accelerator Division Controls Department
MS 347
P.O. Box 500
Batavia, IL. 60510 USA

THE BRIDGE TO FUTURE GUIs

The Accelerator Division control system was proprietary since its inception in the late 1960s and early 1970s. In the area of graphics, in-house procedures were created and maintained to communicate through CAMAC to commercial graphics hardware and to proprietary console TV hardware. In the mid to late 1980s the underlying hardware and associated code for the console TV and auxiliary graphics was replaced with calls to X11 procedures as we moved our controls environment to DECWindows on VAX workstations[1]. It should be noted that the procedure calls were basically the same and the code was done in-house to now interface to X11. Our in-house graphical user interface support continued to grow with more value added procedures[2].

Moving to the X11 paradigm provided the bridge to the use of applications running on other platforms as more and more platforms and applications adopted the X11 protocol for graphics. One such utility was MetaCard[3], a Mac-like HyperCard[4] application that ran on many UNIX platforms. A more detailed account of this utility is given in the sections to follow.

An opportunity was waiting to try an experiment not only with integrating an alternate graphical interface with our Accelerator Division controls environment, but also with computerizing a long standing traditional way of recording log information. A relatively inexpensive license for MetaCard already existed. MetaCard behaved much as another popular utility that was available on the Mac, and thus several people were familiar with the HyperCard-like functionality and scripting. A request was pending from one of the Accelerator Division departments to venture into the field of computerized logs. The electronic log application was an excellent opportunity to gain experience and to introduce operations personnelto an alternate graphical user interface. It was also an excellent opportunity to explore the area of rapid prototyping.

THE ENVIRONMENT FOR CO-EXISTENCE

The VMS DECWindows Motif style of GUI provided a consistent windowing paradigm[5]. The underlying X11 protocol, already adopted for controls system graphics, co-existed very nicely with the entire console environment[6]. The entire MetaCard application was provided as a pull-down menu item on the console Session Manager menu bar. DECWindows allowed one to customize this menu bar and to interface menu items through the VAX DCL command language[7]. In this particular case the command is an RShell one. The RShell command provided the vital link to a host computer[8]. The host computer has a list of trusted users and nodes to allow remote login. Since the console environment on each node runs under its own account, a large list of user names was avoided. The menu item command also allowed for passing the display address for graphical output from X11. The user never sees the details of this command call, only the item on the menu list.

MetaCard itself runs within a window. A "home stack" (application) contains the icons of applications within a window. When MetaCard is started, it appears as just another window on the Accelerator Division controls consoles. Thus the operation of evoking MetaCard and working with MetaCard does not intrude on the current console environment. The MetaCard utility graphical user interface, however, is quite different from that of traditional console application programs[9].

WHAT IS METACARD?

A Motif-like graphical user interface using the X11 protocol

Three dimensional widgets such as push buttons, sliders and scroll bars are available and customizable in color and font just as in Motif[10]. Pull down and scrollable menus are available and customizable. Text areas called fields are also possible. More complex widgets, such as Dialog boxes, that are composites of text areas and push buttons are possible and customizable. Resizable overlapping windows with ornamentation are realized as individual "stacks" or applications in MetaCard.

Multimedia capable

Hypertext capability is possible when used in conjunction with the scripting language called MetaTalk. The script language is very similar to that used by HyperCard on the Macintosh computer and is used to control the flow of MetaCard operations much as in a conventional programming language such as C. One can also execute shell commands and fork other processes such as graphics browsers. One can include simple sounds within his/her stacks just as in HyperCard. Several formats of audio are also supported. One can use the AIFF, the Sun/Next (mulaw), and HyperCard 8-bit linear formats. Image support includes the popular UNIX formats pbm, xwd, xbm for pictures as well as encapsulated Postscript. For movie animation the Macintosh QuickTime format as well as AVI and FLC/FLI ones are supported.

Builds applications by rapid prototyping

Existing widgets (buttons, sliders, scroll bars) can be cloned by copy and paste from one application to another. This includes all attributes including colors, fonts and even any associated scripts. This can greatly aid in building up a new application from a previous one within a very short time. Tools are included to create new widgets with complete control of colors, fonts, labels, sizes and shapes. More complex widgets (such as dialog boxes) can be designed by grouping together more primitive ones.

A script can be attached to a widget so that the widget can react to various events such as mouse or keyboard input. As mentioned above a script is written in the MetaCard programming language called MetaTalk. There are control structures such as if-then-else and repeat-until/while/with. Functions can be written in C and are callable from MetaTalk so that the scripting language can be extended. The scripts are executed by events or direct calls from within other scripts. Mouse up/down, drag and in are the primary events that activate scripts. Other events such as a "stack" (application) open or close can also elicit an action.

Event simulation can be accomplished by sending an event to an object from another script. Widgets can also react to messages (events) which are sent up the object hierarchy. For example if a script to handle an event does not exist in one widget, the event message continues traveling up the widget hierarchy in the "stack" until a widget with an appropriate event handler will consume (act on the event with a script) the message. A script within a widget can even continue to pass the message on up the hierarchy so that other widgets may process the event message.

Finally a drawing editor is provided so that users can create their own graphics and icons.

THE ELECTRONIC LOG BOOK

User criteria

To be successful an electronic log book had to emulate a hand-written log closely. Not only that but there were some criteria imposed on electronic media that did not exist with a hand-written log. One of these was that once was made a log entry could never be erased or modified. This led naturally to the criterion that each log entry be able to have an addendum that, again, once entered could not be modified. Each log entry could be time-stamped automatically rather than have the author enter the date and time as in a hand-written log. To make the electronic version of a log more useful it could be given a table of contents so that an entry could be found much more quickly than in a conventional log book. In conjunction with the table of contents, word and phrase searches could be possible so that pieces of information, which might not appear in the table of contents, could be located quickly. Finally, the electronic log processmust be able to capture and insert figures within the log book.

Design criteria

An electronic tablet form was chosen (see Figure 1). There is a large writable area. No scrollable text is permitted which might hide text. The entire log entry is visible. All header information such as author, date and title of entry appear at the top of the tablet. All controls appear at the bottom of the tablet. The log was designed for a large screen display so that a large enough font could be used for legible text. One can actually take a screen snapshot of the log tablet, and this entire entry will fit within the bounds of an 8.5 by 11 inch sheet of paper when printed thus appearing to be almost the size of an actual hand-written log.

Since the size of a captured image could vary, and even be quite large, a link to the image is made with a button that contains a figure number (incremented automatically by the electronic log application) and a caption that is entered by the user. Any number of links can be inserted into the text field of a log entry (see Figure 2).

Also on-line help exists for the electronic log book. This help was done in a hypertext fashion so that the user can go to other useful information without having to search through the help text for more.

Operational experience

As the electronic log book was increasingly used operational experience was gained. Some important modifications were made to the electronic log. One of these that was interesting was due to the initial design specification. The requirement to have log entries be unmodifiable, although mandatory for acceptance of electronic media for logging, proved to be humanly restrictive. In actual practice log entries do get modified. Usually space is kept for additional information or additional information is side-barred. A modification that seems to have worked was to lock all entries only after the electronic log book was closed. As long as the log book remained open (the application running) any entries made during the current opening of the log could be modified. To ensure that the log book would eventually be closed, a count-down timer was put into the application to close out the log book after several hours of inactivity. Since MetaCard is event driven it was very easy to use keystrokes and mouse events as activity measures.

It is interesting to chronicle the effects in making a modification. After the seemingly innocent minor change to allow log entries to remain modifiable, as in a physical lever a larger operational problem was created. Over long editing sessions, especially since the log book could now be left open over a long time so that log entries would not be write locked, it became necessary to manually save the log entries on demand. Several occasions with the momentary loss of either the server node or host node made this modification necessary.

The table of contents window was pressed into service as a more sophisticated electronic log book navigator. The sequential navigation to previous or next pages within the log book was restrictive. Double-clicking on an entry in the table of contents allows one to go directly to the selected entry (see Figure 3).

The ability to display a reduced image of any given log entry and its addendum was added. This aided in allowing one to reference one log entry while making another (see Figure 4).

Since several electronic log windows could be opened when one wished to take a snapshot it was sometimes difficult to isolate the window that was a target for image capture. MetaCard allows windows to be hidden and to be made visible. This feature proved useful in helping with snapshot taking. All open electronic log windows, including the log book, were temporarily hidden while the snapshot was in progress. After the snapshot was taken, all log book windows that were made invisible were again made visible.

Another interesting issue related to taking snapshots had to do with the design of the Accelerator Division controls console hardware. Each console could consist of up to four physical display screens. Fortunately there was a network naming scheme for these. The existence of such displays could be discovered, and a special dialog box was created requesting the user to select one of the possible four quadrants in the dialog box (corresponding to the four physical displays) before the snapshot could be taken. The information returned from the dialog box was used to insert the proper display address into the snapshot procedure.

Figure 1
The tablet format of the Electronic Log Book

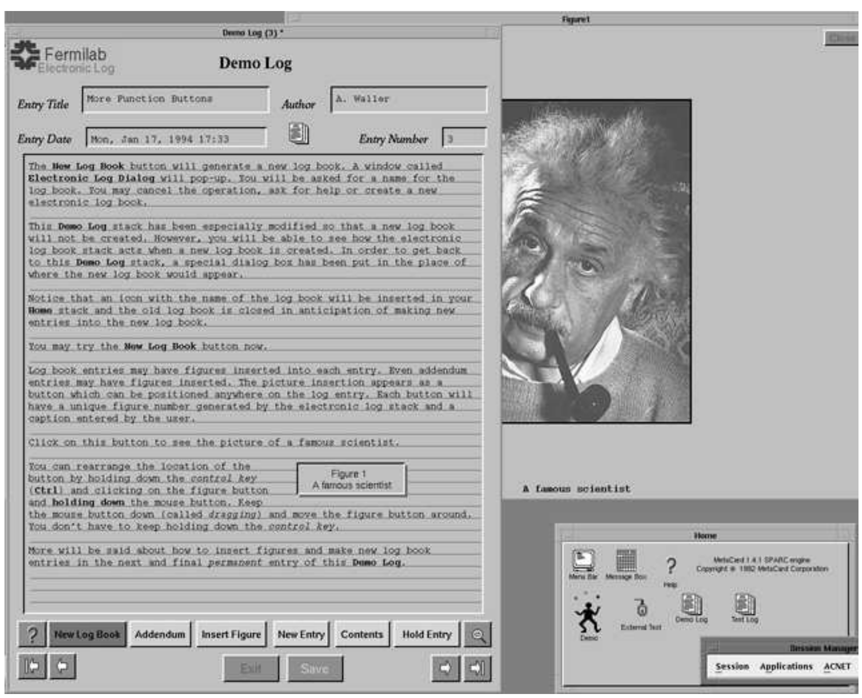


Figure 2
Button (lower right section in tablet) with figure number and caption serve as link for the real image

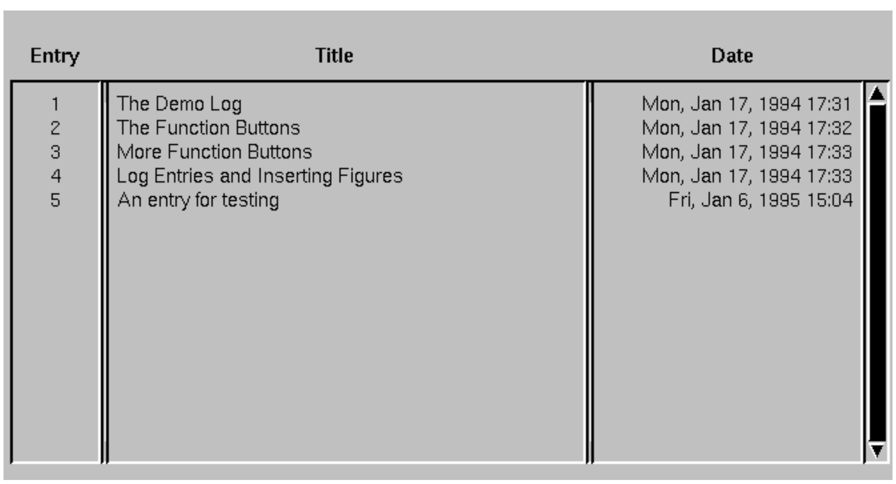
METACARD EXPERIENCES GAINED THROUGH THE ELECTRONIC LOG BOOK

All text, graphics and MetaCard "code" called scripts reside together as a single entity called a stack. The power of MetaCard lies in utilizing this paradigm. This approach enables fast text searches through an entire electronic log with the embedded MetaCard search engine. No special code needs to be written for reading and searching text records within a file. This makes text searching an integral part of the MetaCard functionality rather than a separate part that either must be programmed each time or provided for by an auxiliary utility. Multiple users of a log book must however share the entire application not just the text (data) associated with it. The electronic log book must use a semaphore to synchronize the writer for a given stack (log book).

As new text is continually added the stack continues to grow. If pictures are stored in a log book this can make for an extremely large stack, and the author of MetaCard suggested storing only one image per stack. Images are stored immediately after the snapshot whereas the text entered into a log book may be cached before being written out. Storing the images separately has proved to be a good idea. They are not embedded within the log book and thus are available separately for perhaps reproduction for a report. Also, on the few occasions where the host or server computer has momentarily been down, the pictures have been recoverable and links able to be re-established to the necessary log book entries.

One of the useful features of MetaCard is the ability to clone all objects, including the entire stack (or application). Cloning an application is not advisable unless the application is really going to be used in a different context. For the electronic log book, each new or different log is cloned from an original copy. If one modifies a script that is a part of the log book functionality, perhaps because of a bug, this modification must be made to all existing stacks of electronic log books.

Initially the MetaCard snapshot utility captured pictures quickly and stored the image efficiently. However this utility could not handle multiple displays as other UNIX image capture utilities could. Others were tried but proved to be slow and/or not very efficient in storing the picture information. Eventually the author of MetaCard extended importing snapshots from other displays.



Entry	Title	Date
1	The Demo Log	Mon, Jan 17, 1994 17:31
2	The Function Buttons	Mon, Jan 17, 1994 17:32
3	More Function Buttons	Mon, Jan 17, 1994 17:33
4	Log Entries and Inserting Figures	Mon, Jan 17, 1994 17:33
5	An entry for testing	Fri, Jan 6, 1995 15:04

Figure 3
Double-clicking on a log entry in the table of contents will take one to the entry

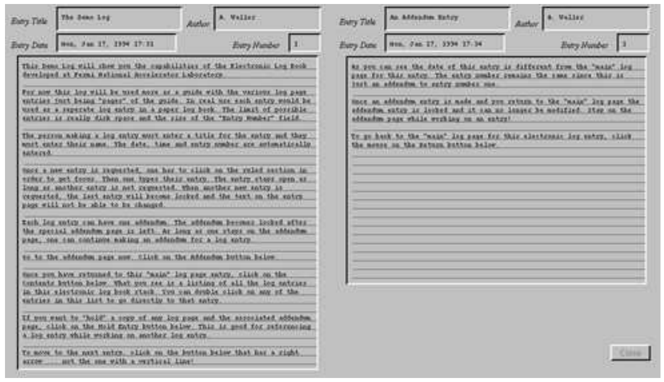


Figure 4
A reduced window of a log entry and its addendum can be displayed for reference

CONCLUSIONS

Rapid prototyping of the electronic log book allowed the visual layout to be done within hours. Writing scripts and learning more about the use of MetaCard took at least a few weeks. Most of the time was however spent on the learning curve for MetaCard. A knowledgeable MetaCard user would be able to write the necessary scripts for the electronic log within a few days. The rapid prototyping approach therefore does appear to be attractive for developing applications; especially in a situation where personnel and time are limited.

The image capture is a very valuable feature of the electronic log book. As it turns out, a log book may contain several hundred log entries but usually contains several thousand pictures. The author of MetaCard is to be thanked for his suggestion of keeping images in separate stacks detached from the main log book entry.

There are several concentrated users of the electronic log book for the Booster, Antiproton and Tevatron operations of the accelerator. While the log book is available to all operations, its acceptance has been mixed. This mixed acceptance has been partly because it was well known that this project was an experiment and as such carried an experimental stigma about it. Operations is apprehensive to use an experimental system, and it is constantly thought that this project, though now finished, is still under construction. Also the graphical user interface is quite different from the day-to-day interface operatiors must deal with. Insofar as this difference exists, the application itself is treated differently and is not warmly accepted as a part of the operational control system. However, the author of this paper believes that this will change in time as more applications with a Motif look and feel become available and a vital part of the Accelerator Division controls console environment.

We are pleased with the seamless integration of the electronic log book. Many people are surprised to discover that the code is actually running on a remote host. Many have thought that the electronic log book was running on the VAX VMS workstations that are used to support Accelerator Division controls consoles.

ACKNOWLEDGMENT

I would like to thank Scott Raney of the MetaCard Corporation for his many suggestions during the development of the electronic log book. Scott also provided technical answers to my MetaCard questions and provided some bug fixes and technical solutions to accommodate and help make the electronic log book a success.

REFERENCES

[1] K. Cahill and J. Smedinghoff, Converting the Fermilab Accelerator Control Consoles to X-Windows Workstations, Nuclear Instuments and Methods in Physics Research, Volume A293 (1990), pp. 442-445.

[2] B. Hendricks and R. Joshel, Overview of the Next Generation of Fermilab Collider Software, National Laboratory for High Energy Physics, KEK Report 1992, pp. 243-245.

[3] S. Raney, Interactive GUI Development Environments; A Comparison of Tcl/Tk, the Desktop Kornshell and MetaCard, O'Reilly and Associates, The X Resource, Issue 11, Summer 1994.

[4] HyperCard User's Guide, Apple Computer, Inc., Number 030-3918-A, 1989.

[5] Using DECwindows Motif for OpenVMS, Digital Equipment Corporation, 1993.

[6] K. Cahill and J. Smedinghoff, Exploiting the X-Windows Environment to Expand the Number, Reach, and Usefulness of Fermilab Accelerator Control Consoles, National Laboratory for High Energy Physics, KEK Report 1992, pp. 464-467.

[7] OpenVMS DCL Dictionary, Digital Equipment Corporation, 1994.

[8] MultiNet User's Guide, Digital Equipment Corporation, 1994.

[9] J. Smedinghoff, The Fermilab Accelerator Control System Parameter Program and Plotting Facility, Nuclear Instruments and Methods in Physics Research, Volume A247 (1986), pp. 172-175.

[10] OSF/Motif Programmer's Reference, Prentice-Hall, ISBN 0-13-640517-7, 1990.