Strategies for Creating an Easy to Use Window Manager with Icons

Brad A. Myers

PERQ Systems Corporation
2600 Liberty Avenue
P.O. Box 2600
Pittsburgh, PA 15230
USA

author's current address:
Department of Computer Science
University of Toronto
Toronto, Ontario, M5S 1A4
Canada

ABSTRACT

Sapphire is a powerful window manager for the PERQ personal work station. It was designed to facilitate the single user's ability to monitor and control many different processes operating in parallel and running in different windows. Sapphire contains a full implementation of the covered window paradigm (where the rectangular windows can overlap like pieces of paper on a desk). In order to make it easier to control *processes* Sapphire provides icons that display six pieces of dynamically changing state information about the process running in the associated window. In order to make it easier to control the *windows*, Sapphire provides a powerful set of operations including: top, bottom, move, grow, reshape, full-screen, back-from-full-screen, off-screen, back-on-screen, and many others. These commands are presented in a way that is easy to use and self-explanatory for novices, without penalizing the expert. Almost all commands can be given either by using the pointing device or the keyboard. Unlike other window managers, Sapphire also allows all commands to be given using only the keyboard for users that do not wish to use the pointing device. This paper describes the user interface to the Sapphire window manager.

RESUME

Sapphire est un puissant système de gestion de fenêtres pour le poste de travail individuel PERQ. Il a été conçu pour permettre à un utilisateur unique de surveiller et contrôler plusieurs processus actifs en parallèle dans plusieurs fenêtres. Sapphire inclue la réalisation complete du concept des "fenêtres couvertes" (où des fenêtres rectangulaires peuvent se couvrir mutuellement comme des feuilles de papier sur un bureau). Pour faciliter le contrôle des *processus*, Sapphire fournit des icones qui affichent chacune dynamiquement six informations sur l'état du processus correspondant à une fenêtre. Pour faciliter le contrôle des *fenêtres*, Sapphire offre un ensemble puissant de commandes, comme: haut, bas, déplace, croit, change-forme, plein-écran, retour-de-plein-écran, hors-écran, retour-à-l'-écran, et beaucoup d'autres. Ces commandes sont offertes de façon à être facile à uliliser et evidentes pour les novices, sans handicapper les experts. Presque toutes les commandes peuvent être données soit par l'intermédiaire d'une souris ou d'un clavier. A l'encontre de la plupart des systèmes de gestion de fenêtres, Sapphire permet aussi de donner toutes les commandes en ulilisant uniquement le clavier pour les utilisateurs qui ne désirent pas utiliser la souris. Nous présentons ici l'interface de l'utilisateur au système de gestion de fenêtres Sapphire.

Key Words and Phrases: Windows, Covered Windows, Window Manager, Personal Work Station, Pointing device, Icons, User Interface, Interaction Techniques.

1. Extended Summary.

Sapphire (the Screen Allocation Package Providing Helpful Icons and Rectangular Environments) is a very powerful window manager running on the PERQ personal work station [Rosen 80]. The PERQ has a high resolution screen and special hardware that can display graphics quickly. Sapphire is now in use by a fairly large community at PERQ Systems Corporation, Carnegie Mellon University, and elsewhere. It supports a full implementation of the covered window paradigm (where the rectangular windows can overlap like pieces of paper on a desk) which is described in a separate paper [Myers 84]. The covered window paradigm was first introduced in Smalltalk [Tesler 81] and DLisp [Teitelman 77]. In Sapphire, windows can cover each other and can extend off the screen in any direction (and may be entirely off screen).

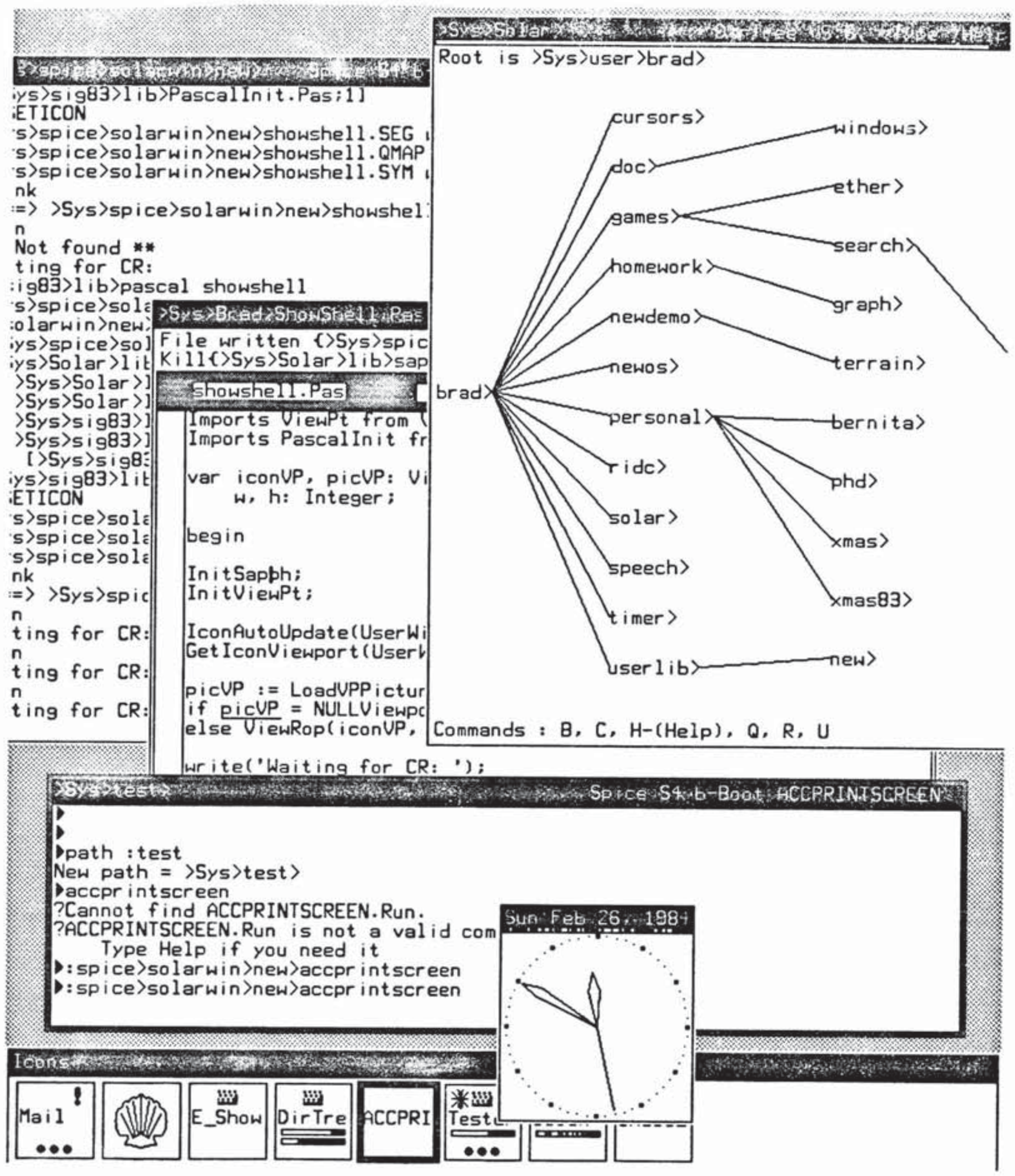
All window managers can be logically divided into three parts: the implementation of the low level graphics primitives (such as how the window manager prevents pictures in covered portions of windows from showing), the graphics the window manager presents to the user (such as window title lines and icons), and the operations that the window manager allows the user to perform to manipulate windows. The latter two parts of a window manager are collected under the heading *User Interface*. This paper is a summary of the user interface of Sapphire. It presents some aspects of the design of Sapphire and explains why they are interesting and novel. The implementation of Sapphire is covered in [Myers 84]. The reader is assumed to be familiar with window managers in general and how icons have been used in them.

Presentation of Windows. Windows in Sapphire typically have title lines and borders (see figure 1). Applications may create windows without either, but the title line is useful for displaying state information, and the border is useful for showing where the windows are. One window is the one that the user is typing to. In Sapphire, this is called the *Listener* window (since it is "listening" to the keyboard). The entire border of the Listener is highlighted in grey so the Listener is easily visible (figure 1).

Icons. The icons in Sapphire are very different from the icons in the Star [Smith 82] or Lisa [Williams 83]. In Sapphire, they were designed to enhance the user's productivity when he is doing multiple tasks concurrently. To accomplish this, the icons present 6 pieces of information about the process being run (see figure 1). First, there is the process name. Next, there are two "percent-done progress bars" [Myers P2]. Progress bars function like the giant thermometers used to mark progress in charity drives. They give the user enough information at a quick glance to estimate how much of the task is completed. These are kept up to date by the application program (or by the system for certain operations), so the user can always know how far along the process is. For example, the compiler reports how far it is through the file. The first progress bar is used by the application and is repeated in the title line of the window underneath the text. The second progress bar, which appears only in the icon, is used for aggregates, such as command files, to tell how much of the entire job is left. Most applications can calculate or estimate what percentage of the work is complete, but for those that cannot, Sapphire provides "random progress" that shows that work is progressing by continually XORing vertical lines at random places along the progress bar.

The last three pieces of process information are shown as pictures in the icon. One picture shows when the process is waiting for user input, another tells whether there has been an error in the last activity, and the third is reserved for specific application-defined attention signals. For example, this signal might be used by a mail program to report that new mail has arrived. The pictures are present when the condition is true and absent when it is false. The actual pictures used are a keyboard, a bug, and an exclamation point (figure 1), but these can be easily changed by the user or application.

Icons also contain two pieces of information about the state of the window. First, the border of the icon for the Listener is highlighted in the same manner as the window itself. Second, the icon displays a small picture if the window has been moved entirely off the screen.



```
ys>sig83>lib>PascalInit.Pas;1]
ETICON
s>spice>solarwin>new>showshell.SEG
s>spice>solarwin>new>showshell.QMAP
s>spice>solarwin>new>showshell.SYM
nk
=> >Sys>spice>solarwin>new>showshel
n
Not found **
ting for CR:
ig83>lib>pascal showshell
s>spice>sola
olarwin>new
ys>spice>sol
ys>Solar>lib
>Sys>Solar>l
>Sys>Solar>l
>Sys>sig83>l
>Sys>sig83>l
 [>Sys>sig83
ys>sig83>lib
ETICON
s>spice>sola
s>spice>sola
s>spice>sola
nk
=> >Sys>spic
n
ting for CR:
n
ting for CR:
n
ting for CR:
```

```
>Sys>Brad>ShowShell.Pas
File written {>Sys>spic
Kill{>Sys>Solar>lib>sap
showshell.Pas
Imports ViewPt from (
Imports PascalInit fr

var iconVP, picVP: Vi
     w, h: Integer;

begin

InitSapph;
InitViewPt;

IconAutoUpdate(UserWi
GetIconViewport(UserW

picVP := LoadVPPictur
if picVP = NULLViewpc
else ViewRop(iconVP,

write('Waiting for CR: ');
```

```
Root is >Sys>user>brad>
Commands : B, C, H-(Help), Q, R, U
```

```
>Sys>test>                         Spice S4-b-Boot ACCPRINTSCREEN
▸
▸
▸path :test
New path = >Sys>test>
▸accprintscreen
?Cannot find ACCPRINTSCREEN.Run.
?ACCPRINTSCREEN.Run is not a valid com
    Type Help if you need it
▸:spice>solarwin>new>accprintscreen
▸:spice>solarwin>new>accprintscreen
```

Figure 1. Sample Sapphire screen.

This is an actual picture taken from a PERQ running Sapphire. (The title line text in this figure may be difficult to read due to properties of the printer.) The covered window at the upper left, which is running the Pascal compiler, is partially off the screen to the left, and the window with the directory tree is partially off the screen to the right. Other windows are running the editor, a clock program and the screen print program. This latter window has a grey border to show that it is the Listener. The icon window at the bottom contains icons for all these windows and some that are off the screen. The first, for the Mail program, has the attention signal displayed to tell us that there is mail, and the three dots show that the window is off-screen. The next icon has an application-defined picture. Sapphire allows applications to define arbitrary pictures for icons, although the standard information will, in general, be more useful. The next icon is for the editor window and shows some of the name of the file being edited (editing "Show..."). The editor is waiting for the user to type something (shown by the keyboard). DirTre is also waiting for input. It is about 80% done displaying the tree (top progress bar, which is repeated in the window title line) and it is part of a command file that is about 20% done (bottom progress bar). The next icon corresponds to the Listener window. The next Icon shows that the Tester program has run into an error (the bug), it is waiting for the user to do something, it is 60% done and the window is off-screen. The next icon is for the clock and shows random progress. This progress is repeated in the clock's title line. Note that the icon window can be covered like any other window.

Although the icons are small (64 by 64 pixels each) they are easy to interpret, and all of the state information is displayed in a convenient manner so the user can simply scan the icons to deduce the state of his entire computer system and easily decide which processes require attention. Of course, if this default icon information is not appropriate for some processes, they can simply define their own pictures to be displayed in the icon. In this way, the icons of the Star or Lisa can be easily simulated by Sapphire.

In almost all other window managers with icons, an icon appears when a window is removed from the screen. Thus the window is either displayed *or* it has been shrunk down so that only the icon shows. In Sapphire, however, it seems clear that we want to see icons for all windows since they provide so much useful information. There is no logical difference between a window that is totally covered by other windows and one that is off screen--they are both invisible to the user. Therefore, icons in Sapphire are visible for all windows. A window may still be removed from the screen in Sapphire by simply moving it so that it is totally off screen (see below). The icons are therefore *associated* with a window rather than being an alternative representation for it.

Another important difference between icons in Sapphire and in other systems is that Sapphire's icons are all grouped together in one window. This allows the icons as a group to be moved around or removed altogether if the user does not like icons. The icon window can be covered by other windows and its size and position can be changed in the same way as any other window.

The icons in the icon window are not rearranged except on user command. Thus, when a window is deleted, its icon is deleted but the hole is not filled until another window is created. Users will probably remember which window goes with which icon by position*, so it is important not to rearrange the icons without the user's permission.

*Of course, there are also commands to identify the icon for a window and the window for the icon if the user forgets.

User Interface. Sapphire, unlike most other window managers, does not reserve ***any*** of the buttons on the pointing device exclusively for use by Sapphire. Any button pressed or released inside the Listener window is sent to the application running in that window. Thus application programs, such as editors, are free to use all the buttons to make ***their*** user interfaces more powerful. To give window manager commands for a window, the user must press a button in the title line of the window or in the icon for that window. To change the Listener in Sapphire, the user must make an explicit action by pressing with a button in a window; just moving the tracking symbol is not enough.

Sapphire provides all window manager commands from pop-up menus (which appear when requested and disappear when a command is issued), but these are too slow for frequent use by experts. Therefore, a single button press is sufficient to give the most common commands. These include top, bottom, moving a window, and changing its size. Another operation provided is making a window full-screen. This might be used, for example, when more contextual information is desired during an editing session. The user can define whether "full-screen" leaves the icons visible or not. When a window is made full-screen, its old position and size are saved so the window can be returned to its original place. Similarly, there is a command to move the window entirely off the screen. This can be used to prevent the screen from getting cluttered with windows that are not in use. A command using the associated icon brings the window back to its original place on the screen.

The title line of a window is divided horizontally into three sections: left, middle and right. The left and right parts have the same functions since windows are often covered on one side or the other. The most common pointing device for Sapphire has three buttons, so we have 3 *(buttons)* × 2 *(areas)* = 6 *(functions)* available on the title line. On the ends of the title line are: (1) top, (2) bottom, and (3) pop-up menu. The pop-up menu provides all the commands that can be given directly from the title line so that a novice can always find commands easily. The pop-up menu also includes the command "help" which generates a window in the center of the screen explaining all the commands.

In the center of the title line, the more esoteric functions are provided: (4) move/grow, (5) full-screen or back from full screen, and (6) off screen. When the user selects move/grow, he then selects a position on the window to move/grow from. The corners are "grow" points, and the sides have both "move" and "grow" points. Since windows in Sapphire may be moved partially off-screen in any direction, it is necessary to be able to move them from any side. It is also useful to be able to grow a window from different points if the user is trying to align one window with other windows. During the move and grow operations, and also during window creation, hair lines are displayed to show the outline for the window so the user can place it accurately.

The icons are too small to have three areas so we are limited to three functions total (one for each button). One button performs "top" and "listener" and "back from off-screen" all at once. Another button provides a pop-up menu with all of the commands, and the third button identifies the window that corresponds to the icon. This operation is accomplished by blinking the window and the icon, and drawing lines from the corners of the icon to the corners of the window. If the window is covered, the lines show where it is so the user can send the appropriate windows to the bottom for it to be visible*.

With all these different functions, it seems clear that the user will not always remember which button will get which action. In Sapphire, we provide a simple method for showing the operation that will occur that does penalize experts. When a button is pressed, the tracking symbol changes to a picture that shows the operation that will be performed (see figure 2). If this is the desired operation, then the button is simply released. If this is not the desired operation, then the user can move the tracking symbol around to the correct place (if in the title line) or move it away to abort before releasing. Thus the expert can simply press and release without waiting for the picture, and the novice can check all operations before executing them. While performing multi-step tasks like growing a window or choosing from a pop-up menu, Sapphire also displays an appropriate tracking symbol picture. This can be used, for example, to verify whether a "move" or a "grow" will be performed. Also, the user can always abort these operations by hitting a keyboard key, so he is never stuck in a mode that he does not know how to leave.

*Of course, the user can simply bring the window to the top using the first icon command.

For some reason, there are a number of people who prefer not to use a pointing device. Therefore, all commands are also available from the keyboard in Sapphire. Since there are a number of commands, we use a special prefix key (that does not have a standard ASCII interpretation). This avoids the problem of taking many characters away from application programs. The keyboard commands are also useful if the pointing device is broken, or if some process has reserved it or changed the tracking in arbitrary ways. The prefix key returns the tracking to the standard default so that users can always give commands using either the keyboard or the pointing device.

All of the operations available from the pointing device or pop-up menu have corresponding keyboard commands. In addition, one keyboard command changes the Listener to the next window in a certain order, and the other keyboard commands typically operate on the Listener. One of the keys on the PERQ keyboard is labeled "HELP" and this provides help for Sapphire if typed after the prefix key. Finally, there is a special keyboard command to find the icon window if it has been moved off screen or covered.

2. Conclusions

Sapphire incorporates a number of different innovations in window management and user interface design. Icons in Sapphire are used in a novel way to enhance user

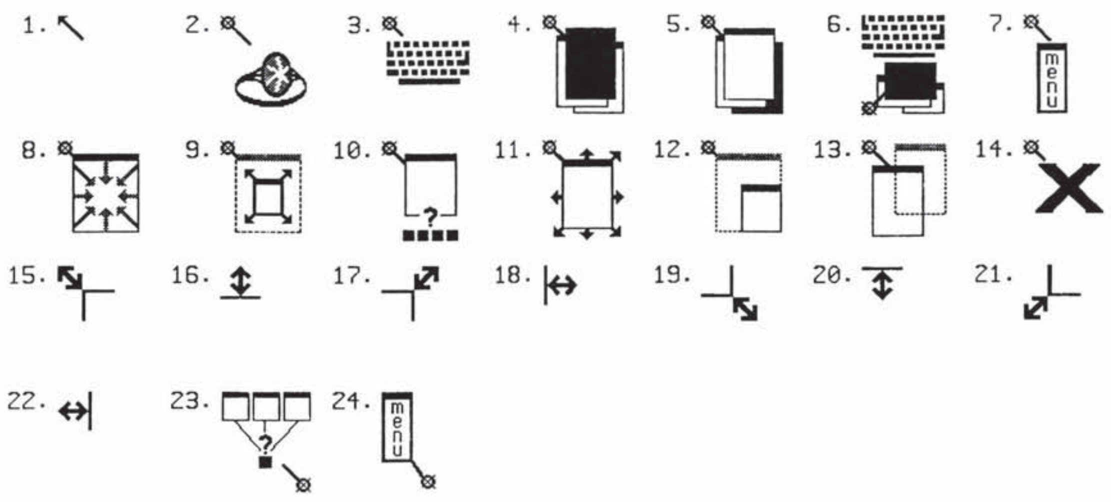


Figure 2. Tracking Symbols used in Sapphire.

When the user presses down on the pointing device, the tracking symbol picture changes to show what operation will be performed. If the operation requires multiple actions (for example, when growing a window), the picture changes at each step to show what is expected next. The pictures above represent:

1. The system default cursor.
2. The default Sapphire cursor (a star Sapphire ring).
3. Make this window be the Listener.
4. Bring this window to the top.
5. Send this window to the bottom.
6. Top and Listener (used in the icon).
7. Get a pop-up menu of commands.
8. Return the window from full screen to original position.
9. Make the window full-screen.
10. Identify the icon for this window.
11. Change size or position of window from any side or corner.
12. Change the window's size.
13. Change the window's position.
14. User-specified abort of any operation.

15-22. Specify corner or side of the window.

23. Identify the window for this icon.
24. Get a pop-up menu from icon.

productivity when multi-tasking. They allow the user to more easily monitor and control multiple processes and windows since they contain eight different pieces of process and window state information. The user interface of Sapphire is easy for the novice while providing simple and powerful operations for experts. The user interface promotes experimentation since there is always appropriate feedback in the tracking symbol to tell which operations will be performed, and it is always possible to abort an operation once it has been started.

Sapphire is currently in use by a growing community of people in different fields. It is continually being modified based on user feedback as we discover how to make it even easier to use.

ACKNOWLEDGEMENTS

The design of Sapphire grew out of discussions with Gene Ball, Fred Hansen, Dave Golub and many others at CMU and PERQ Systems Corporation. Dave Golub and Amy Butler have been largely responsible for maintaining Sapphire. For help and support with this paper, I would like to thank my wife, Bernita Myers, and Alain Fournier, William Buxton, and Brian Rosen.

References

[Myers 84] Brad A. Myers. "A Complete Implementation of Covered Windows for a Heterogeneous Environment," Dynamic Graphics Project Technical Memo. University of Toronto Computer Science Department, 1984.

[Myers P2] Brad A. Myers. "The Importance of Percent-Done Progress Bars for Computer-User Interfaces". In preparation.

[Rosen 80] Brian Rosen. "PERQ: A Commercially Available Personal Scientific Computer." *IEEE CompCon Digest*, Spring, 1980.

[Smith 82] David Canfield Smith, Charles Irby, Ralph Kimball, Bill Verplank, and Erik Harslem. "Designing the Star User Interface." *Byte Magazine*, April 1982, pp. 242-282.

[Teitelman 77] Warren Teitelman. *A Display Oriented Programmer's Assistant*. Palo Alto: Xerox PARC CSL-77-3. March 8, 1977. 30 pages.

[Tesler 81] Larry Tesler. "The Smalltalk Environment," *Byte Magazine*, August 1981, pp. 90-147.

[Williams 83] Gregg Williams. "The Lisa Computer System," *Byte Magazine*, February 1983, pp. 33-50.