CHAPTER 2

Working in the OPEN LOOK Environment

The only way to learn any complex task is by doing it. In the first part of this chapter we show you how to start up an X Window System session, including the X server and at least one terminal window, along with the *olwm* window manager. The chapter then guides you through some operations that are common to all programs using the OPEN LOOK GUI, as well as to most X11 programs. This chapter describes how to:

- Start your X system, if it isn't started automatically
- Use the mouse or other pointer device to choose a window
- Use the pointer to move, reshape, iconify, and even delete windows
- Understand and use all the standard OPEN LOOK controls
- Start additional client programs, on both local and remote machines

This chapter is tutorial, and can be worked through at a computer with a version of the OPEN LOOK GUI software installed.

2.1 Getting Started with X and OPEN LOOK

There are several possible ways you can start X. If you are lucky, your system administrator will have read Volume Eight, *X Window System Administrator's Guide*, and configured your system to use the X Display Manager (*xdm*) login program as shown in Figure 2-1. If this is the case, you need only type your normal login and password to begin using X.

Or, you may have available a comprehensive vendor-supplied X package such as Sun's OpenWindows, in which case you need only log in at the workstation console (see Figure 2-2), and type one command, normally *openwin*.

On the other hand, you may be using a dedicated terminal called an X Terminal. Each of these methods is described in the following sections. If you don't fit into one of these cat-

Figure 2-1. An XDM Screen

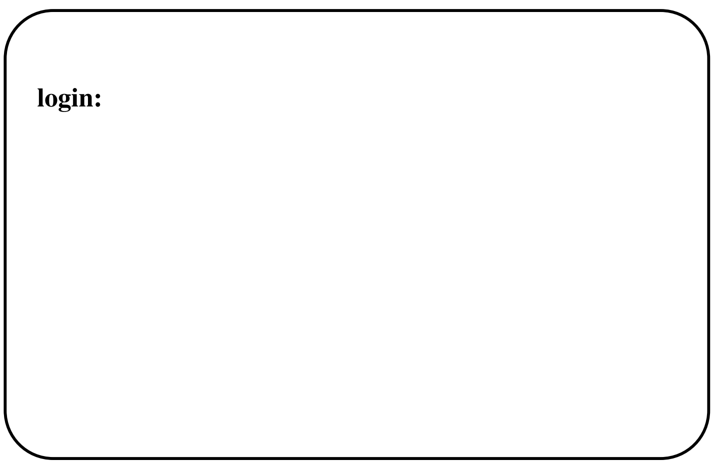

Figure 2-2. A Console Login

egories, you are probably on your own, and should consider reading Volume Eight, *X Window System Administrator's Guide*.

2.1.1 Logging In via the Special xdm Window

If the *xdm* display manager is running X on your system, you'll probably see a window similar to Figure 2-1 when you start up (or reset) your terminal.

Log in by typing your user name and password at the prompt, pressing RETURN after each. Without any user customization, the display manager executes a standard login "session," providing the first *xterm* window and starting the window manager. The *xterm* window will be displayed in the upper-left corner of the screen. If the OPEN LOOK window manager is running, you will see the characteristic frame around the *xterm* window, as in Figure 2-3. The name of the application ("xterm") appears in the frame's title area.

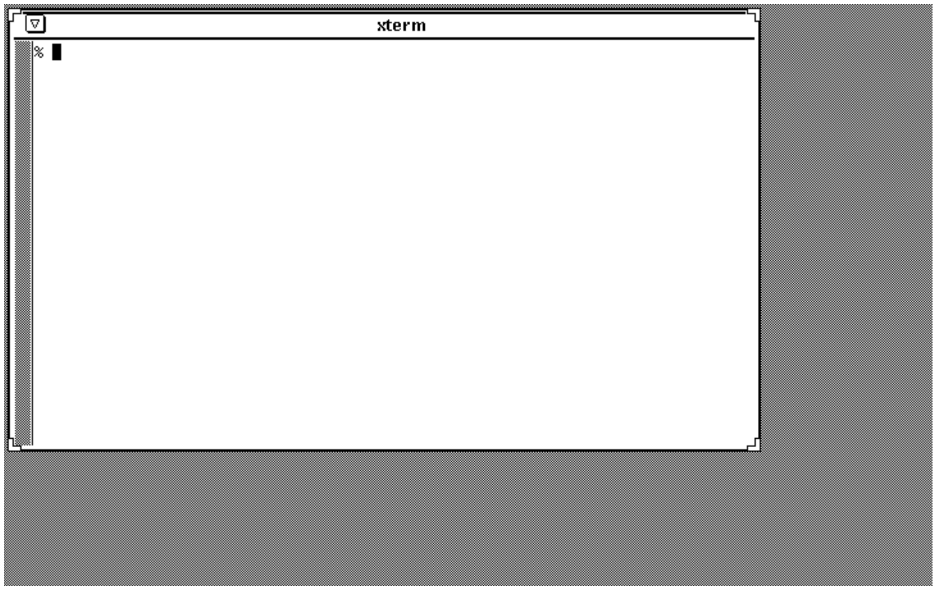

Figure 2-3. Window frame indicates that olwm is running

The frame provides a quick and easy way to move, iconify, resize, and otherwise manage windows on the screen. Some of the basic window manager functions are described below and in Chapter 6, *Using the OPEN LOOK Window Manager*.

You can now proceed to the section "Selecting and Using a Window" on page 35.

2.1.2 Logging In at a Full Screen Prompt: Starting OpenWindows or X

If you want to use a workstation that is not displaying any windows, but instead has a simple prompt ending with *"login:*", you just need to log in and start X manually. Typically, when you log in, no windows are opened; instead the entire screen functions as a single terminal, as shown in Figure 2-2.

If your account was created with the Sun-provided6 startup scripts (for example using SunOS 4.1.x's *adduser* script), then you will get this message

```
Starting OpenWindows, Control/C to interrupt...
```

and OpenWindows will be started for you automatically. If not, maybe your system administrator will be persuaded to configure *xdm* for you...

If no windows are displayed when you log in at a full screen prompt, you can start the X Window System manually. In this section we'll assume that an OPEN LOOK package such as Sun's OpenWindows has been installed. You need only type the command

```
% openwin
```

and you should (after a delay) see the full screen change color. Then an OpenWindows logo similar to Figure 2-4 should appear. If instead, you get a message like "command not found", then you may need to specify the full UNIX path to the start-up script, which is usually

```
% /usr/openwin/bin/openwin
```

The resulting screen should look like Figure 2-4.

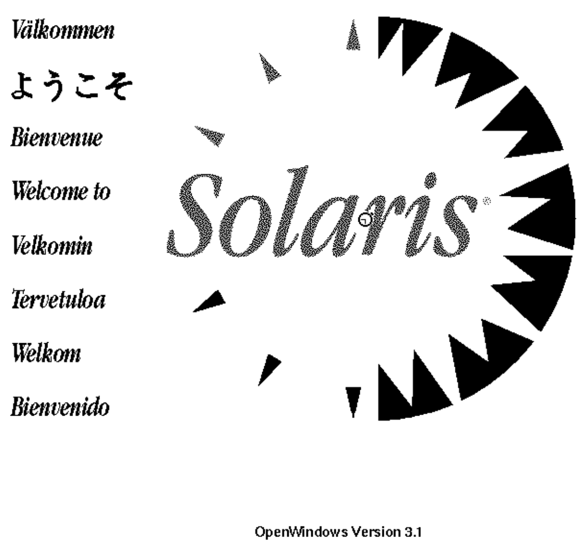

Figure 2-4. Sun OpenWindows startup screen

If you have an older version of OpenWindows then the welcome screen will be simpler, with a logo like that of Figure 2-5 in the center of the screen.

After some time (depending on your workstation), the screen will clear again, and you will get some default clients like those in Figure 2-6.

You can use the optional `-dev` argument to specify a non-default "frame buffer" device—the controller for a graphics monitor connected to your workstation (normally the

Figure 2-5. Older OpenWindows startup screen

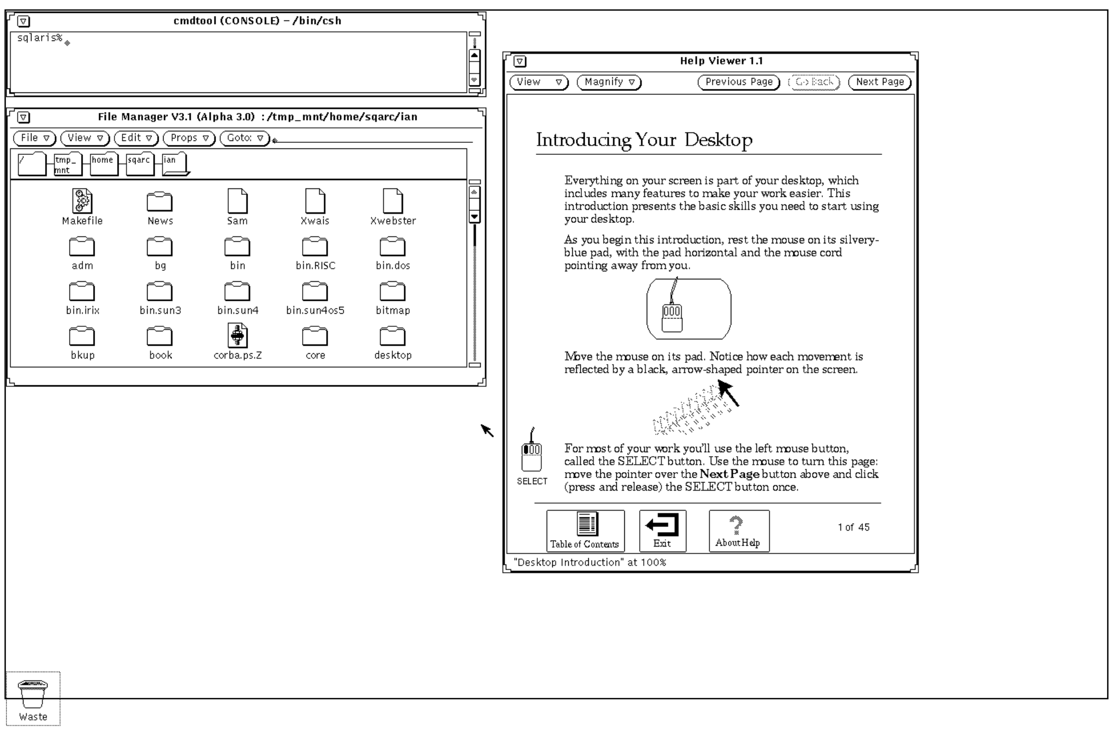

Figure 2-6. OpenWindows initial screen—not customized

console). For "cgfour" systems (3/60, 3/110, and 4/110 with the normal color monitor), you can use this to specify color-only, monochrome-only, or to use both by a kind of "virtual screen" arrangement. The recommended configuration for these systems is to use:

```
% openwin -dev /dev/cgfour0 -dev /dev/bwtwo
```

In this configuration, the color screen (the default screen, or screen zero) is "to the left of" the monochrome screen (screen one); that is, moving the mouse off the right side of the color screen moves you into the black-and-white screen (makes the monochrome screen appear on the monitor); moving off the left of the monochrome screen brings back the color screen. For more details, see the *openwin* and *xnews* manual pages in Part Three of this Guide.

You can now proceed to the section "Selecting and Using a Window" on page 35.

If, on Solaris 1 (SunOS 4.1.x), you get a message like

```
wiwindow: Base frame not passed parent window in environment"
Cannot create base frame. Process aborted.
```

then you have erroneously set your PATH search variable and are picking up the SunView version of some programs (see Appendix L, *Running SunView Applications on OpenWindows*). Ensure that the directory */usr/openwin/bin* is ahead of */usr/bin* and all will be well.

2.1.3 Starting on a 386 with SVR4

A good example of a flexible system is Dell Computer Corp's System V Release Four with X11R5. Merely by setting one line in a system file, you can customize the appearance of the overall system to Sun's OPEN LOOK, AT&T-OL, Motif, or the MIT look. Copy the file */usr/XR5/lib/xinit/xinitrc* to your home directory under the name *.xinitrc*, and edit it. Remove the comment character ("#") from before the line `WM=xvolwm`, and insert the comment character before the line `WM=mwm`, and you will get a Sun look, including the *cmdtool* terminal emulator, that is closest to most of the examples in this book. For an AT&T-OL look, including the OPEN LOOK version of *xterm*, un-comment the line `WM=olwsm.` With `WM=xvolwm`, the screen should look something like Figure 2-7.:

You can now proceed to the section "Selecting and Using a Window" on page 35.

2.1.4 Starting with an X Terminal

Most X terminals now use *xdm*, and so behave as described in the section "Logging In via the Special xdm Window". If you aren't running *xdm*, check your vendor's documentation to see if your X terminal supports the *xdm* protocol; if so, see that documentation or Volume Eight, *X Window System Administrator's Guide*, to configure it. If you have an older X terminal that isn't *xdm*-capable, you normally use the X terminal's built-in *telnet* or *rlogin* session capability to connect to the file server on which the clients reside. Then you can invoke your *.xinitrc* script either directly or through a simple shell script. Here is one example of a script, *xterminal,* that the author has used in the past. The key point is to note that we want to start all the normal clients, but we do **not** want to try to start a server, because the X terminal itself takes care of starting its own server when you turn it on.

```
#!/bin/sh
DISPLAY=${1-myxterm:0};export DISPLAY
OPENWINHOME=/usr/openwin; export OPENWINHOME
LD_LIBRARY_PATH=$OPENWINHOME/lib;export LD_LIBRARY_PATH
            PATH=$PATH:/usr/openwin/bin; export PATH
exec /bin/sh .xinitrc
```

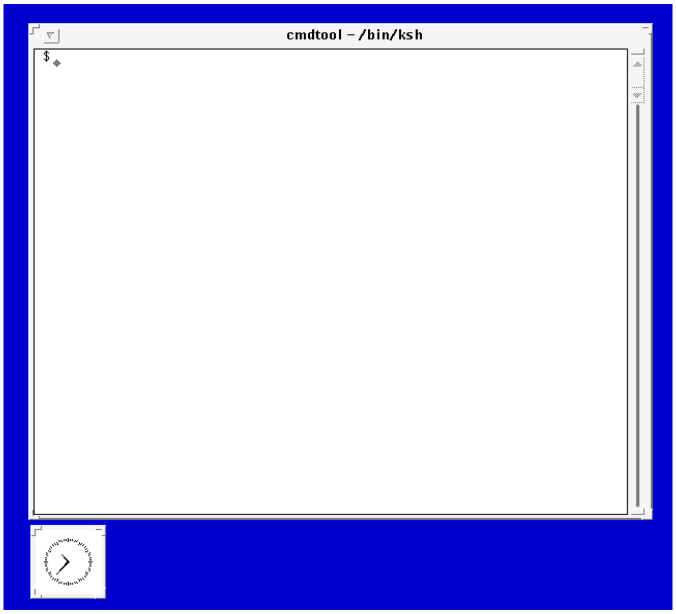

Figure 2-7. Initial screen with WM=xvolwm

In this code, `myxterm` is a default X terminal's TCP/IP hostname—the X terminal that I normally use. For more information on the DISPLAY variable, see Section 3.2.2, "Running a Client on Another Machine: Specifying the Display" on Page 61. The LD_LIBRARY_PATH and PATH settings are needed if you are using Sun OpenWindows clients (normally the *openwin* command sets this, but since we are invoking the clients without using that script, we have to set these explicitly; see also Section 3.2.5, "Complications: LD_LIBRARY_PATH). The last line starts up our *.xinitrc* script; the *exec* keyword is there for efficiency.

2.2 Selecting and Using a Window

Now that you have a terminal emulator and a file manager handy, you can begin using them. To use the terminal emulator, move the cursor into its titlebar and click the SELECT (left) pointer button anywhere except on the Menu Button at the left.†

The titlebar becomes highlighted to show that the window has been given the focus. We describe this "click-to-type" (or explicit focus) policy in Chapter One. (If *olwm* failed to start, or fails during operation, such that you have no window manager, you use the "pointer-focus" policy; simply move the pointer into a window and start typing.) If you start typing when no window has the focus, your keystrokes will be lost without warning.

† In fact, you can click anywhere in the window, but it's safest to click on the titlebar while you're learning the system.

Once you have the focus on a terminal emulator window, it should behave much like an old style ASCII terminal. The details vary depending on which terminal emulator you run and how you have configured it, of course.

From here on, we will be discussing the details of using the previously mentioned clients. The rest of this chapter discusses some specifics of using the OPEN LOOK GUI. The window manager, the file manager, and the terminal emulators will get our attention in the next few chapters.

There are two areas of any OPEN LOOK window that are used to provide feedback: the titlebar, and the footer. The titlebar is "grayed out" to indicate that the window will be busy for some time. And the footer is used to display messages. Watch for messages in the footer of any application using the OPEN LOOK GUI!

2.3 Creating Additional Windows

You can create additional windows whenever you need them. First we'll show you how to do so from within a shell window, then we'll see how you can create some windows from the system menus.

Move the input focus to the terminal emulator (*xterm* or *shelltool*) now. Since you have a UNIX shell running, you can run any program. One such program that you may need to run often is a second (or third...) copy of the same terminal emulator. Just type *shelltool* or *xterm*, an ampersand ("&"), and press the RETURN or ENTER key. This will create a new window, with a new copy of the UNIX shell in it (Bourne, Korn or C, whichever you normally use). It may take a few seconds, depending on how fast your system is. This window will have the default size for the program, typically 24 lines of 80 columns each for *xterm*, or 40 x 80 for *shelltool/cmdtool*.

By default, OPEN LOOK window managers will place new windows in a stack starting at the upper-left corner of the screen, and moving down and to the right a little each time you create a new window. We'll see later in this chapter how to specify the positioning of clients when you create them; this will apply to all clients, not just terminal windows. Note that a new window does not automatically get the input focus. With OPEN LOOK's default click-to-focus policy, you must click SELECT on a new terminal window before you can start using it. As well, anytime a window is partially hidden by another window in front of it, the SELECT operation that gives a window focus will also bring that window to the front, but only if you clicked on the window's titlebar. If instead you click inside the window, the system will give that window the input focus, but will *not* raise the window to the front. There are operations for explicitly moving a window to the back or front of other windows, as well as moving a window to a particular location, and others. We'll look at some of these shortly. But first, we need to learn some more details on using the pointer.

2.4 Using the Pointer

Most mice used with X11 have three buttons. OPEN LOOK defines their use as in the list that follows; this is similar to (but not identical to) the commonly accepted uses in most X

Window System applications. Note in particular that the *xterm* terminal emulator is *not* OPEN LOOK-compliant, so it uses some buttons differently. See Appendix A, *The xterm/olterm Terminal Emulator* for details.

- SELECT—the left button, used to select an object or text
- ADJUST—the center button, used to shorten or lengthen the selection
- MENU—the right button, used to access a Menu.

But what if you don't have a three-button mouse? The designers of OPEN LOOK anticipated this, and assigned these standard alternatives:

- On a two button mouse: Left is SELECT, shift-left is ADJUST, right is MENU.
- On a one button mouse: button is SELECT, shift-button is ADJUST, control-button is MENU.

2.4.1 Selecting Text

When using OPEN LOOK, you normally select something before you operate on it (this is called a "a selection-operation model"). For example, if you wanted to copy a sentence of text from one window to another, or to cut out a line of text from a window-based text editor, you would first SELECT the text, and then perform the cut or copy operation. The most common way of selecting a range of text is as follows:

1. Move the cursor to just before the start of the text
2. Click and hold the SELECT mouse button
3. Move or *wipe* the cursor across the text you wish to select (as you do, each character will be highlighted in inverse video)
4. Release the SELECT button

For example, let us assume that we have some text in a window, as shown in Figure 2-8.

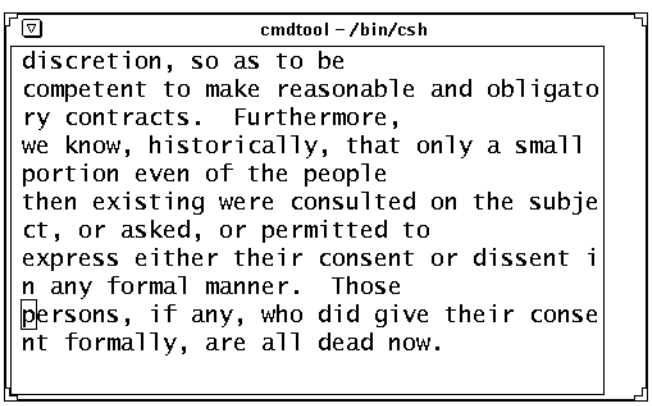

Figure 2-8. Text before selecting

Our goal is to select the words "only a small portion even of the people then existing were consulted." Move the pointer to just before the "o" of "only", then click and hold the SELECT button while you move the pointer to just after the "d" in the word "consulted". The screen should now look like Figure 2-9.

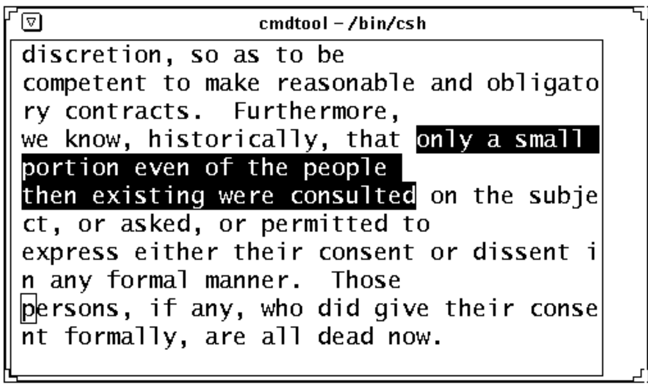

Figure 2-9. Text after selection

If you need to lengthen or shorten the selection *after* you've released the button, no problem. Just move the pointer to the new desired end point, and click ADJUST.†

An alternate method of selecting a range of text is to click and release SELECT at the start of the text you want selected, then move to the end and click ADJUST.

Then you can use the MENU button, or move to another window, to do the cut or paste operation. This will be described, with examples, in Chapter 5, *The cmdtool/shelltool Terminal Emulator*, where we will also discuss other types of text selections. For now what matters is that text can be selected for use in copying, cutting, etc. Other objects can be selected, too: think of setting the input focus by SELECTing a particular window. You select the window and then do something to it (type into it, move it, etc.). In short, SELECTing text or graphical objects is a basic operation.

2.4.2 Menu Choices

Like most window systems, OPEN LOOK makes extensive use of on-screen "menus," or lists of related choices. OPEN LOOK programs generally don't have permanent menus, ones that are always present on the screen. Instead, they use pop-up menus. And in addition

† If you have previously used the MIT version of the *xterm* terminal emulator, you will know that *xterm* uses "button1" to start a selection, and "button3" to adjust it. But in OPEN LOOK, the ADJUST button is normally the middle button, "button2." There is no easy way to reconcile these incompatibilities, though individual users may overcome them with an X Toolkit mechanism called "translations" (see Appendix F, *X Toolkit Translation Table Syntax*). OpenWindows users may find it easier to use *cmdtool* instead of *xterm* while you are learning this material, since *cmdtool* conforms to OPEN LOOK instead of the *xterm* conventions. AT&T-OL, however, includes a version of *xterm* that *does* conform to the OPEN LOOK specification.

they have one of OPEN LOOK's significant contributions to window systems: "pinnable" menus, so that you the user can decide which menus are to stay up.

To use a menu, you must first display it with the MENU button (usually the right button). There are two ways to display OPEN LOOK menus, the traditional "pop-up sliding menu" style, and the "stay-up" style.

- Pushing the MENU button *and holding it down* "pops up" the menu—it stays on the screen as long as you hold the MENU button down while you decide what to do. You can move the pointer to any of the menu items and release it; this will cause the corresponding action to be invoked. As you slide the pointer up and down, the item under the pointer is highlighted, to indicate that it will take effect if you release the button. Figure 2-10 shows one such menu: this example is the normal "titlebar" menu for windows, or *Window Menu,* both as it first appears and after you move the pointer to the right to select `Close`.

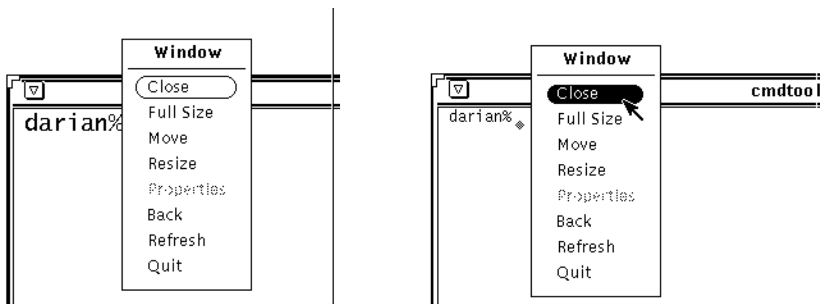

Figure 2-10. Window Menu

- On the other hand, if you push the MENU button *and quickly release it*, then the menu will appear in "stay-up" mode, that is, it will stay up just until you make one selection. You can click the SELECT pointer button on one of the menu choices to select that choice. If you want the menu to stay up for a while so you can invoke several of the actions, you can click the SELECT pointer button on the pin, which causes the menu to become pinned as described below†. Or, you can click any pointer button *anywhere else on the screen* (typically in the *Workspace* or background) and the menu will be dismissed.

Important menus, such as the "root menu" or *Workspace* menu, are *pinnable*, which means you can "pin them up." Some *windows* are pinnable too, as we'll see later. If an object is "pinnable", you can "pin" it, or make it stay up, just by moving the pointer over the picture of the pin and releasing the MENU button. You'll see the pin move before you release the

† Some implementations, such as Sun's *olwm*, allow you to click *any* pointer button to select a menu choice or pin the menu. Others do not, though, so you should consistently use the SELECT mouse button for these operations.

button, to show you that the pointer is in the right place. Figure 2-11 shows a pinnable menu before and after pinning.

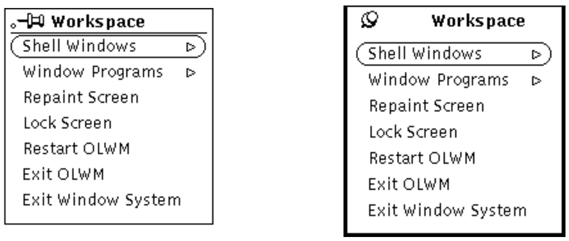

Figure 2-11. Pinnable menu before and after pinning.

The name comes by analogy with pinning up a note on your corkboard, or putting a Post-It note on your wall. Then the menu stays up with the pin in. If you later want to "dismiss" (get rid of) this menu, you can click the SELECT button on the pin, to pull the pin out and make the menu disappear. This discussion of pinning applies to any other item, such as a dialog box, that is pinnable.

2.4.3 Pushing Buttons and Menu Buttons

There are two types of buttons provided by the OPEN LOOK GUI, command buttons and menu buttons. Command buttons are the simplest; they look like this:

The function of a button will naturally be determined by the application. This particular button is from the *snapshot* program described in Chapter 9, *Graphics Clients*, and causes a "view" of the current image to appear. To invoke it, you simply move the pointer so that the point of the arrow is anywhere inside the oval, and click the SELECT pointer button More of OPEN LOOK's visual feedback comes into play when you push on any button: the button is grayed out until the operation it invokes is finished.

Note that some buttons have a "menu mark" consisting of three dots (" . . .") after the text. This indicates that when you select this button, a pop-up window (or "dialog box") will appear. For example, the "Print..." button is used to pop up a dialog that lets you specify which printer to use, number of copies, etc.

Many OPEN LOOK applications have buttons labelled *File, View, Edit*, and the like. These *menu buttons* have a pull-down arrow (▽) which means there is a pull-down menu attached to them, as shown below:†

† The *File* menu is normally used to open a new file, save your work in a file, etc. The *View* menu normally controls some aspect(s) of how the window is displayed. The *Edit* menu normally allows you to edit one or more aspects of the window's behavior, or edit multiple parts of the data. Watch for examples of these as we go, particularly in Chapter 7, *The OpenWindows DeskSet Clients*.

Clicking MENU on this button will pop the menu up, with all the flexibility discussed previously for menus. When you pull the menu down, you can that the item surrounded by an oval is the default. Buttons that do not have the pull-down indicator shown above are simpler—you just click on them with the SELECT pointer button when you want to activate them. The MENU button does not do anything on such buttons.

On the other hand, clicking the SELECT pointer button on it will select the default action, without having to view the menu. This is a convenient shortcut that you use after you're familiar with the button layout of a given application. But because you may not be sure, Menu Buttons also have a feature called "menu previewing." When you push and hold the SELECT button, they preview (display) the first few characters of the default item. If you then release the pointer button, the default action will be taken. If you move the pointer off that button, the default will not be taken. This is useful for applications that change their default (say, between *Load* and *Save* on their *File* menu button), or at any time you're confident but not certain about what the default action on a Menu Button will be.

2.4.4 Other Controls

In the following sections, we will discuss all of the common OPEN LOOK control items. Should you wish to experiment with these, run the "sampler" programs shipped with the OPEN LOOK GUI. These sample programs are aimed primarily at programmers who are developing applications using the widget set, and are similar to those offered by other widget sets, such as Motif. This section discusses the programs available with OpenWindows Version 3.0, but similar programs should be present with any implementation of the OPEN LOOK GUI. Look in *$OPENWINHOME/demo* to see what's available.

The first program, the OPEN LOOK Intrinsics Toolkit sampler, is in *$OPENWINHOME/demo/olitsampler* on OpenWindows. Figure 2-12 shows what it looked like on our system:

The second program is very similar to one provided with OSF/Motif. In both cases there is an attempt to provide a "periodic table" of the design elements, which parodies the original Periodic Table of the Elements used by chemists and physicists to classify all the basic chemical elements out of which all substances are built. In most such programs, all the pictures of movable controls, such as menus, scrollbars, and choices, are active, and can be used to experiment with the behavior of the controls. And, again for the benefit of the programmers in the crowd, the command button at the bottom of each small window pops up another window that reveals the source code needed to use that control. Figure 2-13 shows the OPEN LOOK Intrinsics Toolkit "periodic table":

Figure 2-13 is shown here in monochrome; to really appreciate how it looks, you should look at it in color, either on the page of Color Plates or by running it yourself on a system with a color monitor. Just give the command

```
/usr/openwin/demos/olittable &
```

and you should see a picture similar to the above. Try this now; we'll refer to some of the controls shown in the *olittable* program in the following discussion. You can ignore the two-letter names and numbers in the upper corners of each square; they are just there as a tongue-in-cheek reference to the original Periodic Table.

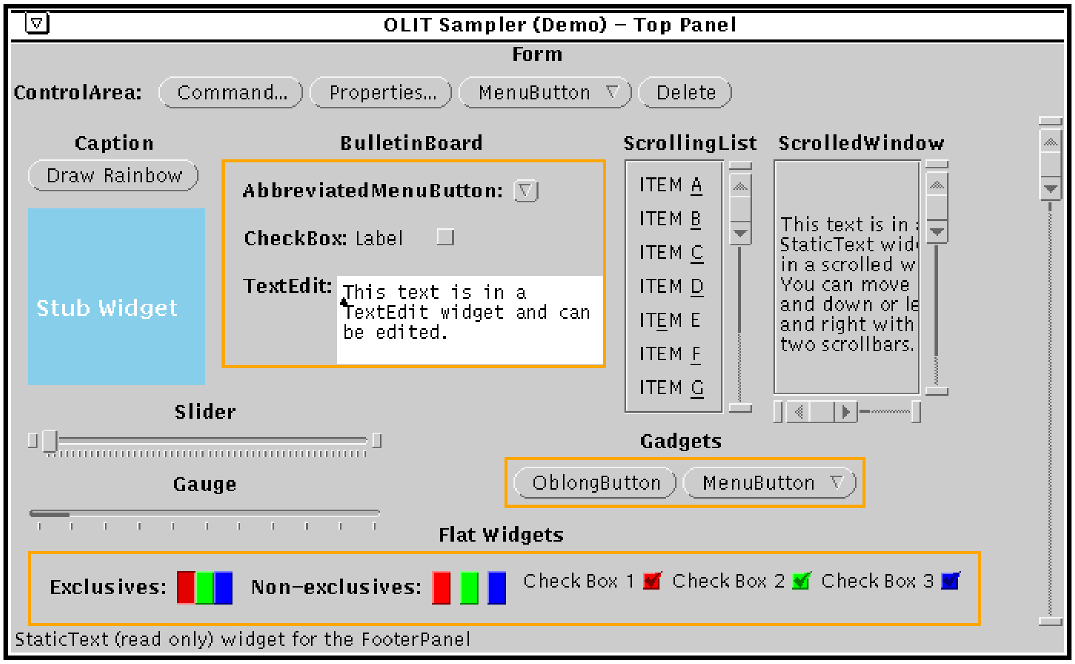

Figure 2-12. OLIT Sampler Program

2.4.5 Choice Items

Most window systems feature a variety of so-called "choice items", or controls that let you choose one or more options from a list. Here we look at the choice items provided by the OPEN LOOK GUI; exclusives and non-exclusives, checkboxes, and abbreviated choices.

2.4.5.1 Exclusives and Non-Exclusives

Exclusive and *Non-Exclusive* selections let you choose from a list of alternatives presented in small boxes. Each box is drawn with a darker outline (monochrome) or with a recessed image (three-dimensional look) when it is selected, and drawn with a thin border or drawn flat when not selected. The only significant difference between Exclusives and Non-Exclusives is that with the former, only one item can be selected, while with the latter, several or all of the selections can be active at once. Exclusive choices are drawn touching each other, to show that the buttons affect each other. For example, look at Figure 2-14, which shows more detail on the "Choices" section of *olittable*. The most common place to encounter these choice types is in "property sheets." A Property Sheet is a pop-up window that controls the "properties" or operation of a program. One example is the property sheet for *tapetool* shown in Chapter 7, *The OpenWindows DeskSet Clients*.

Each little rectangle, such as the one around *Pick Me* or *Bold*, is one "choice". The two sets of choices on the left are Non-Exclusives, while the two on the right are Exclusives. You can ignore the difference between the Flat and non-Flat versions of the choices; this is primarily for the benefit of application implementors.

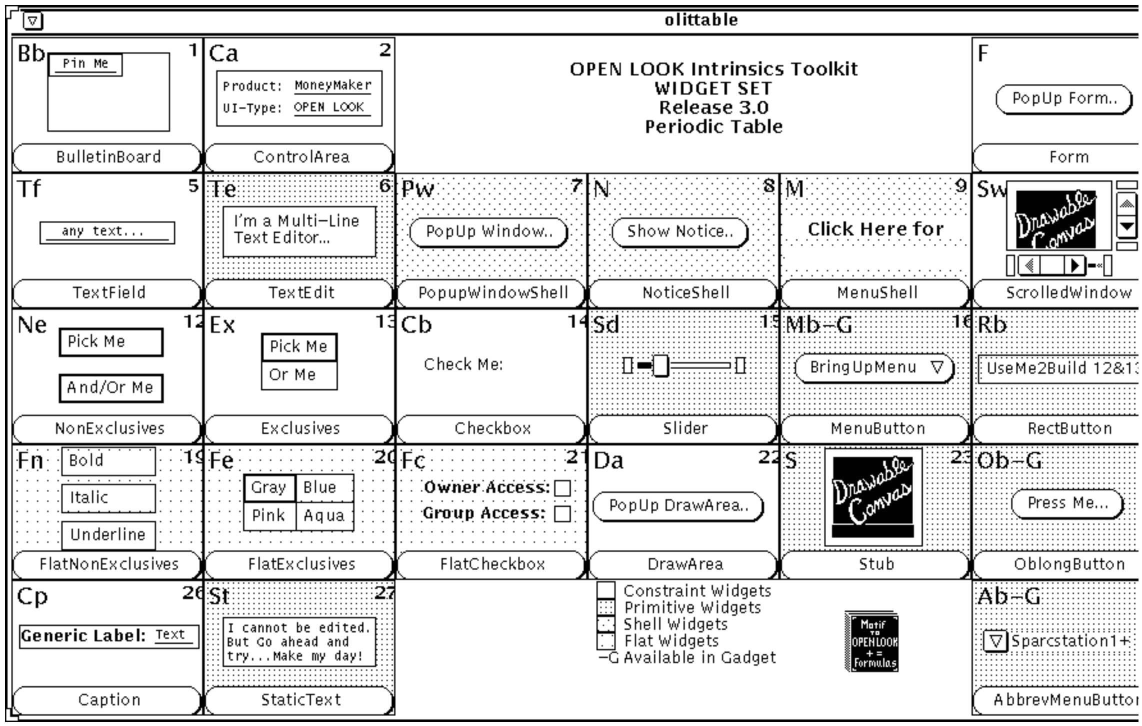

Figure 2-13. The OPEN LOOK "Periodic Table"

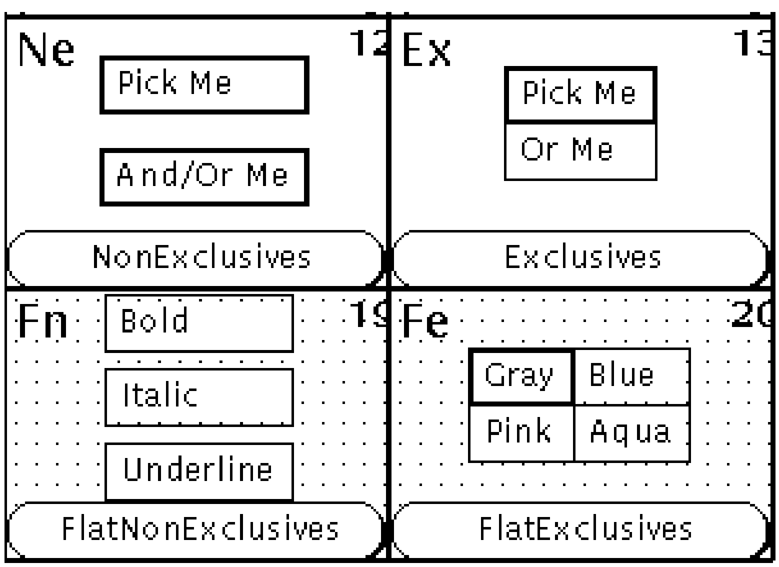

Figure 2-14. Choices section of olittable

Clicking SELECT on one of the Non-Exclusives will either select that item (if it is currently not selected), or deselected it (if it is currently selected). On the other hand, clicking SELECT on one of the Exclusives other than the one that is selected will select that choice and deselect the one that is currently selected.

For example, in a backup program, you might want to choose which tape format to use. It makes no sense to have both *ANSI Tar* and *CPIO* formats in effect at the same time, for example, since a tape can only be written in one format. So this decision would be an Exclusive choice. Figure 2-15 is a picture of such a (hypothetical) backup program:

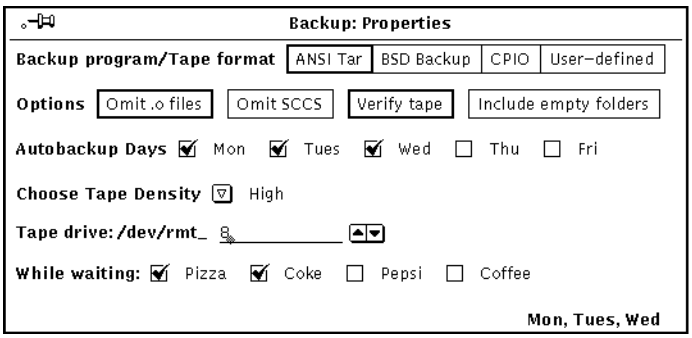

Figure 2-15. Choices in imaginary backup program

On the other hand, there are some options that are not mutually exclusive. Look at the Non-Exclusive settings in the Options row: it *does* make sense to turn on both "Omit .o files" (do not back up compiled "object" files) and "Omit SCCS" (do not back up SCCS archives), so either one, both, or any of the other options shown here can be turned on (or off again) by clicking SELECT in the box.

2.4.5.2 Checkbox

When you see a piece of paper with a list of items, some of which have little check marks beside them, you probably assume that those items with check marks have been done, or completed. The *Checkbox* control item is similar: use it to request that certain actions be done or that certain options be on or off. Figure 2-16 shows a close-up of the Checkbox item from *olittable*:

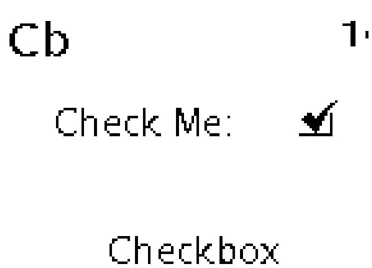

Figure 2-16. Checkbox from olittable

The Checkboxes are like little switches; if you SELECT them when they are on, they go off, and if you SELECT them when they are off, they go on. The third row of the program shown in Figure 2-15 allows you to turn backups on or off for each day of the week. The bottom row of this hypothetical backup program uses Checkboxes to let you order food and drink to consume while watching the tape drive spin.

2.4.5.3 Abbreviated Choice

The “Choose Tape Density” item is like an Exclusive choice, but it only displays the current setting. It otherwise functions like an abbreviated menu, discussed earlier in this chapter; if you click SELECT on the menu mark, the default is set, while if you click MENU, the menu pops up, as shown in Figure 2-17:

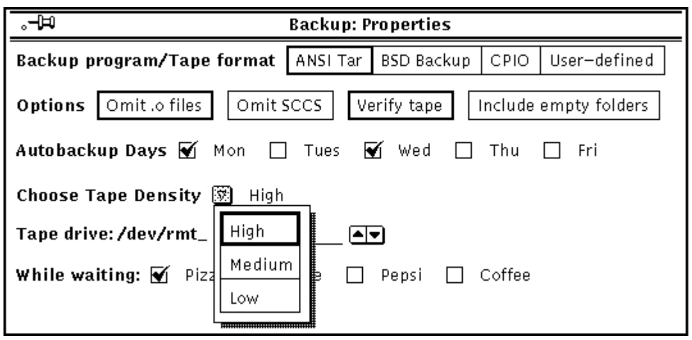

Figure 2-17. Abbreviated menu button pops up

Then you can click on any of the choices; the one you chose will be displayed beside the menu mark, and the menu will disappear.

2.4.6 Sliding items

There are several “sliding items” that let you scroll or adjust a value or a region. The *scrollbar, scrolling list* and *Gauge* controls are discussed together here.

2.4.6.1 Text Scrollbar

Scrollbars are one of the basic controls in any Graphical User Interface, because scrolling of a larger window over a small display area is such a common operation. You can scroll a command history list, a list of files, a list of colors to be used in displaying windows, or a list of paintbrushes in a drawing program. A consistent GUI such as OPEN LOOK makes it easy for the user by using the same scrolling notations throughout. Let’s first look at the scrollbar and how it is used for the common operation of scrolling text. First, we need to know the names of the parts of the scrollbar, as shown in Figure 2-18.

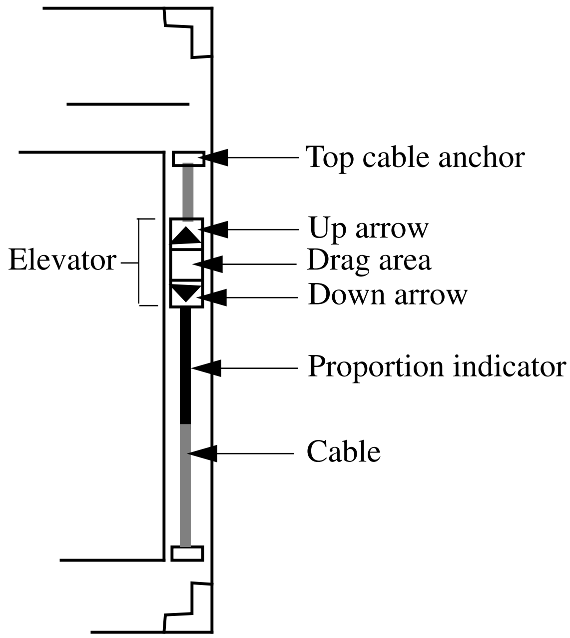

Figure 2-18. Parts of the OPEN LOOK Scrollbar

Here we present just the highlights of the scrollbar. A full explanation will be presented in Section 5.4, "Using the OPEN LOOK Scrollbar" on Page 108.

- To move the text up, click and hold SELECT in the centre box of the elevator, and drag it up or down. As you drag the elevator down, the text will move up, which means you are moving down in the file.
- To move the text down, drag the elevator upward.
- To move toward the top of the file, you can also click the little arrow on the top of the elevator; to move down, click the arrow on the bottom.
- To move up or down a "chunk" at a time (normally a page or half-page), click SELECT on the cable above or below the elevator.

There is much more to the OPEN LOOK scrollbar functionality, including a menu that lets you jump around and split the window into multiple views. We discuss the scrollbar in more detail in Chapter 5, *The cmdtool/shelltool Terminal Emulator*, when we have some text to work with.

2.4.6.2 Scrolling List

The scrolling list lets you view several of a (potentially larger) number of choices, and select one or more of the items. Each item that you select (if the application allows multiple selections) is highlighted. You can visit parts of the list that aren't currently visible by appropriate use of the scrollbar, and select item(s) just by moving the pointer onto them and clicking SELECT. So in a way, this type of item behaves like a cross between a text scrollbar and a vertical list of choices—you can scroll up or down, and you can select items from the list of choices. The *olitsampler* program shown earlier features a scrolling list, as does the simple window shown in Figure 2-18.

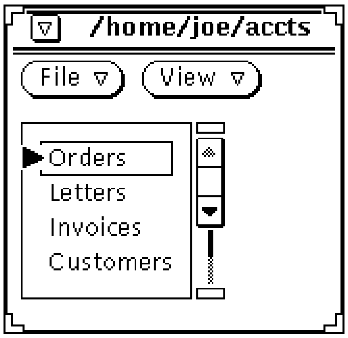

Figure 2-19. Scrolling list file browser

You can scroll the list up or down by using the scrollbar. In effect, the scrollbar here behaves pretty much the same whether it is scrolling around in a text file (scrollbar) or in a list (scrolling list). The Scrolling List scrollbar features a menu similar to the menu on a regular scrollbar.

At any point, you can select an item in the list just by clicking SELECT on its name. Like the Choices items discussed previously, Scrolling Lists can be either Exclusive or Non-Exclusive. In an Exclusive Scrolling List—commonly used for selecting one file or topic to open in an "Open..." pop-up window—only one item can be active at a time; selecting one deselects the currently active one. In a NonExclusive Scrolling List, each item can be selected or deselected individually.

A common shortcut is to double-click SELECT on an item to choose it, particularly where the scrolling list is being used to choose a file to open.

2.4.6.3 Slider

Like the scrollbar and scrolling list, a *slider* lets you move around. However, its purpose is not to actually move something, but to adjust a numerical value within a specified range. An example slider with its default value is shown in Figure 2-20:

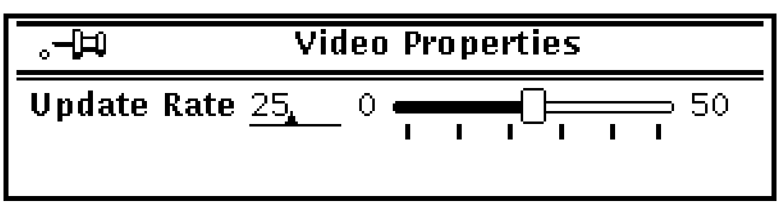

Figure 2-20. A slider

This slider has a range of from zero to fifty; it might represent frames per second on a video source, where zero means pause or "still frame," and positive numbers control the refresh rate. You adjust the value as with a scrollbar, either by dragging the central box or "grip," the analog of the scrollbar's elevator, or by clicking on either side of the grip. If you grab the grip by clicking and holding the SELECT pointer button on it, it gets highlighted, and you can drag it as much as you like until you let go (Figure 2-21).

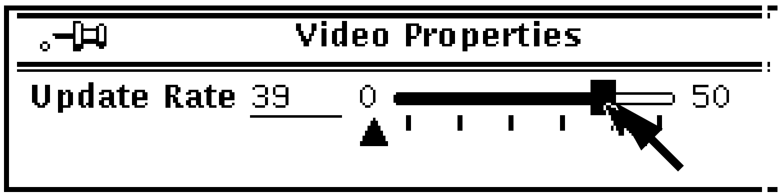

Figure 2-21. Slider being adjusted.

While most sliders are horizontal, some applications use vertical sliders.

The value shown numerically is in a text field, in which you can type an exact value (and press RETURN); this will set the slider to that value.

2.4.7 Gauge

The gauge item shown in Figure 2-22 is rather like a slider, except that it can only be set by the application. That is, it is like a "read-only" slider; you cannot move it with the mouse, but the program controlling the window sets its value to provide you with information.

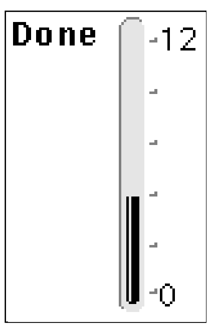

Figure 2-22. A gauge

2.4.8 Text Fields

There are several kinds of controls that let you manipulate text. A simple text field (see Figure 2-23) has a label and one line for insertion of text. The label is "Filesystem", and the current value is */dev/rsd0a.*

Text fields behave a little differently than other controls. To work with them, you must not only ensure that the window they are in has the input focus, but you must *also* ensure that the particular text field has the focus. You can do this by clicking the SELECT pointer button with the pointer in the text field; most applications will also allow you to move from one textfield to the next by use of the <TAB> key or some other accelerator. Once you have activated a text item, its *insertion point* ("caret" marker) will turn from gray to black to

Figure 2-23. A text field

indicate that it is active. Then, you can type text at the insertion point, or you can use any of several control keys:

- CTRL-U clears the text field
- CTRL-H, or Backspace, or DEL, erases one character at a time

A complete list appears in Section 5.7.1, “The Editing Keys” on Page 124.

Although a text field may only display a small number of characters, the application may be willing to accept more. Let’s say we have a version of UNIX that uses very long device names, one of which we are typing into a *Backup* program, and we use up all the character positions on the screen. This is shown in Figure 2-24.

Figure 2-24. Text field about to overflow

Nothing special happens yet, but as we type the next character, OPEN LOOK gives us more feedback, and lets us continue typing, by sliding the text to the left and giving us a *More arrow* at the left end (Figure 2-25).

Figure 2-25. Textfield with More arrow

If we wish to view text that is hidden behind a More arrow, we just click on the arrow, and characters will appear one at a time. If necessary, another More arrow will be created at the other edge of the text field. You can type as many characters as are needed in most applications; if you type more than the application is willing to accept, the text field will stop accepting input, and beep if you try to keep typing.

There are many other characters that can be used in text fields. As well, lines of a text field behave similarly to lines in an OPEN LOOK text window. There is much more detail on this topic in Section 5.6, "Editing Text in OPEN LOOK Applications" on Page 120.

2.4.9 Help Me

Although the OPEN LOOK GUI is remarkably consistent, there are quite a few different types of objects, and they can be assigned different meanings by the application programmer. So you need some way to find out what a particular control is for. Most window systems just tell you to "read the manual," or at best give you a "Help" function that is not aware of what you are doing (that is, they don't give you "context-sensitive" help). But the designers of OPEN LOOK knew that most users would want something more powerful: something based on window technology. So the OPEN LOOK GUI includes a facility for getting online, context-sensitive help. Put the pointer on any OPEN LOOK control, and press the HELP key. This key is labeled HELP on some keyboards (on a Sun-3 keyboard, it is labelled F1). If the developer of the application has provided a help file[†] and it has been installed correctly, you will see a window created to bring you the help data *about that particular OPEN LOOK control.* Figure 2-25 shows an example of a help screen for the default Workspace menu under OpenWindows. This is what you would see if you moved the pointer into the (pinned) Workspace menu and pressed the HELP key.

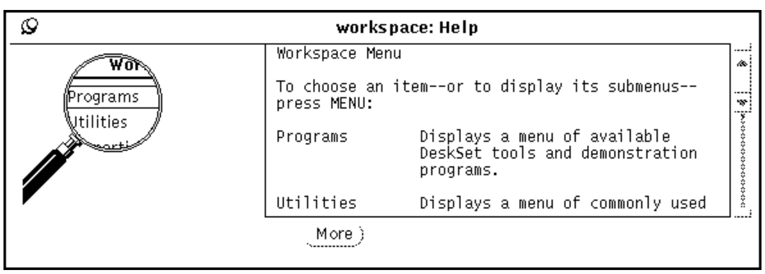

Figure 2-26. Help screen for a QUIT button

The Help screen includes a little magnifying glass to show what it is "zooming in" on; in this case, the Workspace menu. If you press HELP with the pointer over a command button, then only the help for that button will be displayed. If the help text is long, as in this example, there will be a text scrollbar, described a few pages ago. If there is no help file for a particular dialog item, or the help file for that application is not installed, you will see a small dialog box with the message "No help is available for . . ." and the name of the application and object for which the help is missing. For some commercial applications, you may either have to install the help file as part of the installation, or else set the environment variable HELPPATH to include the directory where the help files live. If the help function fails for one program, check that the help files were installed properly and that HELPPATH is set correctly.[‡]

† The help file for an XView application named *foot* has the filename *foot.info*.

You could also use the HELPPATH facility to provide your own alternate help files. If you wanted them to be more detailed or more terse than those provided, you could copy that program's help data into a private help library, put its name at the front of the HELPPATH list, and edit the copy of the help file. The facility is as flexible as the UNIX execution search path on which it is patterned.

One limit is that you cannot have part of one help file in one directory and part in another; that is, the system only opens one library help file per application name. Another is that some programs written with the OLIT toolkit may have the help data "hard coded," and you cannot provide your own help files for such applications. This is probably not such a good idea, but OLIT programs may include it, so beware.

2.5 Moving, Resizing, and Iconifying Windows

As with most window systems, you can use the titlebar to perform common window operations such as moving, resizing, and closing the window. Unlike many X11 window managers, however, OPEN LOOK lets you use the entire frame around the window to do these operations, so you don't have to have the titlebar visible. In the two-dimensional monochrome OPEN LOOK, the frame is very thin on the sides and bottom, but you can find it with a bit of practice. If you watch the screen cursor carefully, it "blinks" briefly as you move onto the frame around a window. In color OPEN LOOK, the frame is normally in a color that contrasts with the background and with other windows, so it's easier to see. You can set it to a different color using X resource variables, shown in *Section 3.4, "Customizing the X Environment: Specifying Resources" on Page 69.* Unlike some other window managers such as *twm*, you cannot change the border thickness under the OPEN LOOK GUI.

Once the pointer focus is in the titlebar or frame, you can use the MENU button to get the *Window menu*. The normal *Window* menu provided by the OPEN LOOK Window Manager has the following items: *Close, Full Size, Move, Resize, Properties, Back, Refresh*, and *Quit*. These will all be discussed in more detail in Chapter 6, *Using the OPEN LOOK Window Manager*; here is a brief look at the more commonly-used items:

- *Close* iconifies (or "minimizes") a window. The window disappears from your screen, and is replaced by a small window with the same name and (usually) a graphic representing the program. These icons appear in a row along one edge of your screen.
- *Back* moves the window behind any other windows that it overlaps. Useful in finding a window when you have several that overlap.
- *Refresh* redisplays the window, in case bits of it were overwritten or lost.
- *Quit* terminates the application; if the application has its own means of termination, such as logging out of a terminal emulator, that should be used instead of *Quit*. However, the presence of this item on the Window Menu is the reason why most OPEN LOOK applications do not provide a *Quit* button of their own.

‡ HELPPATH is a colon-separated list of directories to look in for help files. It defaults to */usr/lib/help* and */usr/openwin/lib/help* in AT&T-OL and Sun's OpenWindows respectively.

Figure 2-27 provides an example of closing a window using *Close*.

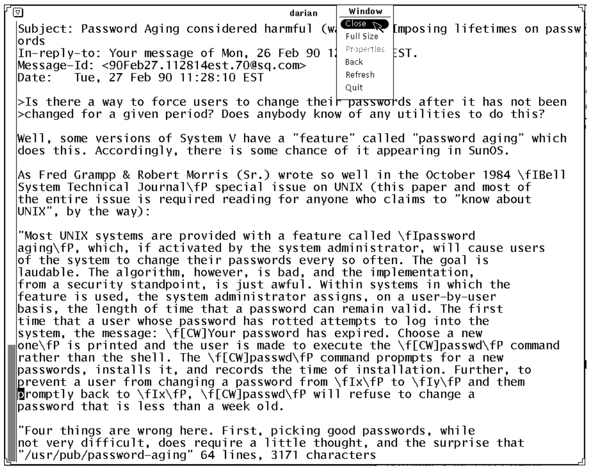

Figure 2-27. Closing a window from the window menu

See Section 6.4, "The Window Menu: Moving, reshaping, and iconifying Windows" on Page 144, for more details on all these operations.

2.6 Exiting an xterm or cmdtool Window

When you are finished using an *xterm* or *shelltool* window and want to get rid of it, you can remove in one of two ways. You can kill a terminal window by typing whatever command you usually use to log off your system. On UNIX this might be *exit* or CTRL-D. In Figure 2-28, we'll quit an *xterm* window by typing *exit*.

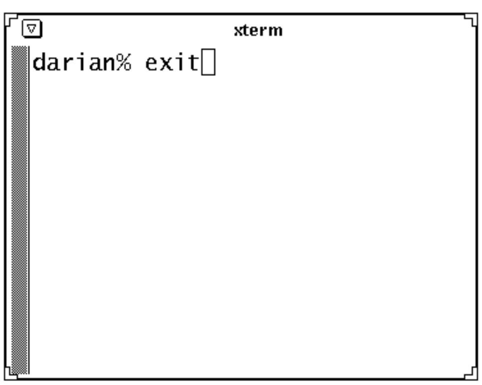

Figure 2-28. Exiting an xterm window.

A second, more drastic method of terminating the window is using the *Quit* item on the *Window* menu (see Figure 2-29). However, some programs that do not fully conform to the ICCCM (see Volume Zero, *X Protocol Reference Manual*), even including some versions of the MIT *xterm*, will give obnoxious messages when killed off this way.

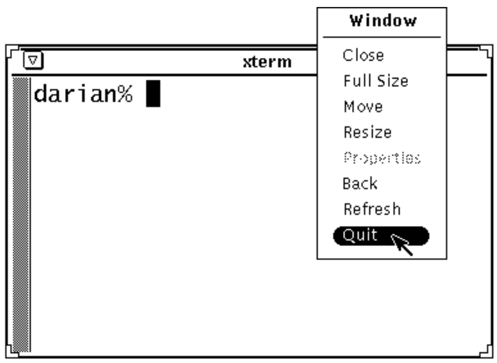

Figure 2-29. Quitting a window

Notice that in either case when we terminate the second terminal window, the first terminal window does not take over the input focus. When explicit focus is being used and a window is terminated, the input focus reverts to the Workspace, rather than to any particular window.

Be aware that on some systems (OpenWindows is not one of them), terminating the "login" terminal emulator window (the first terminal emulator to appear) kills the X server† and all associated clients. Be sure to finish any work you might be doing in all other *xterm* windows before terminating the *xterm* login window. For example, if you are in an editor such as *vi*, be sure to save your data before you terminate that window or the session. If you are on such a system, it may be wise to iconify the login window and use other *xterm* windows instead, so that you don't inadvertently terminate it. Remember: you iconify a window by placing the pointer on the pull-down symbol at the left end of the titlebar, and clicking the SELECT button.

If you are worried about typing CTRL-D accidentally, and you normally use the C shell (csh), you can enter:

```
% set ignoreeof
```

in the login window. Typing *exit* then becomes the only way you can terminate the window.

Note that some C shell implementations have an `autologout` variable, which will automatically terminate the shell if there is no activity for a given period of time. If your C shell supports this feature, be sure to disable it in the login *xterm* window using this command:

† If *xdm* is running X, the server will be reset but only after all client processes have been killed.

```
% unset autologout
```

The contributed software client *contool*, by Chuck Musciano of Harris Corporation, provides an alternative console window. It is normally iconified, to save "screen real estate," and can be set to flash or even pop open when important console messages appear. And it's an output-only window, so you can't accidentally kill it off by typing CTRL-D or exit in the window.†

For *xterm* only, an alternative way of logging off the system is to terminate an *xterm* window by selecting *Send HUP Signal*, *Send TERM Signal*, *Send KILL Signal* or (most reasonably), *Quit* from the *xterm Main Options* menu. (These menu options send different signals to the *xterm* process. Depending on what signals your operating system recognizes, some of the options may not work as intended. See Appendix A, *The xterm/olterm Terminal Emulator*, for more information.)

As we've seen, OPEN LOOK also provides the QUIT item in the *Window* menu to remove any client window, including an *xterm*.

2.7 Summary

Now you know the basics of managing windows under X11 and OPEN LOOK. In the next chapter we show you how to start many new windows, and how to customize the system so your desired windows start whenever you begin using the system.

† This program does require a certain feature in the operating system. To see if you can run it, check your vendor's documentation for the ioctl request TIOCCONS (4BSD) or SRIOCSREDIR (SVR4)— see the tty(4) or console(4) man page on your system—normally used by *cmdtool -C* or *xterm -C* to take over management of the console. If it's there, as it should be on any modern UNIX, you can use contool just by compiling and installing it. Otherwise, you or your support programmer would have to find out how *xterm -C* works on your system, if at all, and write similar code into *contool*.