

GRAPHICAL USER INTERFACE CONCEPTS FOR HYDROLOGIC FORECASTING IN THE MODERNIZED WEATHER SERVICE

Thomas E. Adams, III

Hydrologic Research Laboratory, National Weather Service
National Oceanic and Atmospheric Administration
Silver Spring, MD

1. INTRODUCTION

1.1 Past and Current Modeling Procedures

Most National Weather Service (NWS) hydrologic forecasters are accustomed to hydrologic and hydraulic modeling, with a wide range of models available for flood event modeling, simulations of seasonal and annual streamflow, and hydraulic simulations in dam break analyses. The *procedure* of modeling has evolved over time. Digital simulations of hydrologic processes through the 1970s required the modeler to enter observed data and parametric values onto paper punch cards using a key-punch device. The card deck was then read into a card reader of a mainframe or minicomputer and executed on the computer following the instructions detailed by the job control language (JCL) encoded on some of the cards. Typically, simulation results were reviewed from a lineprinter. When changes were made to improve simulation results, new data cards and, often, JCL cards were made to reflect the needed changes. Subsequently, the card deck was resubmitted for a new run, typically in a batch mode where user's jobs waited in a queue based on the priority given to the job by the user and system administrator. A typical job run cycle could take hours.

The procedure more common now, is the creation of data files by entering data into ASCII-based terminals or microcomputers using a text editor which is, in turn, linked with pre-compiled models and executed. Simulated values are then sent to an output file in a format that can be read by an auxiliary graphics package to display on a CRT (cathode-ray tube) of a graphics terminal, microcomputer, or scientific workstation and are printed or plotted later on some device if the user desires. Nevertheless, lineprinter output remains the primary expression of output for many hydrologists today. Hydrologic modeling still consists, for the most part, of allowing the model to fully complete execution before parametric values are adjusted to improve simulated results and the model is rerun. Thus, the modeler has no opportunity to halt execution of the model during a run cycle when simulated values clearly need improvement. Moreover, the modeler's interaction with the computer is less than seamless or friendly; only fairly recently have preprocessors, programs with a degree of error checking that make creating data files in the correct format easier, gained general acceptance for the most widely used programs.

1.2 Interactive Modeling Procedures

To reach its modernization goals, the NWS has implemented risk reduction plans, including project PROTEUS (Prototype RFC Operational Test, Evaluation, and User Simulation) in the Office of Hydrology. The purpose of project PROTEUS is to demonstrate new and innovative uses of computers and software in the NWS River Forecast Centers (RFCs) to improve

hydrologic forecasting. In contrast to character-based, command-line computer interfaces for models, interactive modeling should permit the user to quickly and efficiently enter observed and parametric data, begin execution of the model, monitor its progress, and halt program execution at appropriate times. Entering and changing parametric data is best accomplished using a graphical interface that simplifies entering individual values, series of values, dates, and choosing between discrete items. Interactively, changes are made by using such CRT screen display entities as buttons, windows, popup and pull down menus, scrolled lists of items, and other graphical icons with real-world analogues (e.g., a pencil to represent a drawing tool). Similarly, CRT screen displays of results should make use of color and other graphical cues — that are possible on today's microcomputers and scientific workstations — because they enhance the visual impact of graphically displayed data, especially when multiple data sets are superimposed on the computer screen. The user of the model interacts with the graphical items on the screen with the computer's mouse, an electromechanical device that moves a pointer (usually an arrow) on the screen and has buttons for selecting menu items, pushing buttons, moving windows, and scrolling through lists. Graphical objects are immediately accessible to the user, usually requiring no more than a movement of the mouse and a mouse button click. The greatest benefit of a graphical user interface (GUI) is that users no longer need to remember cryptic commands that must be typed using the computer's keyboard. Furthermore, the possible choices of what can be done within a GUI are obvious and certain, once a few basic rules of GUI interaction are understood. If properly planned and constructed, the interface both limits what the user can do and provides the user with a great deal of freedom. This seemingly contradictory statement makes sense within a GUI design. For instance, the main menu of a program lets the user access all the commands through pull down and hierarchical menus, but not all the commands listed in the menus may be immediately available because something else must be done first: namely, before a model can be executed using an **Execute** command (a menu item), a data set must be selected. In this example, the word **Execute** would be dimmed or grayed and would not highlight when the pointer moved over it. Consequently, if the user tried to select **Execute** without having previously selected a data set, nothing would happen. The visual cue of dimming or graying lets the user know that trying to select that particular menu item will accomplish nothing.

2. GUI CONCEPTS — USER INTERACTION WITH GUIs

The two underlying principles of graphical user interfaces are:

- (1) ease of use, and
- (2) functional design to increase productivity.

That is, GUIs make computers better tools to do the work at hand. But writing good GUI software is no easy task, since the programmer must anticipate the user's needs and demands in meeting the two fundamental GUI goals. Writing GUI software is easier under project PROTEUS than it may have been otherwise, because of the process of first prototyping software at the NWS Hydrologic Research Laboratory (HRL), then field testing and evaluating the software at the RFCs, and, finally, modifying the code at the HRL in response to suggestions from the RFCs.

Graphical user interfaces for programs, whether they include hydrologic models or not, are possible because they have been programmed using the concept of events. In fact, event driven programming begins with a single infinite loop in the program structure. The user of an event-driven program leaves the main event loop by selecting a Quit command, which ends all processes, closes all open devices, and otherwise terminates the program gracefully. All tasks initiated or subroutine calls made subsequent to the initial start-up of a program written with a GUI are made from within the main event loop: moving the mouse (the screen pointer) evokes an event, as do pressing a mouse button, releasing a mouse button, pressing a key on the keyboard, releasing a key on the keyboard, moving the pointer into or out of certain screen objects, etc.; many other types of events are possible depending on the GUI implementation. All programs, written with a GUI, process events as they "arrive" in the main event loop where subroutines are called to handle events of each particular type. An event is discarded after it has been dealt with appropriately by the program.

Another fundamental element of a GUI is the "window," a graphical object used to hold buttons, scrollable lists, popup menus, and other control objects and to display results (Fig. 1).

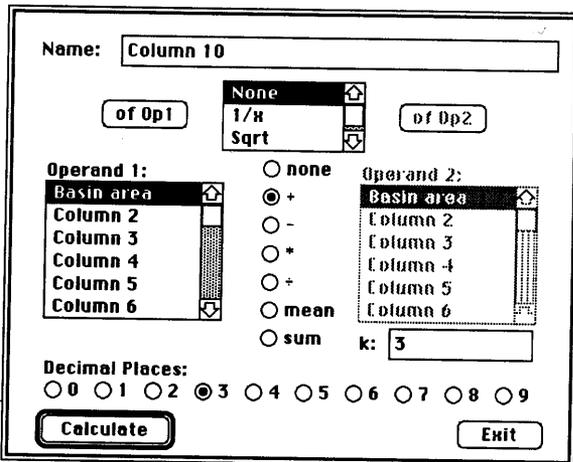


Fig. 1. A popup dialog window showing three scrollable lists with a "grayed" Operand 2 area indicating the user has chosen to enter a "k" value; the Calculate button is emphasized indicating to the user that a carriage return will evoke the same result as a mouse click on the button.

In other kinds of programs, windows are used as a drawing surface for computer aided design (CAD), a typed page in word processing, and spreadsheet for financial and engineering analyses. Typically, windows are resizable and can be moved about on the screen by the user, but specific implementations of these features vary between GUI systems. With many GUIs, windows can be iconified to unclutter the screen, that is, they can be reduced to a small system-defined icon sometimes a pictorial representation of an object that holds meaning for the user. Windows, however, can have certain attributes to convey unique information to the user of a program, such as popup windows that appear on the screen to alert or warn the user to

some problem in the program, operating system, or some error the user has committed (Fig. 2).

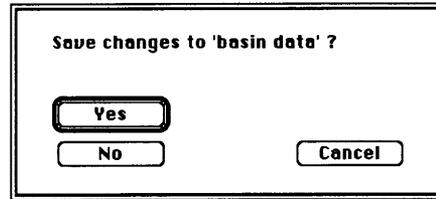


Fig. 2. A popup dialog window warning the user that the most recent changes to a file have not been saved; note the emphasized Yes button, which indicates that the a carriage return will evoke the same result as a mouse click on the button, also showing is a Cancel button.

Some guidelines for user interaction should be followed:

- (1) the user selects an action from alternatives presented on the screen;
- (2) the user first selects an object, then the action to be performed on it. The noun-then-verb logic is very flexible because, at the last moment, the user may wish to choose a different action and does not have to reselect the object. For example, this is analogous to a word processing program having the user first select a word, line, paragraph, etc., then italicize, underline, delete, make boldface, change the point size, etc., rather than first entering an italicize mode and then selecting the word once the user is in an italicize mode, he or she has a difficult time changing to another, making the interface more cumbersome;
- (3) users rely on visual recognition, so graphical analogues to real world objects should be used to convey information, such as a caricature of a stop sign in a popup dialog box to warn the user of an irreversible mistake he or she is about to make;
- (4) screen the user from unnecessary system level and some higher level details that are not needed by the user to complete a task;
- (5) simple and consistent keystroke equivalents to menu choices and other functions; in unambiguous cases, rather than forcing the user to select a push button with the mouse, let the user use a carriage return in its place as an alternative this is particularly appropriate when the user is entering a numeric value from the keyboard and needs to enter the next value or wishes to continue to some other task.

The need to anticipate that users make mistakes is important. In most cases it is possible to either undo some action or back out from it, using a Cancel button (Fig. 2 and 3).

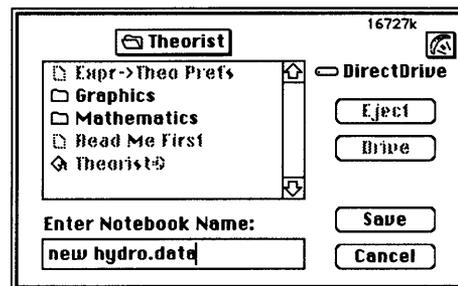


Fig. 3. A popup dialog window showing a Cancel button, scrollable list, and "grayed" Eject and Drive buttons indicating there is no floppy disk to Eject and no other available hard disk or optical drive to switch to for the Save operation.

Also important is the need for simple uncluttered design of windows where buttons, scrolled lists and static text are cleanly laid out and are aesthetically pleasing. Simple design is best, the screen should not be cluttered with too many windows, complex icons, and buttons. Good graphic design must communicate and inform, not just dazzle the user. *Graphics are not merely cosmetic; when they are clear and consistent, they contribute greatly to ease of learning the interface and using it.*

3. IMPLEMENTATION OF GUI CONCEPTS IN THE INTERACTIVE FORECAST PROGRAM

The Interactive Forecast Program (IFP) of the National Weather Service River Forecast System (NWSRFS) (Anderson, 1986) was written to run on any of a range of available scientific workstations. Consequently, the basic graphical windowing environment for the IFP is the industry standard X Window System (version X11R3) developed within project Athena at the Massachusetts Institute of Technology. X is supported by the leading computer manufacturers and many universities, forming the X Consortium, with the intent to further develop and promote the use of X as a system independent and non-proprietary windowing and communication environment for computers. Some preliminary remarks on some of the details of X are needed because many features of X strongly influenced the development and look and feel of the IFP.

Although the X Window System is not necessarily tied to a single language or operating system, it is rarely found on computers not running the Unix operating system and programming calls made from any other language than C are more rare. Consequently, it is safe to generalize that the X Window System is used on Unix-based workstations and X applications are written in C. This is true for the most part with the IFP, except that there are FORTRAN subroutines called within the IFP code which, in turn, call C functions that, sometimes, call more FORTRAN subroutines. The intermingling of languages is inescapable, because the existing NWSRFS program is written, for the most part, in FORTRAN and rewriting this code in C was not considered a viable option since it comprises on the order of 40,000 lines of executable FORTRAN statements. Additionally, a significant portion of the IFP code was generated using a commercially available graphical programming tool for creating the C code that displays simulated values and observed data.

The X Window System is comprised of two fundamental components, (1) the base window system, which provides the lowest level interaction with the computer to create windows and (2) the X network protocol, which is based on the client-server model. A single process, the server, controls all input and output devices, such as the CRT screen, mouse, and keyboard. The server in X is also called a display whereas the term "screen" in X refers to a single hardware output device. Therefore, a single X display can support multiple screens, however, there is usually only one display supported by each CPU. The server also creates and manages all resources, which include windows, text, bitmaps and pixmaps, colors, and other data structures used by an application. The X server maintains resources privately, allowing clients to use and share them transparently. Applications that use the server's facilities, termed clients, communicate with the X server over a network connection using many of the common asynchronous byte-stream protocols, including TCP/IP (Young, 1989). Consequently, any client can communicate with any server provided they both adhere to the X protocol.

The base window system interfaces with the outside world through the X network protocol. The network protocol interface operates both within a single central processing unit (CPU) or between multiple CPUs, consequently, X is device and vendor independent and it operates transparently over networks using, typically, the TCP/IP protocol. The only possible interface with X is through the X network protocol, implying that any software, including X Window Managers are treated as application software by X rather than privileged system software (Jones, 1989).

Programmers access the network protocol through a combination of XLib, a C language subroutine package, or a higher level X toolkit object library. The benefit of using XLib and an X toolkit is that the programmer is shielded from the overwhelming complexity of the network protocol. X toolkits (or widget sets) are particularly useful because they provide high level graphical objects that can be easily implemented by a programmer. Use of a X toolkit forces the programmer into consistent interface design and greatly speeds application development because fundamental graphical objects, such as push-button and scrollable windows, do not need to be recreated, and the desired attributes of windows do not need to be custom programmed each time a new object is needed. It is important to note that all buttons, boxes, sliders, and other graphical elements of windows in any window in X is itself a window a rectangular region on the screen.

The IFP, which is written in a modular form, consists of eight unique programs that serve independent functions. This modular design is possible because data that must be shared between the separate programs are available to each module as X window properties. A window property is a mechanism provided by the base window system to share information between applications (Fig. 4).

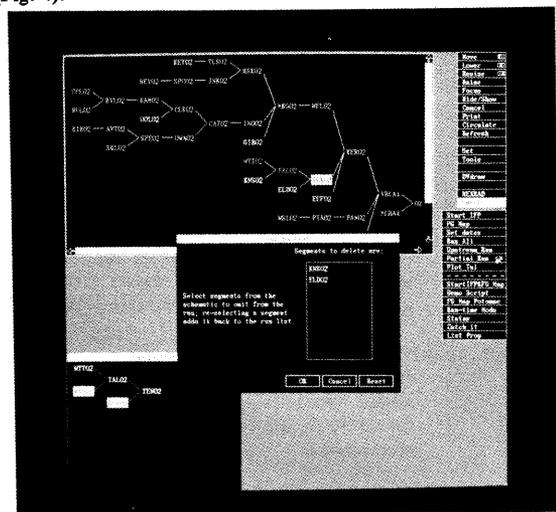


Fig. 4. Windows for the FG_Map and Run_Partial modules with a selected and highlighted forecast point, indicating the most downstream point for this forecast run. Note that two forecast points have been excluded from the simulation because rainfall in those areas are contributing insignificantly to runoff; also shown is the window manager window.

Applications must be written in a manner to explicitly access window property data of a specific type, either as the data appear as new window properties or at specific points within the program when the data are needed. Each of the modules was designed from the perspective of the user and the user's needs, rather than avoiding the associated programming difficulties. The eight modules comprising the IFP are:

- (a) Start_IFP
 - is used by the forecaster to select a forecast group (a group of contiguous and topologically connected basins) to model;
- (b) FG_Map
 - is used to display a schematic representation of the forecast group drainage system where the nodes of the schematic are the forecast points; it is also used as a basis for selecting what basins are included in a modeling run and to display information pertinent to the

forecast point, such as the peak flow of record, location in degrees latitude and longitude, etc.;

- (c) **Set_Dates**
 - is used by the forecaster to set the dates for the forecast period, including the beginning and end of the model run and the end of the period used for observed data;
- (d) **Run_All**
 - allows the forecaster to select all the basins within the forecast group and begin execution of the models;
- (e) **Run_Upstream**
 - allows the forecaster to select all basins above and including the basin selected within FG_Map within the forecast group and begin execution of the models;
- (f) **Run_Partial**
 - allows the forecaster to select all basins above and including the basin selected within FG_Map and exclude other unwanted basins, that do not break basin drainage continuity, within the forecast group and begin execution of the models;
- (g) **Plot_Tulsa**
 - displays as line plots and bar charts the results of the model runs, observed rainfall and runoff time series, and input unit hydrographs and allows the user to edit these time series; this function also provides the controls for re-running the models based on the modifications or continuing model execution with the next downstream forecast point;
- (h) **Model Specific Modifications**
 - allows the forecaster to make changes to certain parameters on a model by model basis for the basin (forecast point) currently under consideration.

Readers are referred to Wiele and Smith (1991) for a discussion of NWSRFS models and their dependencies and interactions. Page (1991) discusses details of the eight interface components of the IFP and their use within a forecast session.

Each of the IFP programs is written to provide immediate feedback if the user attempts an action that is inappropriate or would produce an error. In cases that, for instance, the forecaster may mistakenly enter an ending date for a forecast run that was chronologically before the beginning date, the Set_Date program simply will not accept the value and responds with a "beep." Similarly, the forecaster is simply warned by a message within a popup window of a value's acceptable range if the value being entered falls outside this range. The program responds immediately as the user types the digits. If the user continues, attempting to enter a value that falls further outside what is generally considered permissible, or even approaching what is considered physically unacceptable, the program issues a "beep", does not allow the entry, and indicates within the popup window the value's permitted range. Similar features that give the user immediate feedback to his or her actions are found throughout each of the IFP modules. Many of these, where appropriate, do not allow the forecaster to make avoidable mistakes.

4. FUTURE OF GUIs ON WORKSTATIONS AND IN THE IFP

It is clear from the success of graphical user interfaces on the Macintosh from Apple Computer, Inc., Windows (especially Windows 3) from Microsoft Corp. for DOS based computers, and

the growing interest in OS/2 and Presentation Manager from IBM Corp. and Microsoft, that the X Window System and interfaces derived from it have clear advantages to and bring significant productivity increases over character based user interfaces.¹ Commercially developed X Window based applications are lagging behind the interest and demand for them at present, but this is due, in part, to the recent stability achieved by X and the X toolkits, and the time needed to develop an X application. GUI applications take longer to develop than character based applications, but the trend with character based programming has been to make them behave graphically, with menuing and popup windows, etc., which increases development time. Another important issue is the consistent look and feel between applications that have been programmed within a particular GUI development environment. A user of a program can much more easily learn how to use a new program if it is constructed similarly with menus and menu items, etc., that achieve the same function and behave the same across all applications. This is now being done on computers within individual GUI systems and operating systems and also between totally different computers, operating systems, and GUI development systems with many commercial applications.

However similar GUIs may be in their look and feel with windows, menus, buttons, scrollbars, etc., their performance differs greatly between (a) GUI systems on a specific computer, (b) specific GUI implementations on the same computer, and (c) general performance on different computers, operating systems, and GUI development environments where direct comparisons are not possible. Specifically, it has been the experience of the HRL staff, using X version X11R3 and version R2 of Xlib and the Hewlett-Packard X toolkit (widget set), Xt, on an IBM PC-RT that GUI performance was comparable to that found using a pre-release X version X11R3, Xlib version R3, and version 1 OSF (Open Software Foundation)/Motif X toolkit (widget set) on an IBM RISC System 6000. This is surprising since in all other respects, especially involving floating point calculations, the IBM RISC System 6000 greatly outperformed the IBM PC-RT by a factor of approximately 10 (ten). The disparity could be due to a number of factors, (1) that HRL staff had made the evaluations using pre-release software on the IBM RISC System 6000, so that certain optimizations were not realized, (2) that IBM's specific implementation of the OSF/Motif widget set reduced the performance, or (3) that there is much more software overhead in the OSF/Motif widget set, and performance is simply lagging. Of course, any combination of these is possible and, in all likelihood, probable. Moreover, without a detailed, careful evaluation, these comments are merely anecdotal, as HRL has had no opportunity to evaluate the code on other workstation platforms.

Nonetheless, it is generally known among X application developers that the GUI elements of X applications suffer from a lack of graphic responsiveness. This is not considered a major problem in light of the benefits of using X, Xlib, and the various X toolkits. Developers of X, Xlib, and the X toolkits are committed to enhancing performance through coding optimizations. Also, third-party hardware developers are making custom X graphics accelerator boards available for workstations and some workstation manufacturers are now designing their computers with accelerators for X graphics.

The National Weather Service is committed to GUIs in its modernization efforts and with the IFP, however the NWS has not committed itself to either a specific GUI implementation or workstation platform. Although, a Unix based workstation is

¹The use of trademarks and trade names is for descriptive purposes only and does not constitute endorsement by the National Weather Service, or her parent organizations, the National Oceanic and Atmospheric Administration and the U.S. Department of Commerce.

virtually certain, one of many GUI implementations, NextStep, Motif, Open Look, SunView, or some other, is possible.

5. SUMMARY

The Hydrologic Research Laboratory using the development tools available with the X Window System, its associated object library XLib, and other X toolkits, successfully developed the Interactive Forecast Program as a Graphical User Interface for NWSRFS, the National Weather Service's national flood forecasting system. The IFP includes user interface design elements to simplify the use of NWSRFS and increase the productivity of forecasters. The IFP will be enhanced further based on comments made by forecasters at the River Forecast Centers during imminent test and evaluation periods.

6. ACKNOWLEDGEMENTS

This work is supported, in part, by the Advanced Weather Interactive Processing System (AWIPS) program office. The author would like to thank Drs. Danny Fread and George Smith for their review of the manuscript and Mmes. Elaine Hauschildt and Virginia Radcliffe for their assistance in its preparation.

7. REFERENCES

- Anderson, E.A., 1986: The National Weather Service River Forecast System and Its Application to Cold Regions, *Proc. Sixth Northern Research Basins Symposium/Workshop*, Houghton, Michigan, January 29, 1986.
- Jones, O., 1989: Introduction to the X window system. Prentice Hall, Englewood Cliffs, New Jersey.
- Page, D., 1991: The interactive NWS river forecast program. *Proc. Seventh International Conference on Interactive Information and Forecasting Systems for Meteorology, Oceanography, and Hydrology*, New Orleans, Amer. Meteor. Soc.
- Wiele, S. M., and G. F. Smith, 1991: Improved hydrologic forecasting with the interactive NWS river forecast program. *Proc. Seventh International Conference on Interactive Information and Forecasting Systems for Meteorology, Oceanography, and Hydrology*, New Orleans, Amer. Meteor. Soc.
- Young, D. A., 1989: X window system programming and applications with Xt. Prentice Hall, Englewood Cliffs, New Jersey.

