CHAPTER 9

Graphics Clients

The X Window System is rich in graphical capabilities[†]. This chapter explores some of the graphics utility programs that come standard with X, as well as some of the OPEN LOOK extensions and some commercial and contributed software. We look at:

- programs for collecting and viewing bitmapped graphics from the system,
- programs for editing bitmap files,
- a brief survey of commercial desktop graphics programs for the OPEN LOOK GUI,
- programs for displaying PostScript graphical images, and
- programs for editing font images for use in X11.

9.1 Bitmap Gathering and Viewing

A *bitmap* or *raster* is a grid of pixels, or picture elements, each of which is white, black, or in the case of a color display, a color. There are several methods of obtaining bitmap files. This section describes several programs that use a method called "screen capture"; this term means recording in a file the contents of the actual display. Another is "bitmap editing", discussed in Section 9.2, "Bitmap Editing and Conversion" on Page 231.

After you've been using any window system for a while, there comes a time when you need to record the image of what is on the screen. It may be that you are having problems with somebody else's software, and want to show exactly what all the pop-up windows looked like at a particular time. Or you may be writing documentation about how to use some par-

† For *developing* full-fledged graphics software, see PHIGS, the Programmers' Hierarchical Interactive Graphics Systems, and PEX, the Phigs Extension to X, described in the *PHIGS Programming Manual*. When PHIGS/PEX becomes more popular, we can expect to see a real increase in the number of impressive graphics programs running under X.

ticular graphics tool, and want to get a bitmap file to show how the screen looks at some particular point. OpenWindows features a program called *snapshot* that can grab the whole screen, one window, or a defined region, and lets you save or print the image. *xwd* is a standard X11 program to dump an entire window; it can be used with some filter programs to print screen windows. And all X11 distributions include a program called *xmag* for looking at part of the screen.

Two contributed clients, *xloadimage* and *xv*, do a good job of displaying graphics. They specialize in different jobs: *xloadimage* can load well-nigh any graphics image you are likely to find. The *xv* client can load a smaller variety of image formats, but can do far more with the image once it's loaded, including cropping, scaling, and writing it out. Both programs have the convenient property that they can directly read files compressed with the standard UNIX *compress* facility. *xv* can also grab and save a region of the screen.

See also the unbundled Sun *ShowMe* program in Section 7.18, "ShowMe - graphical conferencing" on Page 180.

9.1.1 Snapshot – the on-screen photographer (OpenWindows)

As we've seen, you can use *xwd* to dump one window or the whole screen, but it's cumbersome to use when you want a region of the screen that doesn't correspond to one window. That is where *snapshot* comes in. *Snapshot* takes a picture of any part of your screen, just as if you had taken an instant camera and shot a photograph of the screen. The advantages of *snapshot* are that it's always in focus, never runs out of film (until your disk is full), and lets you record either the whole screen, or any part of it, or any one specific window. You'd record the whole screen only rarely. To record one window with some dialog boxes, you might use the region (part of screen) mode. And to show a close-up of one program's window, just record that window.

Figure 9-1 shows the control panel for snapshot[†].

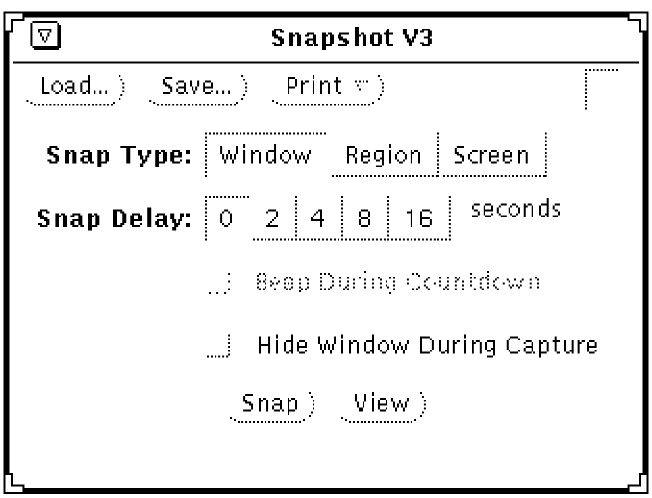


Figure 9-1. Snapshot's control panel

The main operations are `Snap` and `View`. The three kinds of things you can "snap" are a single `Window`, a selected `Region`, or the entire `Screen`. Once you have a snapshot in the program's memory, you can select `View` to display it (to ensure that what you saw is what you got, or to view a snapshot that you loaded with `Load`), or `Print`. And you can `Load` or `Save` an image; the former lets you use this program as a viewer as well as a snapshot program. Let's look at each of these operations in a bit more detail now.

When you press `Snap` with the Type set to `Window`, the `Snap` button becomes busy. Here we are about to take a picture:

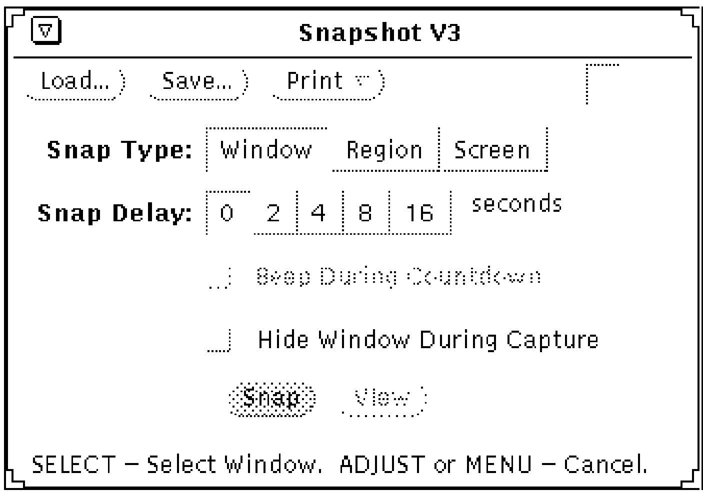


Figure 9-2. Snapshot: Snap Window operation

Notice that the left footer becomes

```
SELECT – Select Window. ADJUST or MENU – Cancel.
```

to tell you what your two choices are. To snapshot the window you want, then, move the pointer into that window, and click the SELECT mouse button. Now *snapshot* goes and reads the contents of the window from X, and saves it for later writing into a file. When you select a single window, snapshot gives you the image of just that window - the titlebar is excluded. To include the titlebar, simply to click the button in the *olwm* titlebar instead of in the window.Your normal next step is *View* to convince yourself that you really succeeded in saving the given window into the program's buffer. Just press the SELECT mouse button on *View* and you will get a display that should be a copy of the window you clicked in. When done with this display, un-pin it or pull down its *Window Menu* and release on *Dismiss*. Then `Save` the image to a file if it's what you want. You can save the file either by the *Save...* menu button, or by dragging it from the drag-and-drop target onto any running File Manager, which will save it under the name *raster* (or *raster0*, *raster1*, ...)

† The OpenWindows Version 2 snapshot has a different window layout but behaves in essentially the same way.

in its current directory; this may be preferable to trying to type a long directory path in the *Save* dialog.

The `Snap Region` control works similarly, but lets you select *any* rectangular area of the screen. For example, if we want to document how a row of icons looks:

Figure 9-3. The Desired Picture

We can move the pointer to one corner (say the upper left), click SELECT and hold it, drag the pointer to the opposite corner (in this case the lower right), and let go. Then when we click the ADJUST button, the snapshot will actually be taken. If we realize we didn't want to proceed, we just click the MENU button to cancel the operation (the word Cancel sometimes got truncated due to a bug in Version 2, or even in Version 3 with -scale large in effect).

If you want to capture an image of the *entire* screen, set the Type to Screen and click SELECT, and it will be done. There is no request for you to Click SELECT, since there is no need; there only is one root window or screen,† so selecting *Snap Screen* is enough – *snapshot* goes ahead and does it. Immediately. This may not always be what you want, though. If you are trying to arrange the entire display, but you don't want the camera (*snapshot*) to appear in the picture, you can check the *Hide Snapshot During Capture* checkbox. When you select this, it automatically sets the timer to 8 seconds, and keeps it there as long as *Hide Snapshot During Capture* is enabled. When not using the Hide feature, the timer can be set to any number of seconds shown in the exclusive choice - 0, 2, 4, 8 or 16. And an audible countdown can be enabled or disabled by selecting the *Timer Bell* toggle. The timer will normally beep while it is running.

If you try to take a snapshot of a program that has a popped up menu or a notice, you will notice that it doesn't work. This is because popup menus and notices use what is called a *grab* in X11 terminology – they take exclusive control of the X Server, and mouse button clicks anywhere in the screen are passed to that program. What you need is a way to make the program run without doing the grab. You have to re-start the application in a special non-grab mode, popup the menu or notice that you want to show, and then you usually have to kill off the application, because it cannot work without the grab. If the program was written with the XView toolkit, use the `-Wfsdb` (window full-screen debug) command line option. For example, Figure 9-2 was produced with the command

```
snapshot -Wfsdb &
```

† If you're on a machine with multiple screens, the *Snap Screen* option captures the screen that it is running on.

to make the *snapshot* program put up its notifier without grabbing the server; this allowed me to click on *snapshot*'s `Snap Region` command and have the mouse clicks passed to *snapshot* rather than just being able to click in the notice.

Another method of grabbing such things is to use the delay setting along with `Snap Region`, and pop up the menu during the delay.

The control *View Snap* has already been discussed; it displays the image in the program's buffer. You can even use this to display Sun Raster format files created by other programs: as long as the name in the *Directory* and *File* text fields matches a valid file, *snapshot* will try to display it.

`Load...` and `Save...` let you save your snapshot, or load another one for examination. In either case, a simple pinnable Load/Save dialog box is popped up in which you can specify the directory (if not the current directory) and the name of the file.

Lastly, the *Print* control lets you print the current image. If you just click SELECT on this control, the current image will be printed. But it has a menu mark; if you pull it down, you see a menu of `Print Snap` or `Options...` The last item has a window mark ('...'), so if you click on it an options sheet appears, like the one shown in Figure 9-4. This lets you adjust various options such as portrait or landscape mode.

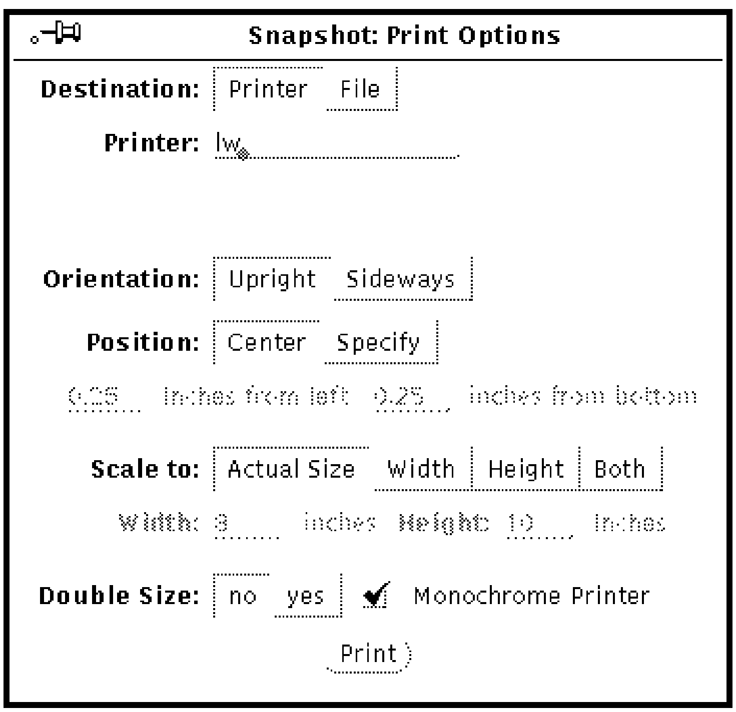


Figure 9-4. Snapshot: Print Options screen

There are numerous print options. You can have the image "printed" to a file, but the default is to print it to a real printer, whose name defaults to `lw` (LaserWriter, a generic PostScript printer). If you click on `File`, the `Printer` field is replaced by Directory and File fields. You can request that the snap be printed in portrait ("Upright") or landscape ("Sideways"), and either Centered or at a particular position. You can have the image printed actual size, double size, or you can specify Width, Height, or Both. Since at present most of us can afford only monochrome printers, that is the default. But if you have a color printer, feel free to check the Monochrome box off to print in color. In all cases, the program *rash*, a standard part of OpenWindows, is used to convert the image from Raster format into PostScript.

A quicker way of printing is to drag the image from the *snapshot* program's drag-and-drop target (the rectangle in the upper right of the main window) onto a copy of the *printtool* program (described in Chapter 7, *The OpenWindows DeskSet Clients*). This way you need only set your print options once, in *printtool*, and all your print requests that go through *printtool* will use these settings.

Snapshot works very nicely with normal X11 windows on OpenWindows. But Sun's OpenWindows also allows you to run NeWS and SunView clients. Because *snapshot* is an X11 program, it cannot know about either NeWS or SunView clients. NeWS client windows can easily be recorded by using *snapshot*'s *Snap Region* facility. However, SunView clients are totally invisible. Neither *xwd* (with the -root option) nor *snapshot* can "see" such clients. This is because of the way the sharing of the screen is implemented; see Appendix L, *Running SunView Applications on OpenWindows*, for details. What matters here is how we can get an image of such programs. The best way we've found so far is to use *screendump*, an ancient and venerable SunOS program that simply copies the physical contents of the frame buffer into a file on disk. Since *screendump* is not part of any windowing system, it doesn't care about NeWS, X11, SunView, or any difficulties or disagreements among them. It simply copies what is visible on your screen into a disk file for you. Let's say you want to get a screen image of the SunView *fontedit* program - this is how Figure 9-22 of the *fontedit* program at the end of this chapter was produced. First, start the program that you want to capture, and set its screen up the way you want it. Then, move the program's window† to the upper left of the screen. This is position (0,0), and makes the following steps easier. From a shell window, run

```
screendump > screen.rs
```

But that dumps the entire screen. If you only want one part of the screen, you can specify options that request part of the screen, or use a copy of *snapshot* with the *Grab Region* control to select the part you want. If you prefer command-line-based methods, you could use the PBMplus package, described later in this chapter, to cut out the part you need. For this example, we used

```
screendump -X 680 -Y 710 > fontedit.rs
```

† The SunView mouse conventions are different from those of OPEN LOOK; see Appendix L, *Running SunView Applications on OpenWindows*. Briefly: click and hold the middle pointer button on the titlebar to move a SunView window.

(note upper case X and Y) to just get the 680x710 region from the upper-left corner. Unfortunately, the 680 and 710 had to be derived experimentally, since *xwininfo* won't report on a SunView window, even in OpenWindows Version 3.0. For more information on *screendump*, see the reference manual page in your SunOS manual set.

9.1.2 xwd, xwud – Dump an X Window

If you don't have the OpenWindows DeskSet tools, you may wish to use the MIT standard client *xwd* to capture images in a "dump" format. *xwd* stores window images in a formatted window dump file. This file can be read by certain other X utilities for redisplay, printing, editing, formatting, image processing, etc.

To create a window dump file, type:

```
% xwd > file
```

The pointer will change to a small cross-hair symbol. Move the cross-hair pointer to the desired window and click any pointer button. The keyboard bell rings once when the dump starts and twice in rapid succession when the dump is finished.

To make a dump of the entire root window (and all windows on it), use the `-root` option:

```
% xwd -root > file
```

When you select a single window, *xwd* takes an image of the window proper. To include the titlebar, simply to click the button in the *olwm* titlebar instead of in the window.

xwd allows you to capture a single window or the entire root window. But what if you want an image that includes more than one window or parts of multiple windows? One method is to use *xmag* (described below) to capture an image of multiple windows and then use *xwd* on the *xmag* window! Since *xmag* is intended to magnify, if you want the window image to be the actual size, you must specify that no magnification is performed. To do this, you run *xmag* with the option `-mag 1`. See the *xmag* reference page in Part Three of this guide for more information. For OpenWindows users, a more flexible method of capturing part of the screen is to use the "Snap Region" option of *snapshot*, which we'll describe shortly.

To redisplay a file created with *xwd* in a window on the screen, use the *xwud* client, an undumping utility. Specify the dump file to display as an argument to the `-in` option:

```
% xwud -in file
```

When you get tired of looking at it, you can remove the image by pulling down and releasing *Quit* in the *Window Menu*. If you prefer, you can type Control-C (your INTR character) in the shell window from which you started *xwud*.

9.1.3 xpr, xdpr – Print an X Window

xpr takes as input an X Window System dump file produced by *xwd* and converts it to a printer-specific format that can be printed on the DEC LN03 or LA100 printer, a PostScript printer such as the Apple LaserWriter, the IBM PP3812 page printer, and as of Release 4, the HP LaserJet (or other PCL printers) or the HP PaintJet. By default, output is formatted for the obscure DEC LN03 printer. Use the `-device` option to format for another printer. For example, to format a window dump file for a PostScript printer, type:

```
% xpr -device ps file > file.ps
```

Other options allow you to change the size, add headers or footers, and so on. See the *xpr* reference page in Part Three of this guide for details.

You can use *xwd* and *xpr* together, using the standard UNIX pipe mechanism. For example:

```
% xwd | xpr -device ps | lpr
```

Because this is such a common operation, there is an *xdpr* shell script (csh) that rolls these three commands into one. *xdpr* accepts most of the options accepted by *xwd*, *xpr*, and *lpr* (1). Thus, you could use the command:

```
% xdpr -device ps
```

to take a window dump (`xwd`), convert that file to PostScript (`xpr -device ps`), and print the output (`lpr`). See the *xdpr* reference page in Part Three of this guide for more information.

If you routinely use *xdpr* with some device other than the now-defunct LN03, you can provide another default device by editing a copy of *xdpr*†

Hopefully a future version of *xpr* will allow use of X resources (see Chapter 12, *Setting Resources*) to specify the default printer.

Note that when you start piping together the output of X clients, you run into some ambiguities. For example, if you pipe the output of *xwd* to *xpr* and for some reason the *xpr* command fails, *xwd* will still be there waiting for pointer input. The original UNIX pipe mechanism doesn't have the concept of data dependent on pointer input! The integration of the UNIX model of computing (in which standard input and output are always recognized) and the window model is not always complete, sometimes leading to unexpected behavior.

As an even more flagrant example, you can create a pipe between two programs, the first of which doesn't produce standard output and the second of which doesn't recognize standard input. The shell doesn't know any better and the programs themselves go on their merry way with pointer and windows.

However, it is nice to know that you can pipe together program output, even when some of those programs may not produce output until you intervene with the pointer.

Even without pipes, you should start thinking about how these programs could work together. For example, the pictures of fonts in Appendix C, *Standard Bitmaps - X11, OPEN LOOK and OpenWindows*, were created by these steps:

1. Display a font with *xfd*. (See Chapter 10, *X11, OPEN LOOK and OpenWindows Font Specification*, for instructions on how to use xfd.).
2. Resize the window to improve readability, using the window manager.

† You may wish to change the initialization of the variable *xprv* from set xprv=() to set xprv=(-device ps) near the start of the file.

3. Create a window dump file with the command `xwd > file`.
4. Create a PostScript file from the dump with the command:

```
xpr -device ps file > file.ps
```

Print the PostScript file on a PostScript printer with the standard print command *lp* or *lpr* command.

Even though the UNIX shell will accept a pipe between *xfd*, *xwd*, and *xpr*, what actually happens is that *xwd* starts up faster than *xfd*, and is ready to dump a window before the *xfd* window appears.

9.1.4 Xloadimage

The *contributed client Xloadimage* does not save images, but will load nearly any graphics image, and present it on your X screen. To use it to view a file, say *face.xbm*, simply type

```
xloadimage face.xbm
```

If the file is a picture in any of the known formats, you will get a message like

```
face.xbm is a 47x64 X11 bitmap file titled 'UofT'
```

and then a window with that image in it will appear:

Figure 9-5. xloadimage face.xbm

In some bitmap formats a "title" is stored with each image, while in other formats it is not. If your bitmap file has a title, it will be printed in the message (as it was in this example) and displayed in the window's titlebar (though it will be truncated if the bitmap is small). If there is no title, the last part of the pathname (the "filename" part) will be used.

Multiple bitmaps may be displayed just by giving more than one bitmap file name on the command line. To display multiple bitmaps sequentially, you just give the filenames on the command line in the order you want them to appear†. To move from one bitmap to another, type a letter "n" in the bitmap display window.

For example,

```
xloadimage face1.xbm face2.xbm face3.xbm
```

This would display the bitmap from *face1.xbm*; to get at the second bitmap, move the pointer into the bitmap and type "n" for "next". There are a few other single-letter commands listed in the reference manual; the most useful is probably "q" for "quit."

† An earlier version of this program would combine multiple bitmaps given on the command line into a single bitmap. If you have this version, add `-slideshow` to display multiple bitmaps.

If the bitmap is too large to fit comfortably on your screen, *xloadimage* will display only a part of it. However, the program does not use scrollbars to move around. Instead, you "grab" the bitmap by pushing pointer button number one, and "drag" the display around. This does not conform to any standard (OPEN LOOK, Motif, or Athena), but in practice it works fine.

The number of formats supported by *xloadimage* is impressive, and it grows regularly because the program source code is structured to make it easy for C programmers to add code for new file types. Here is a recent list of the more common formats:

Table 9-1. Common XLoadImage Bitmap Formats

Formats Supported
FBM (Fuzzy BitMap library) Image
Sun Rasterfile
Sun Icon file
CMU WM Raster
Portable Bit Map (PBM, PGM, PPM)
USENIX/UUNET Faces Project
GIF Image
X Window Dump (from *xwd*)
X Bitmap
X Pixmap
Group 3 FAX Image
MacPaint Image

Some programmers have added their own proprietary formats as well; these are not in widespread use so they aren't shown here. It's fairly easy for a C programmer to add new types.

One problem with *xloadimage* is that it has difficulty with certain full-color images; it often fails to get the colors right. This will probably be corrected in a future release, though.

9.1.5 Xv

While *xloadimage* has a real breadth of file types, another contributed program called *xv* has a wide variety of operations. *Xv* as distributed can only read the GIF, PBM/PGM/PPM, XBM, SunRaster, JPeg and TIFF formats, as well as one local format called PM. But it can do quite a bit with images. It can re-scale an image. It can rotate images in 90-degree steps. It can "crop" images, throwing away extraneous sections of background. It can change the coloring (use fewer colors, perform “gamma correction”, etc.). It can write out images in the same formats it reads, plus a few others such as PostScript. And it can grab regions of the screens (but not particular windows). As the program’s own author says:

“It slices, it dices, and it’ll balance your checkbook if you aren’t careful.”

Let’s run *xv* on the sample bitmaps we used with *xloadimage*. First, the face file. We’ll give the command

```
xv face.xbm
```

and see how it looks:

Figure 9-6. xv face.xbm.

But this is rather hard to see, so let’s resize it, using *olwm* to grab and drag one of the resize corners. When we let go, *xv* gets notified that its window is resized, so it scales the bitmap and redisplays it. For a large full-color bitmap this might take some time, but for this simple example it is almost instantaneous. The result is not impressive, because the underlying bitmap needs to be cleaned up. (We’ll look at tools for doing so in the next section.)

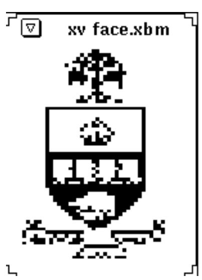


Figure 9-7. Bitmap after re-scaling.

When you want to do more to your image, you can click the right button (which is normally the MENU button in OPEN LOOK), and get the control screen. This program is not written to use OPEN LOOK; indeed, its control interface is more like that of the Macintosh™:

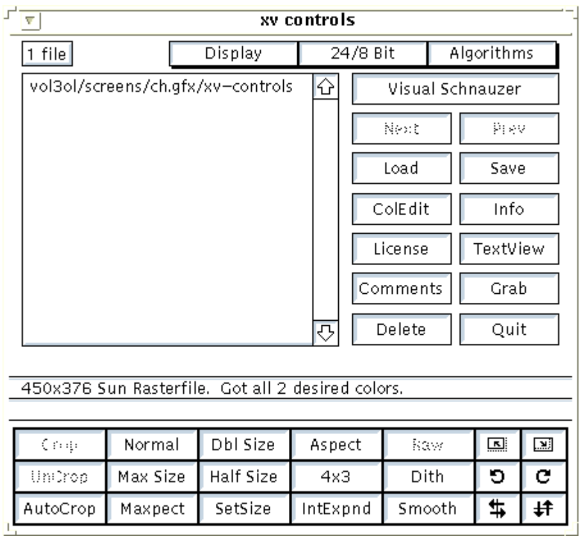


Figure 9-8. Xv controls

If you have multiple files, they will appear in the filename list. To the right of that list are buttons for *Next* and *Previous* (which are grayed out here because there is only one file), *Info* which pops up the Information window in Figure 9-9, *Save* which lets you save the file in any of several formats, and *Quit* which gets you out of *xv*.

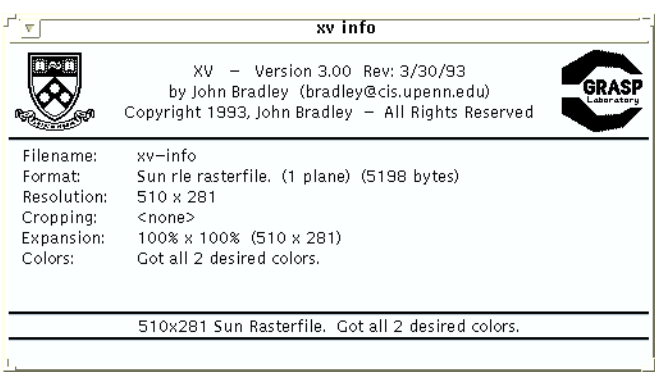


Figure 9-9. Xv Info Screen.

Notice the window mark; this is not a popup window, but another main window. If you want to get rid of this window, you can just click in it. Or, click on the *Info* button again. If you select the *Save* button, the program will pop up another dialog box, the *xv save dialog*, shown in Figure 9-10.

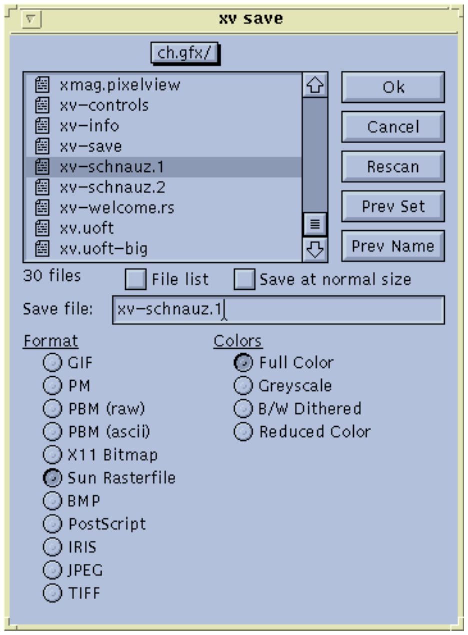


Figure 9-10. Xv Save Dialog.

Like the control panel, the save dialog is a full window, not a popup window, and it is more Mac-like than OPEN LOOK-ish. To select a format, for example, you click the left button (normally SELECT) in the circle beside the name of the format you want to use, and that circle gets a black circle inside it to indicate it is highlighted. This type of choice item is commonly called a “radio button”. An OPEN LOOK program would use either an exclusive choice or (more likely) an abbreviated choice, since you are unlikely to change formats very often.

To save a file, type its name into the *File name* text box. As soon as you start typing, the *Save* button becomes active, and can be clicked on when you have finished typing the file-name and chosen the size, format, and color model you want from the button selection panel.

The *Visual Schnauzer* gives a directory view similar to that of the Sun File Manager described in Chapter 4, *Using the OPEN LOOK File Manager*. Initially it appears as shown in Figure 9-11

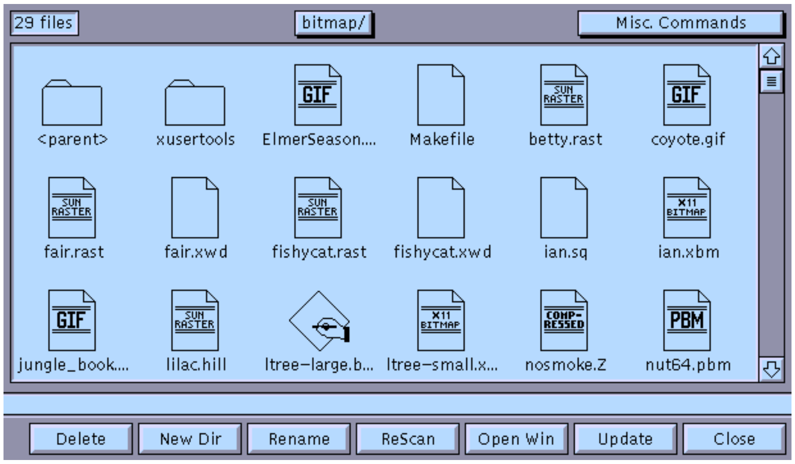


Figure 9-11. xv Visual Schnauzer

This in itself is fairly impressive. But it really comes to life when you notice that the box at the top right, labelled *Misc Commands*, is a pull-down menu (use the *Left* pointer button to activate it). If you select *Generate Icons* from this menu, any files in that directory that are graphic files that xv can handle get replaced by iconic views of the pictures. The display now looks like Figure 9-12

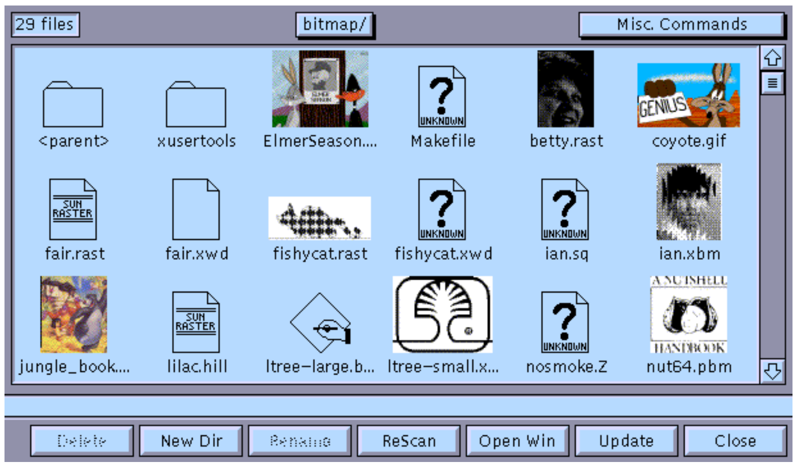


Figure 9-12. xv Visual Schnauzer with icons

This is a very powerful feature for browsing a directory of images. However, be aware tht the images are stored on disk (in a subdirectory called #), and can take up to 4Kb per image for color. When you're finished with them, you can remove the *.xvpics* directory, since it only takes a minute or two to regenerate.

9.2 Bitmap Editing and Conversion

This section looks at some programs that can be used for creating or modifying simple bitmap images such as those used by The X Window System for program icons, and by the OPEN LOOK GUI for *filemgr* icons.We also look at one more advanced program, *touchup*, for editing larger bitmap pictures. Finally, we take a look at a toolkit for converting bitmap images among dozens of different formats and otherwise massaging them.

9.2.1 iconedit (OpenWindows)

Iconedit is a bitmap editor that uses the OPEN LOOK GUI, so it is consistent with what you may expect, and it edits files in Sun "Icon" format in addition to XBM (X BitMap) and XPM (X PixMap) format. *Iconedit* is also similar in operation to *bitmap* and *olpixmap*, which will be described below. When started, *iconedit* puts up an empty edit window with numerous controls (see Figure 9-13). At this point, you can just move the pointer into the large drawing region and use the pointer buttons. The left (SELECT) button will turn individual points (pixels) on, while the middle (ADJUST) button will turn them off. Dragging the pointer with either of these buttons depressed will continue the action, so you can draw simple figures just by holding SELECT and dragging the pointer.

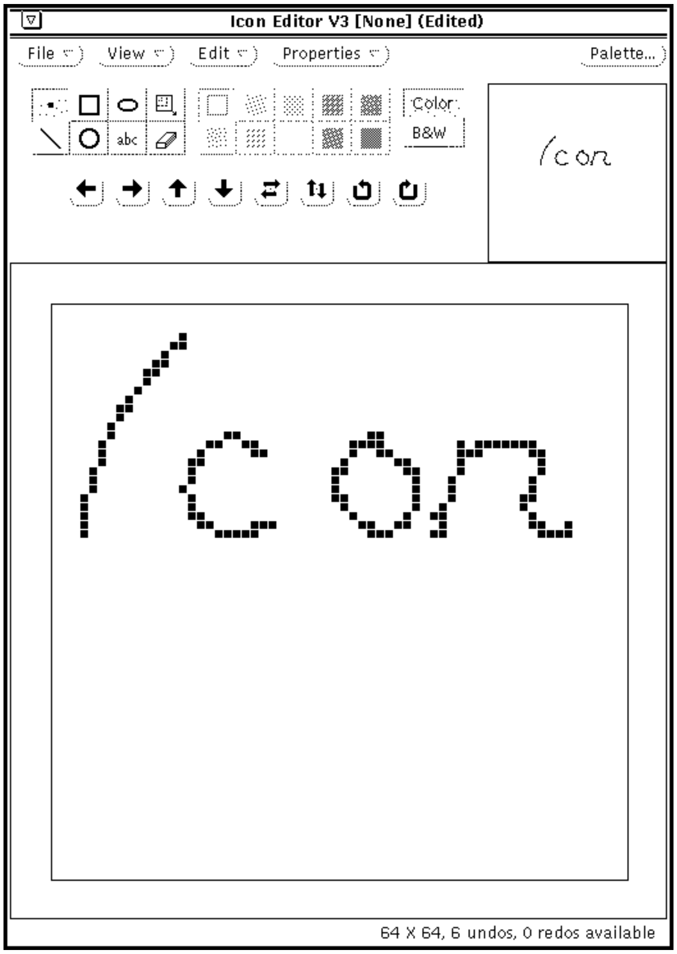


Figure 9-13. Iconedit start-up screen after a bit of freehand drawing.

The screen features Menu Buttons for File, View, Edit and Properties. File lets you load an existing image, save the screen image to a file, or print the screen image. The main window also has icons to select pixel drawing, lines, rectangles, circles, text ("abc"), region selection, and erasing. Unlike simpler programs such as *bitmap* (see Section 9.2.2, "bitmap" on Page 234), this program lets you add *text* in a variety of faces and sizes into your bitmap. These are controlled from the Text window, which appears when you click SELECT on the button with "abc" on it.

You can also select from one of several fill patterns by selecting one of the icons at the top right of the window. These will be used for filling rectangles, circles, or ellipses that you draw. The default is white, that is, no filling.

If you are on a color display, you can choose whether you want to edit a color or monochrome ("B&W") image. On a monochrome display, you can only edit monochrome images.

Iconedit also has the ability to all or part of move the bitmap up, down, left or right one pixel at a time (the four directional arrows), to flip it vertically or horizontally (the double arrows), and even to rotate the bitmap either clockwise or counterclockwise (the circular arrows). These can be very useful features.

For example, here is a simple demo icon, with the text window shown, just after we used the *Text* mode to put the words "My Demo" at the bottom:

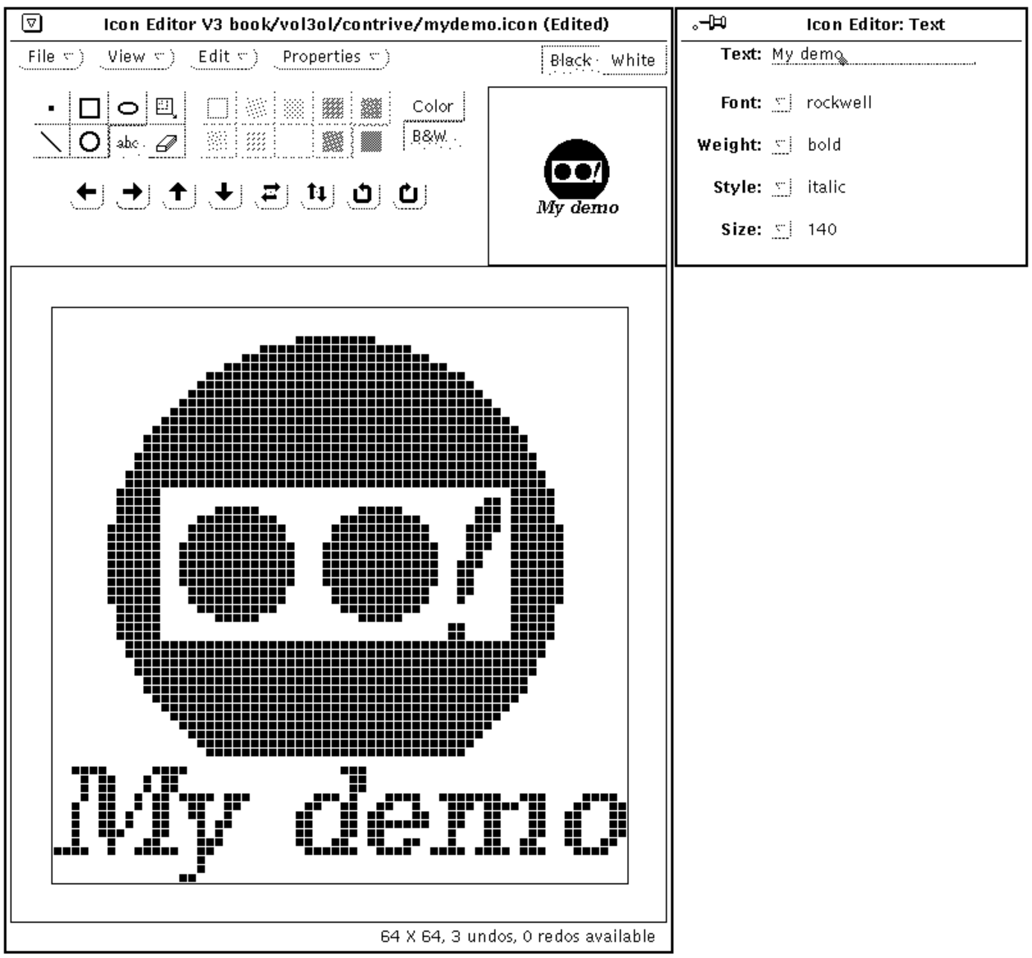


Figure 9-14. Iconedit in action

The *View* menu has only one item, that is *Grid on or off.* The Grid is a background of vertical and horizontal lines to help you align objects on the screen., The *Edit* menu (which is also the default menu when you press the MENU pointer button in the drawing area) lets you cut or paste an area of the screen, undo changes, etc. Notice that the OpenWindows L-keys, such as L6 for Copy, L8 for Paste, L10 for Cut, and L4 for Undo, all work in this context and indeed in most of *iconedit*.

The *Properties* menu does not get a Properties sheet, but lets you choose the format and size of the image. The known formats are XView Icon, X Bitmap, Color X Pixmap, and Monochrome X Pixmap. The sizes are shown in Table 9-2.

Table 9-2. IconEdit File Formats

Size	Format
16x16	Standard X Cursor
32x32	OpenWindows File Manager Icon
48x48	(unused)
64x64	*olwm* program Icon
128x128	(unused)

There is much more to *iconedit* than we have shown here. Try it out! *Iconedit* is a good choice for the OpenWindows user to edit bitmaps.

9.2.2 bitmap

If you have no OPEN LOOK package installed, you can use the *bitmap* program. the simplest and most widely available of the bitmap editing programs discussed in this section. It does not conform to the OPEN LOOK GUI nor to the OSF/Motif conventions, but it *is* included in the standard X11 software, so it is available almost anywhere that The X Window System goes.

Like *iconedit, bitmap* allows you to create and edit small *bitmaps*. You can use *bitmap* to create backgrounds, icons, and pointers. *bitmap* is primarily a programming tool for application developers. However, several applications allow you to design your own icon or background pattern with *bitmap*, save it in a bitmap file, and specify that filename on the command line.†

For example, *xsetroot* (described in Chapter 14, *Customization Clients*) allows you to specify a bitmap that will be used as the background pattern for the root window or as the root window pointer.

To invoke *bitmap*, type:

```
% bitmap &
```

† There are many bitmaps included in the X distribution. These can generally be found in the directory *$OPENWINHOME/share/include* or */usr/include/X11/bitmaps*. Samples are shown in Appendix C, *Standard Bitmaps - X11, OPEN LOOK and OpenWindows*.

If you provide a filename that contains an existing bitmap, it will be loaded (X11R4 required you to do so), otherwise a blank *bitmap* window is displayed, as shown in Figure 9-15.

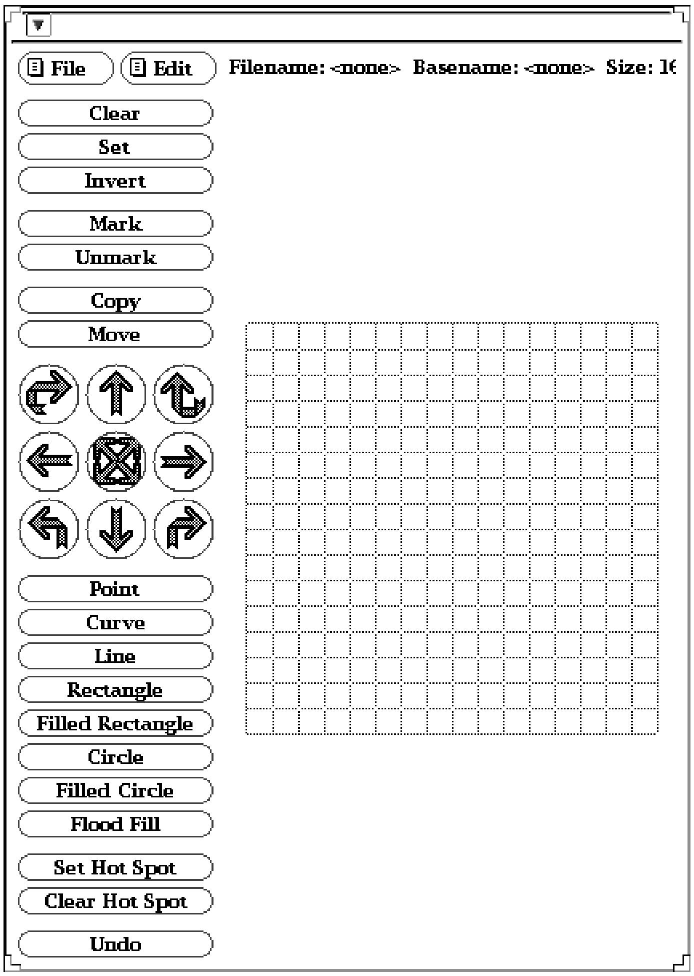


Figure 9-15. Bitmap window

The window that *bitmap* creates has three sections:

1. At the top is a partial "menu bar", with a File and Edit menu and a display of the current filename (if any) and size (16x16, but it usually gets truncated).
2. The largest section is the checkerboard grid, which is a magnified version of the bitmap you are editing. The default size grid for new bitmaps is 16x16. (For existing bitmaps, the size of the grid will be the size of the bitmap. For example, if you open

a file containing a bitmap of size 100 pixels by 62 pixels, the grid will be 100 cells wide and 62 cells high.) If the cells in the grid aren't large enough for comfortable editing, resize the window. Each square on the grid will be enlarged proportionally.

3. On the left side of the window is a list of commands in command buttons that you can invoke with any pointer button.

If you want to create a new bitmap in a grid of different proportions than the default size 16x16 grid, you can specify ***WIDTH*x*HEIGHT*** on the command line after ***filename***. For example, to create a grid double the default size, enter:

```
% bitmap filename 32x32 &
```

The ***WIDTH*x*HEIGHT*** argument is used only when creating a new bitmap. Existing bitmaps are always edited at their current size; this program *cannot* re-size bitmaps.

Keep in mind that there is an interaction between the size of the bitmap being edited and the size of the *bitmap* window. By default, each cell in the *bitmap* editing area is 13 pixels square. If the bitmap being edited is large, this may result in an application window larger than the screen. (Since *bitmap* does not provide a scrollbar, a large window may make it impossible to edit!) Specifying an explicit size for the overall application using the `-geometry` option (or resizing it with the window manager) will change the size of the editing window because *bitmap* will automatically adjust the size of each editing cell to fit.

However, this type of adjustment has limitations. The *bitmap* application defines a minimum size of 3 pixels for the cells in the bitmap editing area. This means that in the smallest *bitmap* window you can create, each pixel in the bitmap itself will be represented by a cell 3 pixels square. Even if this adjustment creates a window that fits on your screen, it is extremely difficult to edit individual pixels represented by cells 3 pixels square.

Since we have covered the OPEN LOOK bitmap editors, we do not discuss the MIT *bitmap* editor in detail here. Consult the client reference page in Part Three of this manual for more details on the *bitmap* client.

9.2.3 touchup (SunView only)

Touchup is a sophisticated "draw" program, rivalling some of the Macintosh draw programs. *Touchup* is a contributed software application that at present is only configured for the SunView system.†

However, it will therefore run on early versions of OpenWindows, and is interesting for several reasons. It can edit much larger bitmaps than can the three text-mode bitmap applications described above. It has a wide range of brush types built in, and allows you to define your own, including by picking part of an existing bitmap for use as a brush. And it lets you specify write the bitmap out with a different size than you read in, which is useful for cropping or fitting bitmaps. (This is not a scaling operation, but a cutting or padding operation.)

† SunView is Sun's popular pre-X11 window system. This program may have been ported to Sun's XView toolkit by the time you read this, which would make it available on any X platform. However, this is not guaranteed to happen.

Since it's not (yet) a true application of The X Window System, we don't have space to describe the program in detail in this book. Here is a sample screen, editing my world-famous "fishycat" bitmap (Copyright © by Ian F. Darwin):

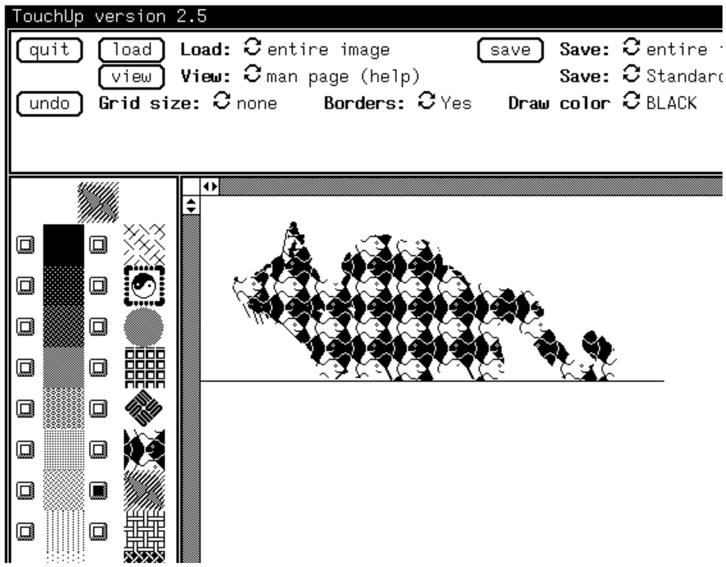


Figure 9-16. Touchup in action

Use of *touchup* will be obvious to anyone who's used a paint program; to anyone who hasn't, and wants to do bitmap editing, I recommend getting a copy of this program (see the Preface) and learning it.

9.2.4 Magnifying Portions of the Screen: xmag

Although it doesn't actually capture part of the screen for you, *xmag* is a useful little program for displaying part of the screen. The *xmag* client enables you to magnify a portion of the screen. The close-up look *xmag* affords can assist you in creating and editing bitmaps and other graphic images.

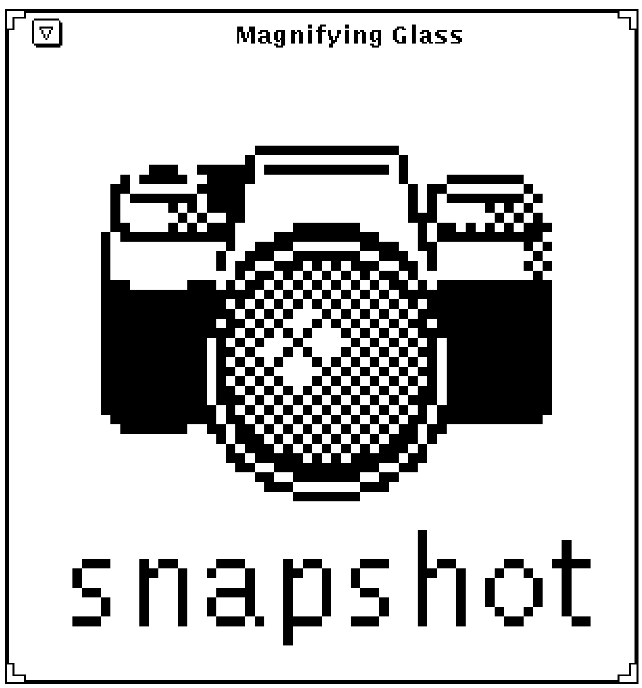
xmag is primarily a tool for application developers using sophisticated graphics programs. But you could also use *xmag* in concert with the *bitmap* client. For instance, say you're running a program that creates a special image on the root window and you'd like to create a *bitmap* file of a part of that image. You can display a magnification of the image you want with *xmag* and try to recreate the image by editing in an open *bitmap* window.

If you invoke *xmag* without options, you can interactively choose the area to be magnified (the *source* area) and position the magnified image on your screen. At the command line, type:

```
% xmag &
```

The pointer changes to a small cross (the cross-hair cursor) in the center of a small, hollow square with a wavering border. (By default, the square is 64 pixels on each side.) Move the cross-hair cursor, placing the square over the area you want to magnify, and click the first pointer button.

The hollow square becomes enlarged to the size of the magnified image. (By default, the image is magnified five times.) By default, *olwm* places the *xmag* window containing the magnified image in the upper-left quadrant of the display. Let's say we want to see the icon of *snapshot* blown up. Conveniently, *xmag*'s default viewing size is 64x64, the same as OpenWindows icons. Center the active area over the icon, and click any pointer button, and you should see a window similar to that shown in Figure 9-17



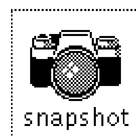


Figure 9-17. xmag window displaying magnified screen area.

The string "Magnifying Glass" will be displayed in the *xmag* window titlebar. This is the default title string of the application, and unlike most programs, *xmag* provides no way to change it.

The default size *xmag* window shows an area 64 pixels square, magnified five times. This magnification enables you to see the individual pixels, which are represented by squares of the same color as the corresponding pixels in the source image.

Rather than use the default source area and magnification, you can specify other values on the command line. See the *xmag* reference page in Part Three of this guide for a complete list of options. Quitting xmag

As a shortcut to quitting the *xmag* program, you can type q, Q, or Control-C in the *xmag* window.

9.2.4.1 What xmag Shows You

xmag enables you to determine the x and y coordinates, bitmap bit setting, and RGB color value of every pixel in the *xmag* window. (See Chapter 11, *Command-line Options*, for a discussion of the RGB color model.) If you move the pointer into the *xmag* window, the cursor becomes an arrow. Point the arrow at one of the magnified pixels and press and hold down the first pointer button. A banner across the top or bottom edge of the window displays information about the pixel, as shown in Figure 9-18.

I

Figure 9-18. Displaying pixel statistics with pointer in xmag window

The banner displays the following information about the specified pixel:

- The x and y coordinates relative to the window. The default *xmag* window is, in effect, a grid of 64 squares on each side. Therefore, each pixel has x,y coordinates between 0,0 and 63,63.

- The bitmap bit setting. This is either 0 if the pixel is in the background color or 1 if the pixel is in the foreground color.
- The RGB value. This is a 16-bit value. The RGB specification is in three parts (of four hexadecimal digits each), corresponding to the three primaries in the RGB color model.

If you are trying to create a graphic image on a grid (such as the *bitmap* client provides), the x and y coordinates of each pixel can be especially useful. Also, the 16-bit RGB value specifies the color of each pixel with moderate precision. Depending on the number of colors available on your display, you can learn to use RGB values to specify an enormous range of colors (other useful tools for learning RGB values are the colormap editor in *xv* (described above) and also *xcoloredit).*

xmag provides these pixel statistics dynamically. If you continue to hold down the first pointer button and drag the pointer across the window, the banner will display values for each pixel as the pointer indicates it.

9.2.4.2 Dynamically Choosing a Different Source Area

If you want to magnify another portion of the screen using the same source area size and magnification, you do not have to start *xmag* again. Simply move the pointer into the *xmag* window and click the ADJUST pointer button, or press the space bar. The magnified image disappears and the cursor again becomes a cross-hair surrounded by a hollow square. Move the cross-hair cursor, placing the square over the new source area you want to magnify, and click any pointer button. The magnified image is immediately displayed in the same location as the first image.

9.2.5 The Portable Bitmap Toolkit

Unlike the on-screen editing provided by the above bitmap editing programs, the Portable Bitmap Toolkit (in the user-contributed distribution) provides dozens of non-interactive utilities for converting graphics files to and from portable formats. Developed by Jef Poskanzer, the Toolkit is composed of four parts, three of which correspond to a particular portable format:

- PBM: utilities to convert files to and from portable bitmap format.
- PGM: utilities to convert files to and from portable graymap format (grayscale images).
- PPM: utilities to convert files to and from portable pixmap format (color images).

The fourth part of the toolkit, PNM, provides utilities to manipulate images in any of the three formats. For example, the program *pnmenlarge* enlarges a portable "anymap" by a factor you supply. *pnminvert* inverts an image in any of the three portable formats.

The available utilities and the conversions they perform are summarized in the *README* file in the source directory. Some representative conversion utilities and their functions are listed in Table 9-3.

Table 9-3. Some PBM Toolkit Conversion Utilities

Utility	Converts
giftoppm	GIF to portable pixmap.
ppmtogif	Portable pixmap to GIF.
ppmtoxwd	Portable pixmap to X11 window dump.
xwdtoppm	X10 or X11 window dump to portable pixmap.
ppmtopgm	Portable pixmap to portable graymap.
fstopgm	Usenix FaceSaver file to portable graymap.
pgmtops	Portable graymap to Encapsulated PostScript.
pgmtopbm	Portable graymap to portable bitmap.
pbmtomacp	Portable bitmap to MacPaint.
macptopbm	MacPaint to portable bitmap.
pbmtoxbm	Portable bitmap to X11 bitmap.
pbmtox10bm	Portable bitmap to X10 bitmap.
xbmtopbm	X10 or X11 bitmap to portable bitmap.
pbmtoxwd	portable bitmap to X11 window dump.
xwdtopbm	X10 or X11 window dump to portable bitmap.

As the table indicates, some of the available utilities come in pairs—they can be used to convert a file to a portable format and back to its original format again. (The table also includes a group of three related utilities to convert X10 and X11 bitmaps to portable bitmaps and back again.)

Certain conversions can only be performed in one direction. For example, you can convert a portable graymap to a portable bitmap (using *pgmtopbm*), but you can't convert a bitmap to a graymap. The one-way conversions generally involve changing a file to a simpler format.

You'll probably be most interested in converting graphics files to formats suitable for use with X, namely X11 bitmaps or window dump files. Keep in mind that a portable bitmap has a different format than an X11 bitmap. The program *pbmtoxbm* converts a portable bitmap to a bitmap compatible with X11.

The conversions you may want to perform can be simple (directly from one format to another) or complex (through several intermediate formats). An example of a simple conversion is changing a portable pixmap to a portable graymap using *ppmtopgm*:

```
% ppmtopgm pixmap > graymap
```

The PBM Toolkit source directory includes a file called *TIPS* that provides helpful hints on using the utilities. Based on these suggestions, we performed a fairly complex conversion: a Usenix FaceSaver image to a bitmap suitable for use with X. The following command performed the conversion on the file *myface* to create *myface.bitmap*:

```
fstopgm myface | pnmenlarge 3 | ppmscale -yscale 1.125 | ppmtopgm |\
            pgmnorm | pgmtopbm | pbmtoxbm > myface.bitmap
```

Notice that this particular conversion requires seven utilities! This procedure is by no means intuitive. We relied heavily on the *TIPS* provided.

The seven conversions performed are:

- Convert FaceSaver image to portable graymap (`fstopgm`).
- Enlarge a portable anymap three times (`pnmenlarge 3`).
- Scale pixels in y dimension; x dimension is adjusted accordingly (`ppmscale -yscale 1.125`). This program produces portable pixmap output.
- Convert portable pixmap to portable graymap (`ppmtopgm`).
- Normalize contrast of portable graymap (`pgmnorm`).
- Convert portable graymap to portable bitmap (`pgmtopbm`).
- Convert portable bitmap to X11 bitmap (`pbmtoxbm`).

Be aware that the command:

```
ppmscale -yscale 1.125
```

may not be necessary on all systems or the necessary arguments may vary. If you omit *ppmscale* and the command is necessary, the system should return a message to that effect and also tell you what arguments to use.

The possible uses of the PBM Toolkit programs and the ways in which they can be combined are extremely varied. You'll have to do some experimenting. To orient yourself, read the files *README*, *TIPS*, and *FORMATS* in the source directory. The source directory also includes reference pages for each utility.

9.3 Commercial Desktop Graphics Offerings

As The X Window System becomes more popular, the variety of commercial offerings available for it will continue to grow. This is especially true in the graphics area. Here are some current offerings.

9.3.1 Arts & Letters

A Sun OPEN LOOK version of this popular MS-Windows desktop graphics tool was released in 1991. It offers scaling, shading, freehand drawing with Bezier curves, vectorization ("auto tracing") of bitmaps, numerous special effects including fitting text to arbitrary shapes, and a library of some 5,000 clip-art images, with many more available. Contact the vendor: Computer Support Corporation, 15926 Midway Road, Dallas TX 75244, phone 214-661-8960.

9.3.2 IslandPaint, IslandDraw

Island Graphics distributes the Island Productivity tools, which includes IslandWrite, IslandDraw, and IslandPaint. The former is a publishing package, while the latter two are graphics packages that integrate with it. Contact the vendor: Island Graphics at 415-491-1000.

9.3.3 Artisan

Artisan is a graphics package for manipulating bitmap images, that is, a full-featured "Paint" program. Contact Media Logic at 213-453-7744.

9.3.4 Adobe Illustrator

A Motif version of Adobe Illustrator is available for Solaris 2.3 and later. Contact Adobe Systems, Inc. Their World Wide Web address is http://www.adobe.com.

9.3.5 Corel Draw

A Motif edition of this full-function desktop graphics package is available from Corel Systems.

9.4 PostScript Viewing and Editing

Many systems in use today support the PostScript™ language. It provides a largely device-independent imaging model that can be used by printers or displays to produce attractive text and graphics. It was initially made popular by the Apple Macintosh. Lately, with the rise of printers with reasonable resolution (600 dpi and up), PostScript has even begun to be used in some commercial graphics and typesetting.

Since a large number of Suns have "Sun LaserWriter" printers (a private label for the Apple LaserWriter), there is considerable support for PostScript in current versions of SunOS. In fact, many of the OpenWindows tools discussed in Chapter 7, *The OpenWindows DeskSet Clients* assume that they can print to a PostScript printer. And Sun provides a software product, NeWSprint† that provides PostScript support on printers that lack it, such as the common Hewlett-Packard LaserJet™ and even some dot-matrix printers. Because of the relative slowness of printers compared with graphics screens, however, it is common to want to "view" a PostScript file before committing it to paper. That is what

† NeWSprint runs only on SPARC systems running SunOS4.1 or later and OpenWindows.

PostScript Previewing is about. Let's look first at Sun's *pageview* previewer, and then at some other previewers.

9.4.1 Pageview (OpenWindows only)

Pageview is a PostScript viewer that depends on PostScript features of the server included with Sun's OpenWindows offering. As you'd expect, Openwindows versions up to 3.2 use the NeWS PostScript system, while 3.3 and later use Display PostScript. It will not work with other servers, such as the standard X11R6 server (users of such systems that need PostScript should see the section "Other PostScript Viewers" below). It should work with other servers that include the Display PostScript extention, however..

Pageview can be started from the *olwm* root menu, or from the command line. If it is invoked with no filename, it waits for you to provide a file. It displays this message:

```
Use File->Load... to load a PostScript file
 or
use File Manager to drag and drop one from $OPENWINHOME/demo/PostScript
```

As the message says, you can either use the *File* menu's *Load* item to load a specific file, or you can use OPEN LOOK's drag-and-drop mechanism to tell *pageview* what file to read. On the other hand, if *pageview* is invoked with a filename, it will read that file. As a special case for use in pipelines, it accepts the sporadically-enforced UNIX convention that a filename of a single minus sign ("-") means the standard input, so you can say

```
troff -ms myfile | troff_to_ps | pageview -
```

to run a *troff* and some troff-output-to-PostScript converter, and feed the result of that into *pageview*.†

As has been mentioned, the *File* menu has a *Load* option that lets you load one PostScript file at a time. It also has a *Print* option that lets you send either the current page or the whole document to a PostScript-capable printer.

Pageview's normal operation when it starts up is to display the first page of the document, and wait for you to do something. You can move around, if the page is bigger than the view window. However, the present version does not use scrollbars; you drag the canvas around by clicking and holding SELECT, then dragging the pointer in the direction you want to move the canvas. Or you can move to different pages by using the *View* menu, which lets you move to the First, Last, Next (default) or Previous page.

The *Edit* menu lets you edit either *PostScript* or *Properties*. The *Edit->Postscript* option lets you edit the PostScript input, assuming that you know the PostScript language well enough to make changes. The editing facility is identical with that used in *cmdtool* and *textedit*, which are described in Chapter 5, *The cmdtool/shelltool Terminal Emulator*. *Edit->Postscript* is also a great learning tool if you are teaching yourself the PostScript language. You can make a few changes and re-run the file, and see right away what effect your change has. For example, start up *pageview* with no options, and you will see this:

† Several companies provide versions of *troff* that produce PostScript; one such offering that works with Pageview is from SoftQuad Inc., 56 Aberfoyle Crescent, Toronto, Ontario CANADA M8X 2W4, phone 416-239-4801, 800-387-2777 (from U.S.A. only). E-mail mail@sq.com.

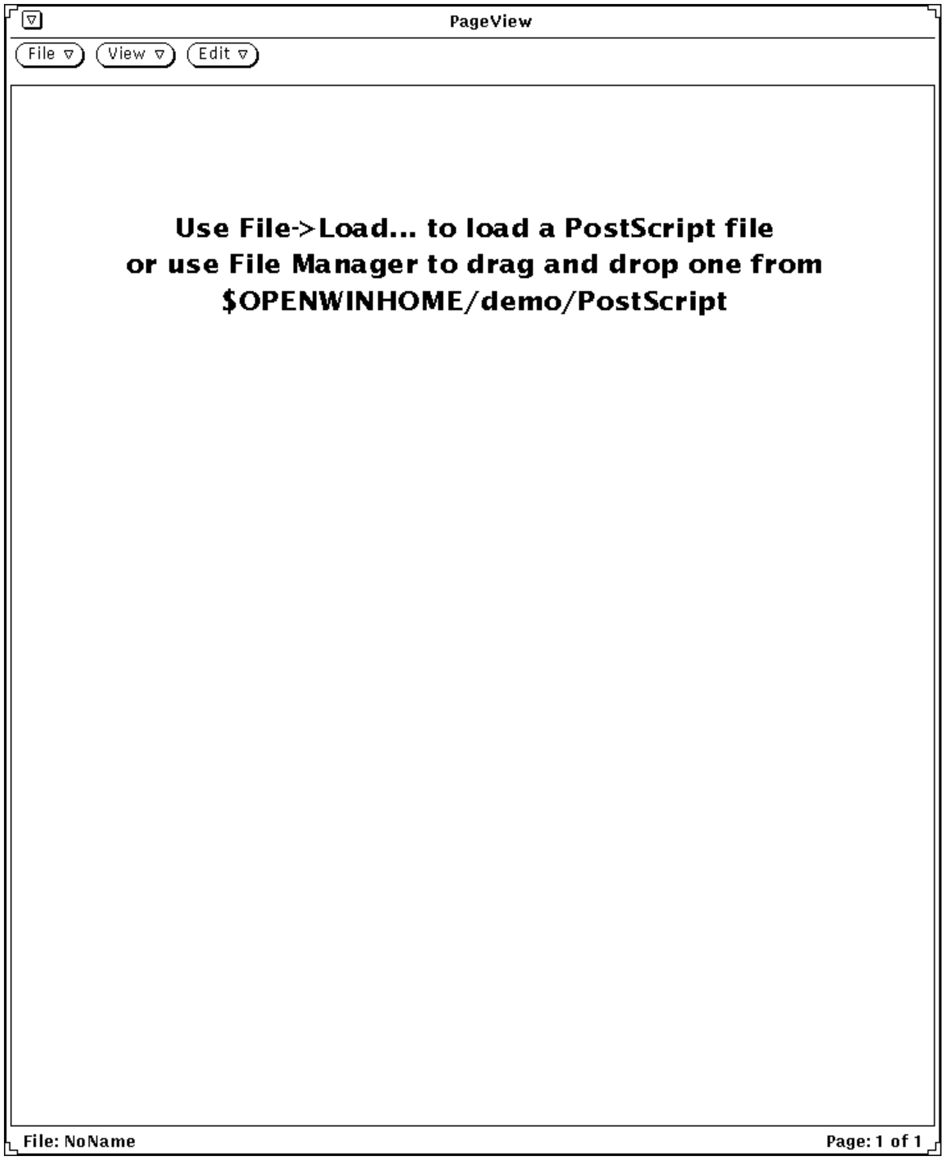


Figure 9-19. Pageview welcome screen

Just click SELECT on the *Edit* button, since *Edit->Postscript* is the default, and you will get an edit window:

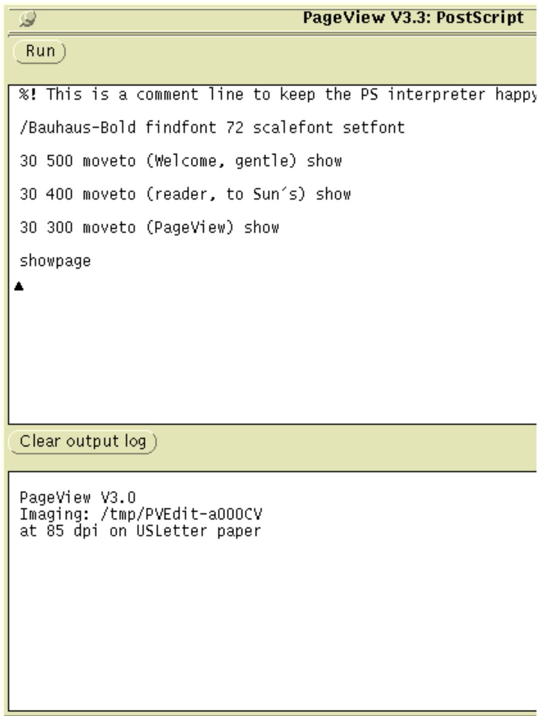


Figure 9-20. Pageview edit window

Type a few lines like those shown,† and click on *Run*. The main window will display the output:

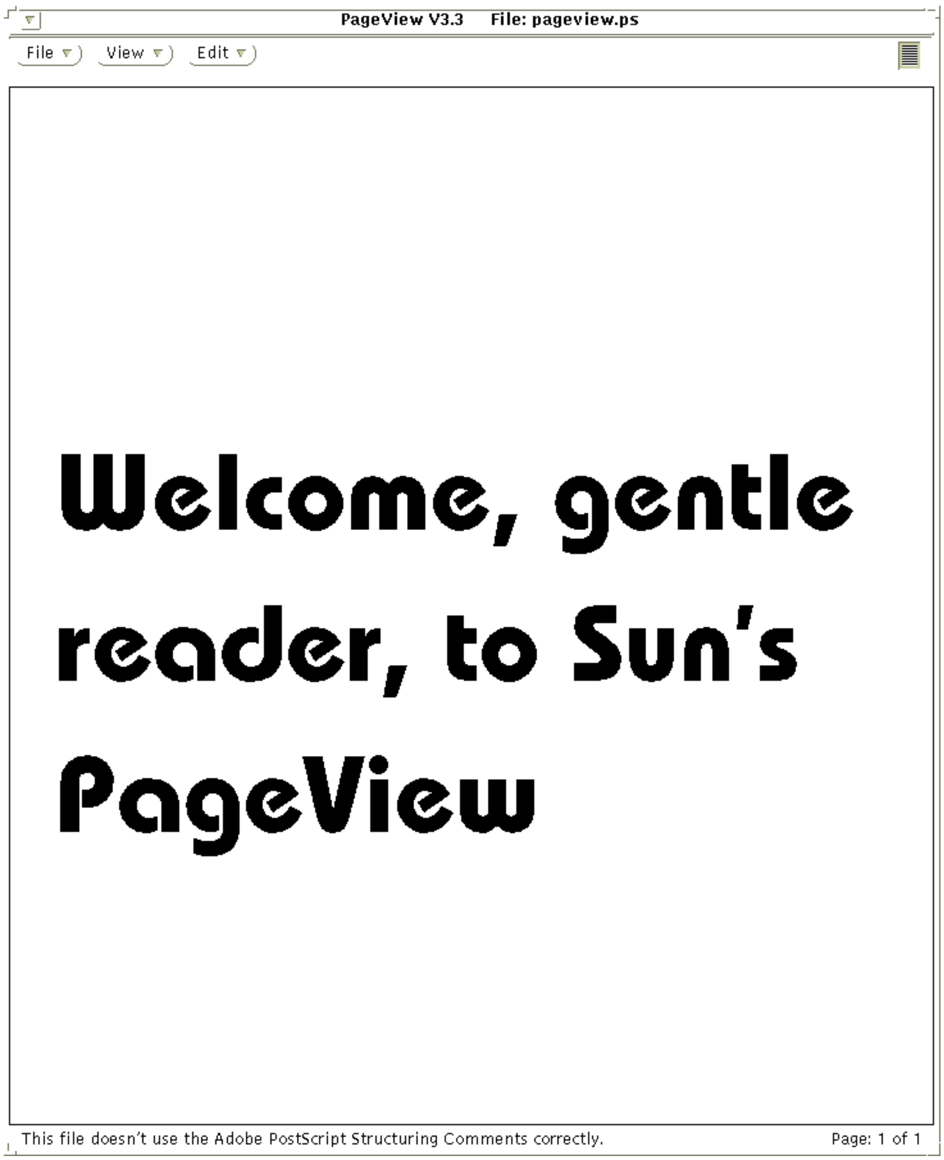


Figure 9-21. Output of pageview.

† Bauhaus is a commercial Type1 font; if you don't have it, use `Times-Roman` or some other font.

The *Edit->Properties* lets you change some settings, such as the size of paper, relative dots-per-inch, page orientation, etc. One useful mode is "anti-aliasing", which produces better quality type. These changes are not saved in a file; they only apply to the current *pageview* session; many of these can also be specified on the command line.

Pageview's command line options are detailed in the reference page in Part Three of this Guide. The most commonly-encountered difficulties with *pageview* in Release 2 of Open-Windows had to do with impure PostScript. There are several levels of "standardization" for PostScript. The base language has two versions (Version 1 is in use; Version 2 has recently been released by Adobe but isn't widely used yet). As well, "proper" PostScript documents include something called "structure comments", and in OpenWindows Release 2 and 3 the *pageview* program depended on them. In the current release of OpenWindows, *pageview* will work with or without the structure comments; if they are missing, it will read only the base PostScript language.

9.4.2 Other PostScript Viewers

Some system vendors include a PostScript interpreter called Display PostScript™ as part of their X servers. Display Postscript is in some ways patterned after the NeWS component of Sun's OpenWindows, but the two implementations of the PostScript language are, alas, not compatible. So you can't use the 2.2 or earlier *pageview* to display output on an IBM RS-6000 with AIX nor on a DECstation with Ultrix. However, these systems do provide their own PostScript viewers. Users of these systems should consult their vendor documentation.

In 1992, Sun Microsystems signed an agreement with Adobe Systems, the originators of the PostScript language and the vendors of Display Postscript, to make Display PostScript available in OpenWindows. Current releases of OpenWindows support Display PostScript rather than NeWS..

Finally, there are two programs in the "contributed software" category that can interpret PostScript and display it on an X11 window. These programs, *psview* and *ghost-scrip/gsviewt*, are included in the CD-ROM accompanying this book, and available from many Internet archive sites including UUNET (see description of UUNET in the Preface). We do not provide any further information, as these programs are still undergoing change.

9.5 Font Editing

Since this chapter is about graphics, it would be incomplete without some mention of one of the most basic graphics that we use in our daily lives, that of text fonts. X11 does not include any official font editing facilities. The contributed program *xfed* provides minimal facilities for producing bitmap fonts in the X11 "bitmap distribution format" (BDF fonts). BDF is an X Consortium standard, and any implementation of X11 is supposed to either accept them (directly, as the DECwindows™ server does), by conversion using *bdftosnf* (as the MIT X11 servers do), or by conversion to its own format (as the older OpenWin-dows server does, using *convertfont*). Fonts in this format can therefore be used on any reasonable implementation of The X Window System. More detail on these and other font conversion programs can be found in Volume Eight, *X Window System Administrator's Guide*.

The Sun operating systems (SunOS 4 only) includes *fontedit,* a bitmap editor originally written for SunView™, and unfortunately not yet (as of OpenWindows Version 3) converted to run on X11 for OpenWindows. It, too, provides simple editing of bitmap fonts. In usage, both *xfed* and *fontedit* are similar in flavor to *iconedit* or *bitmap,* but with extra controls for editing the font information. Figure 9-22 is a picture of the (SunView version of the) *fontedit* program in action

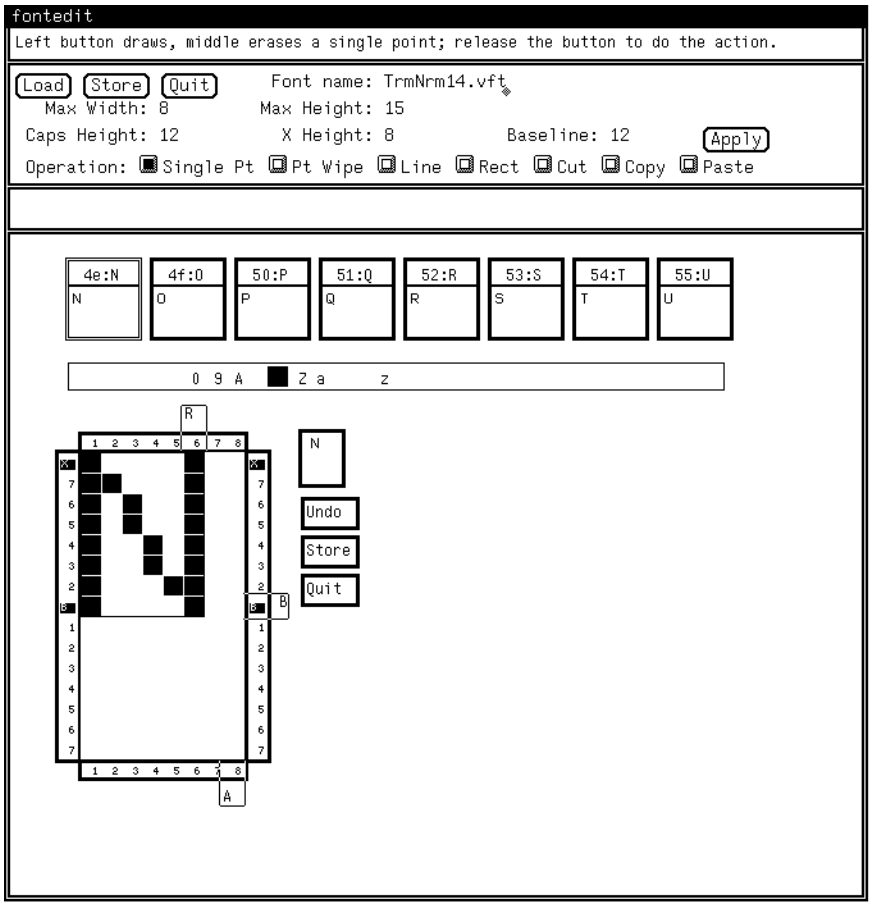


Figure 9-22. SunView fontedit.

For commercial production of fonts, there are several schemes in use. Most of the large commercial type foundries either have their own software, or use a package called *Ikarus* (from URW Software AG). *Ikarus* lets you edit outline fonts, and produce either scalable fonts or bitmaps in a variety of sizes and formats. As well, many companies are now producing software that helps you generate PostScript™ fonts. OpenWindows users who wish to pursue this should investigate Sun's *Typemaker*™ offering, which makes fonts in an outline format ("F3") that the OpenWindows server can use. SunPics produces a CD-ROM called "Printer's Palette" that contains a variety of fonts and other tools for use with OpenWindows and its printing software package NeWSprint (described in Chapter 7, *The OpenWindows DeskSet Clients*).

Current versions of The X Window System (including OpenWindows versions that use Display PostScript) support the use of "Type 1" PostScript fonts. These can be created using tools that are popular on the Macintosh, such as *Fontographer* and *Font Studio*. These could be run under Apple's Macintosh Application Environment under Solaris 2, or you could use them on a Mac and convert the font files to "MS Windows" format and install them on your OpenWindows server, as discussed in Chapter 10, *X11, OPEN LOOK and OpenWindows Font Specification*.