JPEGView

Version 3.3, May, 1994
Written by Aaron Giles

Aaron Giles

Table of Contents

Introduction

In simplest terms, JPEGView is a fast, flexible postcardware utility for the Macintosh designed to provide simple, high-quality viewing of PICT, JFIF, GIF, TIFF, BMP, MacPaint, and Startup Screen images.

JPEGView is *not* an all-purpose format conversion utility; however, it does support a save function to allow QuickTime previews and custom color icons to be added to images of any format. It can also perform the very simple translation between JPEG-compressed PICT files and the standard JPEG interchange format JFIF.

What's new in JPEGView 3.3

JPEGView 3.3 has been designed to fix several problems that were found in the version 3.2 release, and to add a few more interesting bits to JPEGView's functionality. Specifically, here's an overview of what's new, and in which chapter the associated changes are reflected:

Improved JPEG support. JPEGView started its life as a QuickTime-dependent JPEG viewer, and has since evolved into something much more. Beginning with version 3.3, QuickTime is no longer even needed to view JPEG images, as JPEGView incorporates the Independent JPEG Group's code for decoding JPEGs. The new code is much more robust than QuickTime, but significantly slower on 680x0 Macintoshes (and about the same speed as QuickTime on a Power Macintosh). Thus, you can trade off speed for robustness by choosing not to use QuickTime for JPEG images; this choice is controlled from the last panel in the Preferences window (*Preferences Settings*).

Better icon creation. This latest version of JPEGView now supports four distinct icon styles, so you can create your icons in your favorite style; this choice is made in the "Saving" panel of the Preferences window. In addition, you can now select a smaller area of the image to be represented by the icon, thus ensuring that you can accurately distinguish your files from within the Finder. (*Preferences Settings*, *Saving Images*).

Faster drawing. A new scaling algorithm has been incorporated into JPEGView's High quality drawing, resulting in a 25-75% speed increase when using this drawing mode. Image quality under the new algorithm is slightly worse than in previous versions, so the default quality is now Very High, to ensure the best quality drawing. (*Colors and Color Reduction*).

Drag and drop support. If you have the *Macintosh Drag and Drop* extension installed in your System Folder (or if you are running System 7.5 or later), JPEGView can now take advantage of it in several ways. First, you can drag a selection made in JPEGView outside of its original window

and drop it into any other application which supports dropped pictures. This is similar to copying the selected portion of the image in JPEGView and pasting into another application, only much faster. You can also drag the text from JPEGView's Statistics and Comments floating windows into another application. Finally, when using Finder 7.1.2 or later, you can drag a folder from the desktop into the Slide Show Options window to choose that folder for a slide show. (*The How and Why of Floating Windows*, *Special Effects*, *Slide Show*).

More robust memory management. The tricks that JPEGView uses to reign in QuickTime's use of memory on systems running with virtual memory or with Connectix's *RAMDoubler* product have been improved. Previous versions of JPEGView would sometimes incorrectly report corrupt images or prevent certain system extensions from working. The memory management in JPEGView 3.3 should fix most if not all of these problems.

Additional minor features. Cropped images of any format are now allowed to be saved from JPEGView, and their cropping rectangles will be remembered in the future (Saving Images). A redesigned "marching ants" animation technique allows selections to be seen much more clearly. PICTs saved at other than 72dpi are always opened to their full resolution. And a new version of the *JPEGView JFIF Preview* extension works more smoothly as a fat binary.

If you have been using JPEGView 3.0 or later, you should have no trouble adjusting to the new features in 3.3. If you have used any versions of JPEGView prior to 3.0, it is highly recommended that you familiarize yourself with the operation of the new features by reviewing this updated documentation carefully. The most important new features are described in the *Scripting JPEGView*, *Colors and Color Reduction*, and *Slide Show* chapters.

Postcards, registration, and licensing

Once again, I am releasing JPEGView to individual users as postcardware, meaning that if you use it, even casually, I'd really like it if you sent me a real "snail-mail" postcard of some sort—and no excuses! Even if you live next door to me, or have sent me an email message, please take the time to drop me a real pen and ink postcard anyway. I guarantee you that the dollar or so that you spend is one of the best software bargains you'll find anywhere—and I'd really love to have tangible proof that you're using JPEGView!

Any user of JPEGView 3.0 or later also has the option of becoming a fully registered user. Registered users who send in a one-time registration fee of US$20 receive a printed, bound copy of the full JPEGView documentation, along with an official JPEGView 3.3 release disk, containing the complete JPEGView 3.3 release plus a few of my favorite JPEG images. My current U.S. Mail address can be found in the chapter *Contacting the Author*, later on in this documentation.

Overseas users who wish to avoid the hassle and fees of international money orders are asked to simply send the equivalent of US$20 in cash, plus about US$3 extra for postage (everything is sent airmail). Although this is not recommended by the post office, you can usually get away with it as long as you do well to mask the fact that there is cash enclosed, i.e., make sure it is wrapped in sufficiently opaque paper.

Note that registration is *optional*; the only "payment" that is required for individual use of JPEGView is a postcard.

Finally, a very simple site license is available for universities and companies that wish to make JPEGView 3.3 an officially supported product—and it typically only costs you the low low price of a few individual registrations (US$50 to be exact)! Contact me via email

(giles@med.cornell.edu) or U.S. Mail (in *Contacting the Author*) for full details, or just go ahead and send in the cash (most people seem to do it anyway!) Purchase orders are welcome.

Organizations and individuals who are interested in distributing JPEGView with their products should contact me first before doing so. I'm usually pretty flexible about licensing, especially for "good causes", so if you're interested, please don't hesitate to get in touch with me about it!

Legal mumbo jumbo

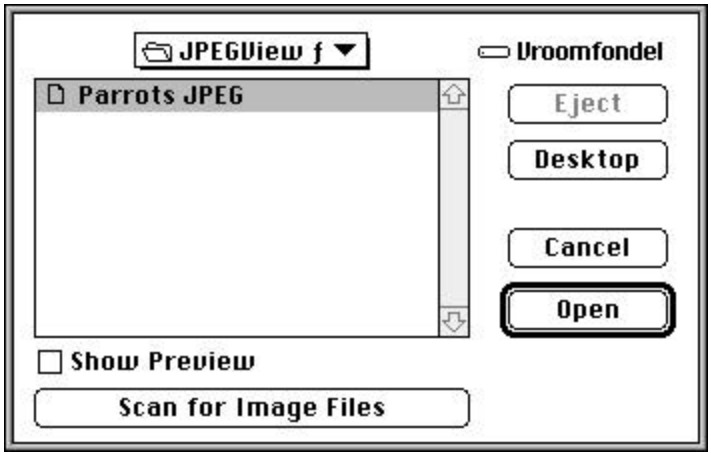
Viewing Images

At its heart, JPEGView is a simple image viewer, designed to provide a high-quality screen image in as little time and with as little fuss as possible. Opening images for viewing is as simple as with most any other Macintosh application: select the **Open** menu item from the **File** menu (or type O), choose your image file in the dialog box, and let JPEGView do the rest. Closing an image just requires clicking in the close box or choosing **Close** from the **File** menu (W). For its basic operations, JPEGView works just as you'd expect.

Beyond these simple operations, however, JPEGView supports a rich variety of features which allow even easier navigation through your image files. This chapter describes how these features work in greater detail, under JPEGView's normal, or "default" settings. Like many Macintosh applications, the exact way JPEGView works can be changed by altering the settings presented in the Preferences dialog box, available under the **Preferences** item in the **File** menu. For a precise description of how these changes affect JPEGView's operation, see the chapter on *Preferences Settings.*

Opening images, with and without previews

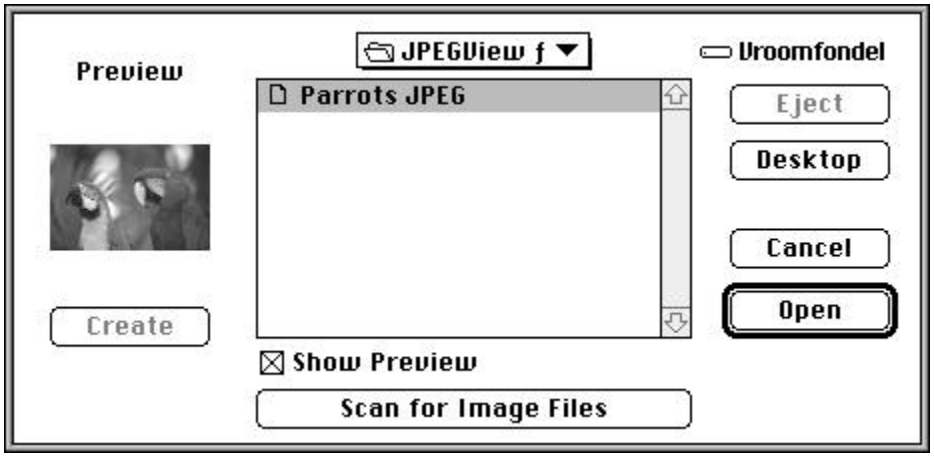
When you choose **Open** from the **File** menu, what you get back is not the usual dialog you're used to. Fortunately, all the basic features are still there, allowing you to select a file to open in the usual manner. But JPEGView offers you two important additions: the "Show Preview" checkbox and the "Scan for Image Files" button.



The standard Open dialog, with JPEGView's and QuickTime's additions.

If you've used QuickTime applications before, you have undoubtedly encountered the "Show Preview" option in your Open dialogs. The principle is simple: in certain types of files which QuickTime understands, a preview (a.k.a. thumbnail) image can be stored along with the original image data. This preview image is simply a small (usually about 80 pixels square) representation of the full image contained in the file. This allows you to see in advance what you will get once you open the image file, before having to actually open it and wait through the full decompression of the image inside.

To turn the previewing feature on, simply check the "Show Preview" box beneath the file list. In response, the Open dialog will expand horizontally to include a new Preview area, where these small preview images will be displayed:



Check the "Show Preview" box to expand the Open dialog box for displaying previews.

Once you have done this, you can view the preview stored in an image file with just a single click on the name of the file in the list. If the file you select contains a preview image, it will be read from disk and displayed in the Preview area after a second or so.

If the "Show Preview" box is checked and you highlight an image file which has no preview, the "Create" button beneath the Preview area will become active, allowing you to automatically create a preview for that image inside the Open dialog. Occasionally when you select an image file, the "Create" button will change to an "Update" button and become active; this simply means that the preview stored with the image isn't up-to-date. To remedy the situation, click on the button to create an updated preview.

Finally, if you do not wish to see previews anymore—they can be slow at times, especially from slower disks like floppies and CD-ROMs—you can always uncheck the "Show Preview" box, which will return the Open dialog to its normal, skinnier size. QuickTime handles the job of remembering whether you checked this box the last time you used it, so that the next time you run a QuickTime application (including JPEGView), this option will be left as you last set it.

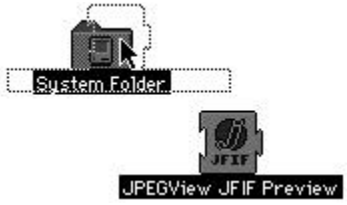
The JPEGView JFIF Preview extension

Preview images are very nice, but unfortunately not all previews are created equally. JPEGView, together with QuickTime, provides support for viewing preview images for almost all image types automatically, without any special mechanisms. However, there is one limitation, and one exception to this feature.

The limitation is that although JPEGView can create previews when saving any of its supported image formats, QuickTime only knows how to create previews for PICT images; this means that the "Create" button under the Preview area will always be dimmed when a non-PICT image is selected.

The exception to free preview images is the JFIF file format, which has a preview format all its own. In order to display these special previews, you will need to install the *JPEGView JFIF Preview* extension, which ships with this release of JPEGView.

To install the *JPEGView JFIF Preview* extension on your Macintosh, simply drag its icon on top of your System Folder icon, or manually place it in the Extensions folder within your System Folder. You will need to restart your computer to get QuickTime to notice the new extension, after which you should be able to see JFIF previews effortlessly, even from within applications other than JPEGView.



Drag the JPEGView JFIF Preview icon on top of your System Folder icon to install it.

It is important to realize two quirks about the way this extension works. First, you will notice that the "Create" button under the preview is always dimmed; this is simply because the *JPEGView JFIF Preview* extension was not designed to allow you to create previews from within the open dialog. To create JFIF previews, you need to open the image and save it back again with the preview option checked in the Save dialog. This technique works equally well with other non-PICT images; see *Saving Images* for a full description of this procedure.

The second quirk relating to the *JPEGView JFIF Preview* extension is that it will not display an icon at the bottom of your screen during startup. However, don't worry—as long as it's in your Extensions folder, you can be fairly certain that it has, in fact, loaded successfully.

How to see images that don't appear in the file list

Sometimes you may receive a file that JPEGView doesn't immediately recognize as a valid image file. This is especially true if you have copied an image from a BBS or from another type of machine. In order to draw up the list of files as quickly as possible, JPEGView initially checks only the Finder's file type information. Although this works quickly, it is possible (and sometimes inevitable) that the Finder's information about a file is incorrect. For example, when you double click on such an image, the Finder doesn't know what to do with it, or which application it should use to open the file with.

So how do you get JPEGView to recognize these other files? This is where the "Scan For Image Files" button comes into play. Clicking on this button does two things: first, it checks all the files in the folder you are currently viewing for valid images. In order to make sure the Finder isn't lying, JPEGView actually opens each file and peeks inside at the contents. This works well, but be forewarned: if there are a lot of files, it can take some time!

Use the "Scan for Image Files" button to find incorrectly-typed files from within the Open dialog.

After it has finished checking all the files, JPEGView then goes through the list and fixes the Finder's information about these files. This means that in the future, the files that were found will appear in the Open dialog without any further intervention on your part. Plus, when you double click on these "fixed" files in the Finder, JPEGView will automatically load and open them for you. (If you have the folder visible on the desktop, you will see the icons for these files change to JPEGView icons after several seconds.)

Unfortunately, certain graphics formats do not lend themselves well to automatic identification in this manner. Specifically, it is impossible to guarantee successful identification of MacPaint and Startup Screen files, apart from what the Finder says about them. For this reason, JPEGView requires that the Finder's information properly identify such files; incorrectly-typed files in these formats will give you an error if you attempt to open them.

If you think that you might have a MacPaint or Startup Screen image with the wrong Finder file type, drag and drop its document icon onto the *MacPaint AutoTyper* or the *Startup Screen AutoTyper* application as appropriate. This will force the Finder information to properly identify the file, and may allow JPEGView to open it successfully. For an example of how to use the AutoTyper utilities, see the chapter *File Formats and File Types.*

Under certain circumstances, it is possible that the Finder's file type information cannot be changed. For example, if your incorrectly identified files are all on CD-ROM, there is no way to modify the information contained on the disc. To allow you to view such files, JPEGView supports an **Import** command, also available under the **File** menu (or by holding down the Shift key as you type O):

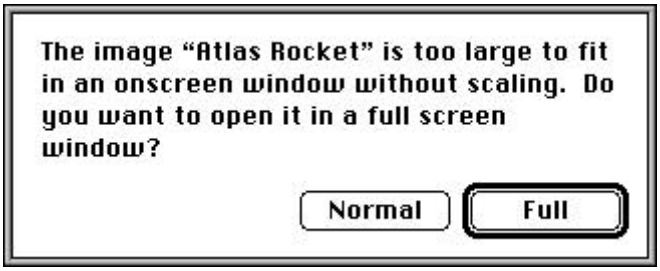
The import menu item allows you to open any type of file, regardless of what the Finder says about it.

This will give you a dialog identical to the Open dialog, only showing every file available, regardless of what the Finder has to say about it. Of course, JPEGView can still only open image files it knows about, but at least importing them means that they are all visible to you.

What happens to the images when they're opened

Once you've chosen an image file in the Open dialog, JPEGView loads the file into memory and immediately attempts to verify that the file does in fact represent a valid image (remember that the Finder isn't always correct!) If the image passes this test, but its Finder information was incorrect, JPEGView will quietly fix this for you, to help keep everything in order.

At this point, JPEGView also checks the size of the image you opened, comparing it to the size of your preferred monitor. If the image is too large to fit vertically on your screen, JPEGView will ask you if you wish to open the image in a full screen window:



When appropriate, JPEGView will ask if you want to open a given image in a full-screen window.

The advantage of full screen windows is that they have no title bar and hide the menu bar, giving you a little extra height in which to view the image. The *Special Effects* chapter describes how JPEGView's full screen windows work more comprehensively.

If the image you have opened is a high-color image (with thousands or millions of colors) and your screen is only capable of displaying 256 or 16 colors, JPEGView will then perform two-pass color reduction on the image to find the best set of colors for display. Because this process requires JPEGView to decompress the image, a progress window is displayed to let you know how far along JPEGView is:

A progress window is displayed during two-pass color reduction; the operation can be cancelled by clicking on the "Cancel" button, or by pressing Esc or by typing a "." (period) while holding down the key.

For more information on exactly what two-pass color reduction is and how it helps image quality, see the chapter on *Colors and Color Reduction*.

Once everything is ready, a new window is opened on your desktop, and the image is decompressed. During the decompression process, the bottom right-hand corner of the image's window will display a progress message, keeping you informed of how far along JPEGView is in decompressing the image data. Additionally, JPEGView indicates when it is performing a scaling or dithering operation by changing the progress message from "Decompressing…" to read "Scaling…" or "Dithering…" as appropriate.

Drawing progress is displayed as a percentage in the lower-right corner of a window.

An important distinction between JPEGView and nearly every other image viewer available is the fact that JPEGView automatically shrinks large images so that they fit on your screen without any distortion. This is important not only because it allows you to always see the full image at one glance but also because you can easily select the area of interest when you do decide to zoom in and look at the details. It also means that you don't lose the extra area taken up by scroll bars, which is an important consideration for those with monitors smaller than 14".

The amount of time it takes to decompress a given image can vary widely, from under a second for a small uncompressed image to over three minutes for a huge JPEG-compressed image. JPEGView has been designed to give you nearly optimal speed for all compression methods. If the wait becomes too tedious, however, you always have the option to abort the drawing proce-

dure by pressing either the Esc key on your keyboard, or by typing a period while holding down the key. When you abort the drawing in this manner, JPEGView will fill in the remaining area with a pattern of X's, to let you know why it wasn't drawn.

A pattern of X's fills the remaining area when you abort the drawing.

Fast and slow window updating

Memory requirements for most image viewers are stiff, usually on the order of several megabytes for even reasonably sized JPEG files. This is because they require you to give them enough memory to decompress the whole image at once and store it offscreen for future use. While this may be simple and fast for scrolling around, if you try opening more than a couple of decent sized images you may be wondering why 20 megabytes of RAM doesn't seem like so much anymore.

Once again, JPEGView takes a different approach. Internally, the workings of JPEGView's memory management are a little complicated, but on the surface all you see is that some windows will redraw themselves immediately, while others need to have their images decompressed before display. This is known as *fast updating* and *slow updating*, respectively. Here's how it all works:

When JPEGView first loads an image, it attempts to create an offscreen copy (called an offscreen bitmap) of what you see on the screen. It is important to realize that this offscreen bitmap need only be the size of the image you see on the screen, and not the size of the original image. This means that the amount of memory needed for the bitmap is only dependent on how large your screen is, and not on how large the original image was, which is precisely what most other image viewers *require* of you.

If it turns out that there isn't enough memory available to make an offscreen bitmap for an image, it's all right; JPEGView can do without it. But if there is enough memory, this offscreen bitmap allows JPEGView to redraw obscured portions of that image's window instantly. This is known as fast updating and can only be achieved if there is an offscreen copy of what you see on the screen available in memory.

Notice that JPEGView does not require this offscreen bitmap. This is important, because when JPEGView realizes that it needs more memory, it will cheerfully get rid of enough of these offscreen bitmaps to make room for other, more important things. If you later close other images and free up more memory, JPEGView will quietly re-create these offscreen bitmaps; until then, such images are resigned to using slow updating, which means they have to be decompressed to be displayed again. When part of such a window needs to be updated, it is filled in with a sparse pattern of dots until JPEGView decides there is time to do the slow update.

This pattern of dots is used to indicate a pending slow window update.

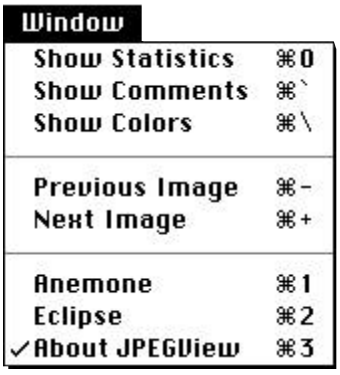
Of course, all this happens behind your back, so you never need to worry about it explicitly. What you actually see is this fast and slow updating. Fast updating always occurs immediately, whereas slow updating—because at times it can be really slow—is postponed until later, so that if you're in a rush and know what you want to do, you do not need to wait for it to happen. JPEGView determines this by waiting until you leave it alone for five seconds. Once you do this,

it will begin doing all its slow updating one window at a time, from frontmost to backmost, waiting two and a half seconds between each to let you do something else.

It's a simple matter to see the difference between the two types of updates: fast updates immediately replace the obscured portion of the window, while slow updates paint the obscured area with a sparse pattern of dots until a full update is done. This lets you know that JPEGView is not ignoring the obscured area, only waiting a few seconds before updating it. Even if a slow update is in progress, you always have the option of stopping the decompression either by pressing the Esc key, or by typing a period while holding down the key.

Switching between JPEGView's windows

With JPEGView's sparse memory requirements, it is possible to have many images open at once, even with only a megabyte of RAM available. In order to facilitate movement between these different windows, JPEGView has several useful windowing commands, as well as a slide show option—described later in the *Slide Show* chapter—which can cycle through all the images in memory. With the exception of the slide show, all of JPEGView's window navigation commands can be found in the **Window** menu.



*The **Window** menu lets you quickly switch between any of JPEGView's open windows.*

The most direct way to bring a given image's window to the front (apart from clicking on the window itself) is by selecting it from the **Window** menu. As you open images in JPEGView, their window titles are added to the bottom of the **Window** menu for immediate access to every image currently in memory. Additionally, the first nine windows added to this list are assigned to the keyboard shortcuts 1 through 9, for even faster switching. You can quickly identify which window is currently active by looking for the entry in the list with a checkmark beside it.

In addition to an entry for each open window, the **Window** menu contains the **Next Image** (+) and **Previous Image** (–) items. With these two options you can cycle forward and backward through the list of open images. Finally, the topmost items in the **Window** menu control the visibility of JPEGView's three floating windows (sometimes called "windoids"). Floating windows are a special type of window whose behavior is independent of the other standard windows. See the chapter *The How and Why of Floating Windows* for a description of how they work and what information they convey.

Putting things away

Once you've mastered the art of window navigation, getting rid of the images seems a rather simple exercise. As you would expect, the close box on the window, the **Close** item in the **File**

menu, and the W shortcut all close the active window, releasing any memory used by the associated image. Further, the convenient **Close All Images** option (Shift-W) quickly closes up all open image windows—but not the Help window, the options dialogs, or any visible floating windows. This allows you to keep those windows open and available at all times, even letting you close all the image windows while they are active.

*Use the **Close All Images** command to put away all currently open images at once.*

You can also close several open windows at once by holding down the Option key as you click in the close box on one of the windows. If you option-click the close box of a floating window, JPEGView hides all floating windows; doing the same thing in a standard window closes all open standard windows (this time including the Help window and the options dialogs).

Printing images

In addition to all of this, JPEGView also supports a rather simple method for obtaining a hard-copy print of an image, assuming your printer properly handles printing of color images. If you're not sure whether your printer provides this support, simply try it out and see how the resulting image comes out. Either you'll like it or you won't; if you do, you're in luck. If not, c'est la vie.

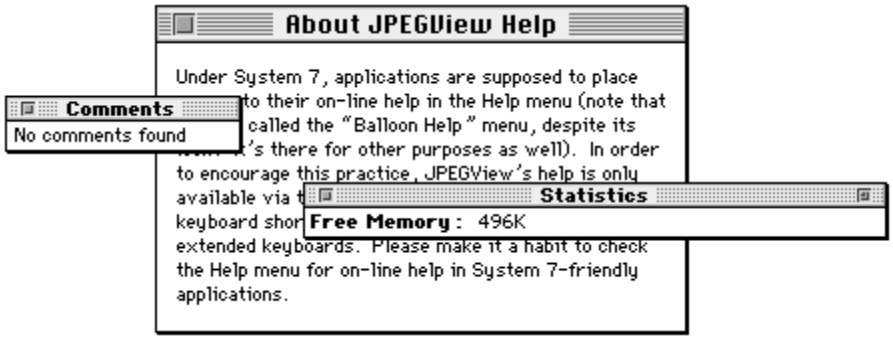
Before printing an image, you will likely want to choose **Page Setup** from the **File** menu to set the paper direction (portrait or landscape) and other options. Once set, these options are saved and remembered until you change them again. From there, you can go ahead and choose **Print** (P) from the **File** menu; this will give you some more options, depending on which type of printer you are working with, and then let you start the ball rolling.

*Choose **Print** from the **File** menu to print the image in the frontmost window.*

JPEGView then puts up its familiar progress dialog, telling you how far along it is in printing the image, and proceeds to dump the image to the printer. Note that at this time, you have no control over what size the resulting image is printed at. JPEGView always scales the image to fit the current paper size, with half-inch margins on all four sides. Thus, printing an image from JPEGView will always give you a single-page printout.

The How and Why of Floating Windows

In the past few years it has become quite fashionable—especially in imaging applications—to support a new type of window whose behavior breaks all the rules of how windows were originally supposed to operate on the Macintosh: the floating window (sometimes called a "windoid"). Floating windows always appear in front of standard windows, so that you can always see what information they convey; yet their presence in the foreground does not prevent operations from affecting the active standard window behind them. For the purposes of everything but visibility, they are effectively nonexistent.



An example of two floating windows floating above a standard window. Note that the title bar of the standard window indicates that it is still the active window, even though it is visually behind the floating windows.

The onscreen appearance of a floating window is similar to that of a standard window, except for the floating window's title bar, which is smaller and highlighted in a different manner. In general, floating windows use the terminology *show* and *hide*, rather than the more familiar *open* and *close*. This is reflected in the menu items relating to them: for instance, you can make the Statistics floating window visible by choosing the menu item labeled **Show Statistics** from the **Window** menu; similarly, you can hide a floating window by choosing the **Hide Statistics** menu item from the same menu (it is in fact the same menu item, with the name changed as appropriate).

The menu item for a visible floating window (here, the Statistics window) is changed to read "Hide".

When you make a floating window visible, one of the first things you'll notice is that title bar of the topmost standard window is still highlighted, despite the existence of the floating window

above it. This serves to remind you that the standard window is still really the "active" window and that all actions chosen from the menus will apply to the standard window and not to any of the floating windows. An important first example is that choosing **Close** from the **File** menu (or typing W) closes the active standard window, not the floating window. To hide (close) a floating window, you either choose **Hide *Window*** (where *Window* stands for the name of the floating window) from the **Window** menu, or click in the floating window's close box.

Another important fact about floating windows is that they can never be manipulated such that they end up behind a standard window. That is, clicking in a standard window will bring it in front of all other standard windows, but it will still appear behind any floating windows that are visible. Along the same lines, you can also click and drag to make a selection in the topmost standard window, without affecting the floating windows. At first, all this may seem a little disconcerting, but give it time and you will soon realize how useful floating windows can be.

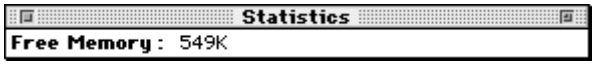
Finally, the position and visibility of each floating window is saved and remembered whenever you quit JPEGView, and is restored when JPEGView starts again. This means that you can position your floating windows and choose which ones to show, knowing that they will be visible in the same positions next time you run JPEGView.

JPEGView currently supports three floating windows: the Statistics floating window, which displays information about the active image window; the Comments floating window, which is used to hold any comments extracted from the image in the active window; and the Colors floating window, which shows the current color selection on whichever monitor it is displayed. The slide show controls window is also a floating window; a description of its operation can be found in the *Slide Show* chapter.

The Statistics floating window

JPEGView's Statistics floating window displays a host of useful information about the image currently displayed in the active window. To see this information, make the Statistics floating window visible by selecting **Show Statistics** from the **Window** menu (0). Once visible, this floating window gives you a frequently updated list of many important bits of information about the currently active image and about how it is being displayed. Below is a summary of what each field describes and what you might find displayed there.

If the list of data presented seems a little too extensive for your purposes, you can shrink it down to a one-line summary by clicking in the zoom box on the right side of the title bar. The format of the summary takes one of two formats, depending on whether you have any open images. If there are currently no open images in JPEGView, the reduced Statistics floating window looks like this:



The one-line Statistics floating window when no images are open.

This version simply displays the amount of memory available for opening images. Once you open an image, the small Statistics floating window changes to display a one-line summary of information about the frontmost image, for example:

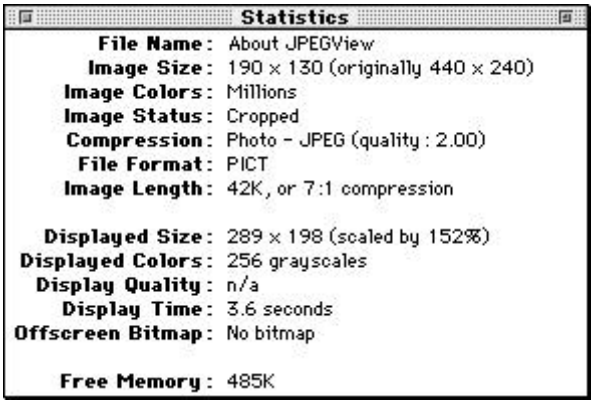
An example of the one-line Statistics displaying a summary of the frontmost open image.

From this example, you can see that the summary displays the name of the image, in boldface, followed by the compressed data size (in kilobytes), the file format (PICT, JPEG / JFIF, GIF, etc.), the width, the height, and the number of colors stored in the image. Since all this information is given in the full statistics display, you can read in more detail about exactly what everything means below.

Also, if you are running on a drag-aware system (System 7.5, or an earlier System 7 release with the *Macintosh Drag and Drop* extension installed), you can take advantage of the drag-and-drop functionality to drag the statistics information as text into another application. Simply click and drag on the text in the Statistics floating window, and drop the resulting text into your favorite drag-aware text processor.

Statistics for the original image

Here is what the full-sized Statistics floating window looks like:



The full Statistics window, displaying information for the About JPEGView image.

Each line is described in further detail below.

File Name: This is simply the name of the original file from which the image was read.

Image Size: This tells you the width and height of the original image in pixels, before any scaling performed by JPEGView. For cropped images, this field tells you the size of the cropped region, with the original size given in parentheses immediately afterwards.

Image Colors: This is the number of colors described by the image data, given in the same way you're used to seeing it in the Monitors Control Panel, i.e., "Thousands" means 32,768 and "Millions" means 16.7 million.

Image Status: This field tells you the "status" of the image, which can be a combination of:

- *Aborted*: the drawing of the image was interrupted (using the Esc key or the . combination).
- *Banded*: the image was compressed using QuickTime under low memory conditions. This means that the full image actually consists of several smaller images, each of which is a horizontal strip of the full image.
- *Corrupt*: the compressed image data was found to be damaged in some form or another. Portions of the image may still be decipherable.

- *Cropped*: the image is actually only part of a larger image; the full image can be recovered using the **Uncrop Image** function, described in the *Special Effects* chapter.

Compression: This field tells you the type of compression used on the image and, if the information is available, the vendor of the compressor and the quality factor used in the compression. There are a number of possible values that can be found here:

- *JPEG (QuickTime, Adobe,* or *I.J.G.)* for JPEG images compressed by QuickTime, Adobe, or the Independent JPEG Group's software.
- *GIF/LZW* for images compressed using GIF's variant of Lempel-Ziv-Welch compression.
- *Modified CCITT RLE* for TIFF images compressed using a variant of CCITT Group 3 fax compression.
- *PackBits* for MacPaint images or for TIFF images compressed using the PackBits RLE compression.
- *LZW* for TIFF images compressed using Lempel-Ziv-Welch compression.
- *LZW with prediction* for TIFF images using horizontal prediction together with standard LZW compression.
- *4-bit* or *8-bit RLE* for BMP images compressed using Microsoft's wacky RLE compression.
- *Photo CD* for images compressed using Kodak's Photo CD technology.
- *Photo — JPEG* for QuickTime PICT images compressed using Apple's JPEG compressor.
- *Animation* for QuickTime PICT images compressed using the Animation compressor.
- *Graphics* for QuickTime PICT images compressed using the Graphics compressor.
- *Compact Video* (or *Cinepack*) for QuickTime PICT images compressed using the Compact Video compressor.
- *Video* for QuickTime PICT images compressed using the Video compressor.
- *Uncompressed* for uncompressed images.

For QuickTime-compressed PICT images, the quality factor used during compression is given, in the standard range from 0.00 (worst) to 4.00 (best). For JFIF images, which do not store the original quality factor used in compression, only an estimate of this quality factor can be made from the tables stored in the image. Currently JPEGView can only estimate the quality of JFIF files created by the Independent JPEG Group's code, which uses a different quality scale than QuickTime, ranging from 0 (worst) to 100 (best).

File Format: This is the file format that the image was saved in. Possible values include PICT, JFIF, GIF, TIFF, BMP, MacPaint, and Startup Screen.

Image Length: This is the length of the image, given in kilobytes. If the image was compressed, then the savings gained by compression is computed and displayed either as a percentage for small compression gains (e.g., 60% compression, meaning that the compressed image is 60% smaller than the original) or as a ratio for highly compressed images (e.g., 7:1 compression, meaning that the compressed image is 7 times smaller than the original).

Statistics for the displayed image

Displayed Size: This is the width and height of the image you are actually seeing onscreen. If the image has been scaled, the scaling factor is given in parentheses as a percentage. If the hori-

zontal scaling on the image is different from the vertical scaling, the scaling factor is given as two separate percentages, horizontal scaling first.

Displayed Colors: This is the currently selected color set for the image, as seen in the Colors menu. Usually, JPEGView dithers images to ensure the highest quality display; if dithering is for some reason turned off, this is indicated as well.

Display Quality: This is the quality used for drawing the image, if it was drawn on a screen with 256 colors or more. An explanation of what this quality means is given in the *Colors and Color Reduction* chapter. If the image was drawn on a screen with less than 256 colors, this field displays "n/a" (not applicable).

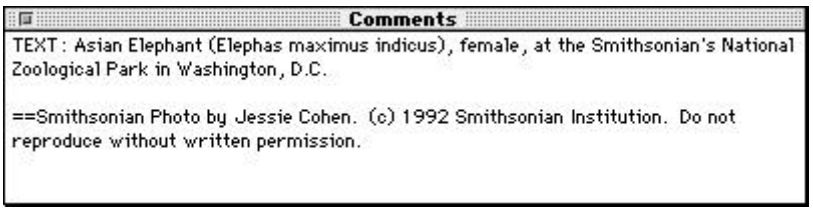
Display Time: This is amount of time it took for JPEGView to last draw the complete image, given in seconds. Note that this time is only updated when the image needs to be decompressed while the entire image window is visible on-screen; if the image is copied from an offscreen bitmap, this field is not affected.

Offscreen Bitmap: If JPEGView has created an offscreen buffer for the current image, its size is displayed here; otherwise, it will read "No bitmap."

Free Memory: This represents the amount of memory JPEGView has remaining. Note that, unlike previous versions, this takes into account the extra memory JPEGView sets aside for internal use (about 100k). Therefore, don't be worried if you see "0K" on this line—more than likely, JPEGView still has a comfortable amount of memory to work in.

The Comments floating window

JPEGView's Comments floating window allows you to view comments extracted from GIF files, JPEG files, and from JPEG-compressed PICT files. To make the Comments floating window visible, select **Show Comments** from the **Window** menu (or type `). When the window appears, it will contain the text of the comments, or "No comments found" if the image contained no comments. The Comments floating window is also automatically resized to display the full text of any comments.



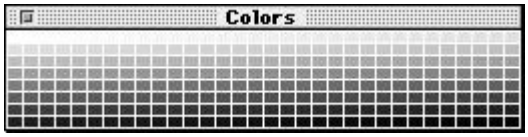
An typical use of image comments is to provide textual information along with an image.

JPEGView also provides a mechanism for automatically hiding and showing the Comments floating window depending on whether the active window has any associated comments. The *Preferences Options* chapter explains how this operates in greater detail. If you leave this option off, you can find out whether the active image contains any comments by looking for a diamond mark next to the **Show Comments**/**Hide Comments** menu item.

Finally, like the Statistics floating window, you can drag the text out of the Comments floating window and into another application which supports drag and drop (assuming you are on a drag-aware system).

The Colors floating window

The last of JPEGView's floating windows is the Colors floating window, which displays the current set of available colors on whichever monitor the window is currently residing. This is most useful on screens with 256 or fewer colors, where the selection of available colors becomes important for improving image quality. For screens with Thousands or Millions of colors, four smooth bands of red, green, blue, and gray are displayed, ranging from darkest to lightest.



The Colors floating window shows you what colors are available for displaying images.

To make the Colors floating window visible, select **Show Colors** from the **Window** menu (\). To hide it again, choose **Hide Colors** from the same location. As an interesting aside, the Colors floating window can be positioned across two or more monitors, in which case it will display the color set for the screen which can display the most colors.

Special Effects

JPEGView provides a small number of useful on-screen manipulation tools that allow you to adjust the scaling factor of the active image, to perform cropping and zooming, or to make use of the full area of your monitor. All these features are described in detail below.

The wonders of full screen windows

The Macintosh's interface is very nice, what with the menu bar and windows and all, but sometimes don't you just wish that a 640x480 image could be displayed in full on your 640x480 display? When you account for the height of the menu bar plus the height of the window's title bar, you find that you're losing a little less than 10% of your vertical resolution on a 13" monitor. (If the windows had scroll bars, you'd be losing even more!) With JPEGView, you can gain this area back and view images to the full resolution of your screen simply by choosing the **Full Screen** option from the **View** menu (F).

*An image can be switched to or from a full-screen window by choosing **Full Screen** from the **View** menu.*

The **Full Screen** menu item acts to toggle the view of the current image between a normal window and a full-screen, menubar-less window. In case it's not obvious which sort of window you're currently looking at, you can check the **View** menu to see whether the active window is in a full screen window by looking for the checkmark next to this menu item.

When a full screen window is in front, the menubar is hidden from view (that is, assuming the image window is on your menubar monitor). Although you cannot see the menus, all the standard -key shortcuts work just fine. In addition, by clicking anywhere inside the full screen window (but outside the current selection, if there is one), you can bring the menubar forward and access the menus in the usual manner. A second click inside the window will put the menubar away again. Several other actions can also force the menubar back to the front, including: closing the full screen window; selecting a normal (non-full-screen) window; and switching to another application.

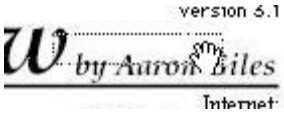
Selecting portions of an image in JPEGView

You may have noticed that when you move the mouse pointer inside the window of the active image, the pointer changes to a crosshair. In JPEGView, as in most graphics applications, you can select an area of the active image by clicking in the image, dragging a rectangle around the desired area, and then releasing the mouse button to make the selection final.

Selecting an area of an image in preparation for cropping.

Once a selection is made, an animated dashed rectangle is displayed around the selected area. If you select an area less than 16 pixels in one direction, or simply click without dragging, the selection is canceled.

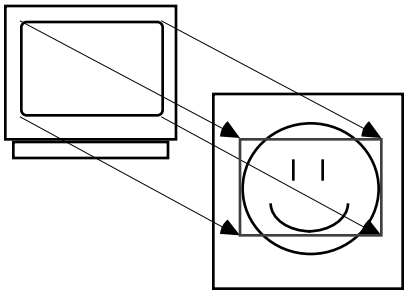
Once you've made your selection, moving the mouse pointer around some move reveals yet another cursor: the hand cursor. When the mouse pointer lies within the selection rectangle, the pointer becomes a hand to indicate that you can drag the selection area anywhere within the image while preserving its original size.



Once a selection is made, it can be dragged anywhere else within the image.

Note that in order to define a new selection rectangle, you must begin by clicking outside of the original rectangle; otherwise, JPEGView will think you want to drag the original selection around rather than begin a new selection. Use the cursor shape as your guide: if it is a crosshair, clicking will produce a new selection; if it is a hand, clicking will drag the current selection.

Another intriguing option for making a selection is to use the **Select Screen Area** (A) option under the **Edit** menu. This creates a selection rectangle in the middle of the active image that is exactly the proportions of the monitor that image is displayed on, scaled in the same way as the image. That is, the selection that is created represents the area of the image you could view if the image were displayed unscaled on your screen.



*The **Select Screen Area** command is one of JPEGView's most useful and most misunderstood features. Using the **Select Screen Area** command when a large image is displayed will produce a selection of the original image exactly as large as your screen. Of course, what you see on your screen is the scaled down image with the scaled down selection area, but the effect is nonetheless the same.*

Once created, this selection rectangle can be dragged about the image like any normal selection. If this concept seems unclear, read on into the next section to find out how this can be used in

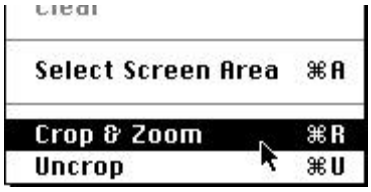
conjunction with cropping and zooming to form a very powerful tool for examining images in more detail.

After selecting a portion of the image, you are free to copy it to the clipboard, assuming you have enough memory available to do so. To give it a try, choose **Copy** (C) from the **Edit** menu. Because JPEGView often deals with very large images, it is easily possible that you will run out of memory during the copy operation; if this happens, you are probably best off copying the image in pieces and reassembling them by hand at the other end. Also note that if no selection is active, choosing **Copy** will attempt to copy the entire image.

Similarly, on a drag-aware system, you can drag the selected portion of the image outside of its window and into another application which supports dropping pictures. Because JPEGView already allows you to drag selections *within* windows, this operates a little strangely: when the mouse pointer is displayed as a hand, you can drag the image anywhere within the image itself. However, once you drag the mouse pointer outside the bounds of the window, it will change to a standard pointer, and you will begin the drag-and-drop operation. At this point, the original selection will remain unchanged, even if there was no target for the drop.

Cropping and Zooming

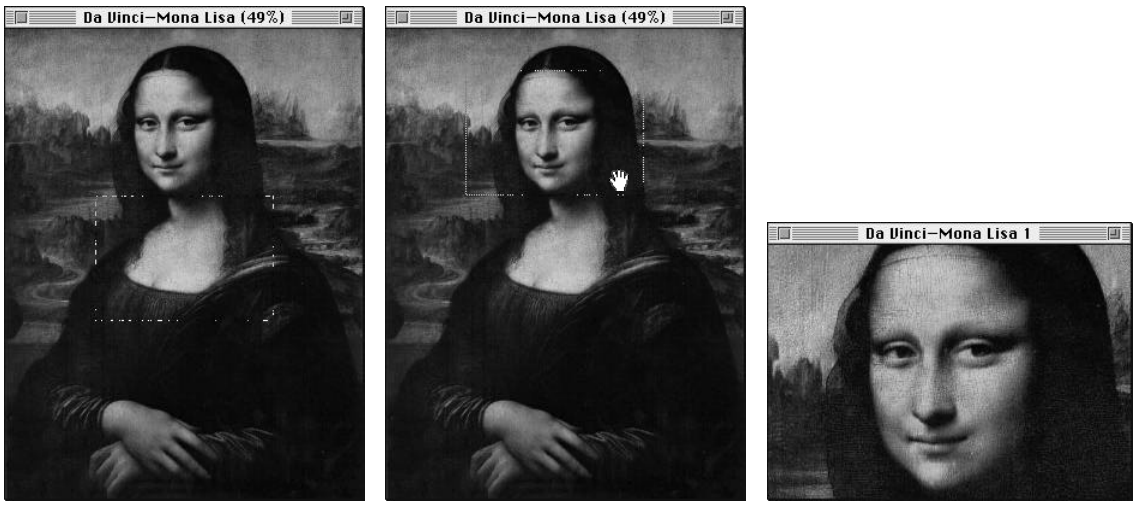
Once you've made your selection, you are ready to use one of JPEGView's most powerful features: **Crop & Zoom** (R), located in the **Edit** menu.



*The powerful **Crop & Zoom** function is available at the bottom of the **Edit** menu.*

This option takes the area of the image that you have selected and creates a new window containing only that portion of the image. That takes care of the "Crop" part. The "Zoom" part comes in when JPEGView tries to make this smaller portion of the original image fit on your screen. If you've selected an area in the original that will fit unscaled on your screen, then it is displayed as such. If the cropped area is still too large, it is scaled to fit on your screen at the maximum size possible.

The usefulness of this feature becomes apparent when used in conjunction with the **Select Screen Area** option. Let's say you long for the good old days, when scroll bars were in vogue and you could use your whole screen to look at the minute details in your images without having to see the image scaled down to 50% or whatever. Well, you can still do that! Just **Select Screen Area** on your image, drag the selection over the area you wish to examine, and then **Crop & Zoom**. The area you chose will expand to fill up your screen exactly—and you didn't have to scroll around helplessly looking for the good bits.



*How to live without scroll bars. First, use **Select Screen Area** to get a screen-sized area of the original image; this initially appears as a selection centered within the image (left). Next, drag the selection to the area you wish to examine in detail (middle). Finally, choose **Crop & Zoom** and watch as the area you chose zooms in to fill your screen with all the detail scroll bars could have ever afforded you (right).*

Once you've cropped an image, you can always return to the original full-size image by choosing **Uncrop Image** (U), again in the **Edit** menu, which will remove all cropping from an image and replace it with the complete original.

Changing the scale factors

The last of JPEGView's special effects is the ability to play with the on-screen scaling of images. This can be accomplished in one of two ways: either by hand, by dragging the lower-right corner of the image; or automatically, by using the commands at the bottom of the **View** menu. The former technique is probably the more intuitive of the two.

If you move your mouse pointer down into the bottom-right corner of the active image, you will notice that the cursor takes on yet another shape: that of a resizing box, which you're used to seeing in the bottom-right corner of windows in most other applications.

Moving the mouse pointer to the lower-right corner gets you the ability to resize the window by hand.

Clicking and dragging the mouse while the cursor is thusly shaped will allow you to change the size of the current image. By default, the new size of the image is constrained to have the same ratio of height to width as the original; however, you can remove this restriction by holding down any of the modifier keys (Shift, Control, Option, or).

Your other options for adjusting an image's onscreen size are accessible from the bottom part of the **View** menu, and all are relatively straightforward. You can double or halve the size of the current image with the **Double Size** (D) and **Halve Size** (H) menu items, respectively. Or you can force the image to expand to its maximum size on the current monitor with the

Maximum Size (M) item; the zoom box in the upper right-hand corner of an image's window will do this trick as well.

*The bottom part of the **View** menu, where you can adjust the displayed size in a number of ways.*

For more precise scaling, you can increment and decrement the scale factor in steps of 10% using the **Shrink by 10%** ([) and **Expand by 10%** (]) options. And when you've finished playing with scale factors, you can select the **Normal Size** (L) option to return the image to 100%.

Finally, for those with multiple monitor systems, there is the **Resize to Screen** (E) command, which will recompute the standard scaling for the image based on the size of its current monitor. This is useful, for instance, if you drag an image from a 13" monitor to a 19" screen and want to expand the image to its natural size on that screen.

Colors and Color Reduction

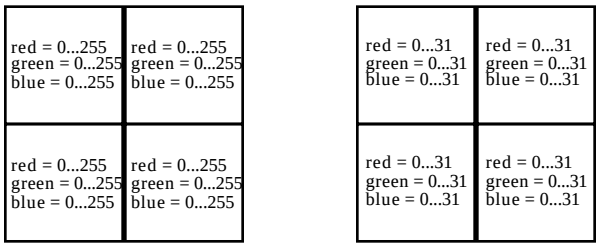
JPEG-compressed images, as well as many PICT, TIFF, and BMP images, can contain a lot of color information—16.7 million colors' worth, to be exact. This is just fine if you have 24-bit display hardware, capable of displaying in Apple's "Millions" mode, but unfortunately we aren't all so lucky. The fact is that the single most popular color setup on the Macintosh is the standard 8-bit, or 256 color display. It may not be immediately obvious that you can, in fact, view 24-bit color images with rather high quality on a 256 color display. After all, how can you possibly go from 16.7 million colors down to 256 colors without losing *something* significant?

Video modes on the Macintosh

To understand how colors and color reduction work, it is necessary to understand something about the way the Macintosh handles its monitors. The current version of 32-bit Color QuickDraw, which is responsible for all drawing on your Macintosh, supports six different screen modes, depending on how much video memory (sometimes called VRAM) your Macintosh has available for the screen. These modes are identified by the number of colors that each pixel, or dot, on your screen can represent. There are the direct video modes (Millions and Thousands in the Monitors control panel), the CLUT (or indirect) video modes (256 and 16 colors), and the monochrome modes (4-color and Black & White).

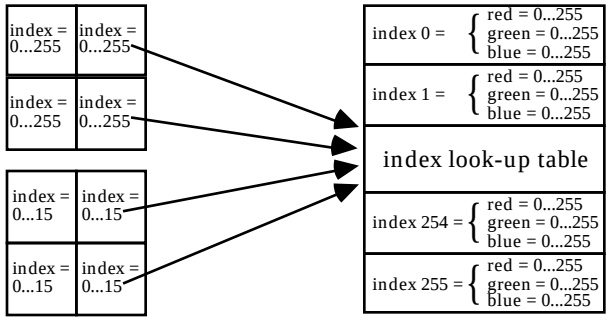
The direct video modes are so named because you can directly control the exact color of each pixel on the screen independently. When you look at a pixel on your color screen, what you are actually looking at are three dots of light superimposed on top of one other: one red, one blue, and one green. By varying the brightness of each of these three components, you can "create" virtually any color imaginable. Thus, you can say that red, green, and blue are the three "primary" colors of your monitor, in much the same way that red, yellow, and blue are the three primary colors of your watercolor set; if you combine them in just the right proportions, you can get any other color you want.

In direct video modes, you specify for each pixel exactly how much red, blue, and green it should display on your screen. This gives you complete control over the final result. What then becomes important is how finely you can adjust the brightness levels of each of these components. In Apple's Thousands mode, you can specify one of 32 possible brightness levels for each component, red, green, and blue, giving you 32 x 32 x 32 = 32,768 colors total. In Millions mode, you get to choose one of 256 possible brightness levels for each component, or 256 x 256 x 256 = 16,777,216 colors.



Direct color modes allow precise control over the color of each pixel. With Millions of colors (left), you have finer control over each component's brightness than with Thousands of colors (right).

When you scale back to the 256-color and 16-color video modes, this direct control over the color components isn't nearly as effective, since you would only be allowed at most 6 or 7 possible brightness levels in each component. The Macintosh's solution to this problem lies in its use of color lookup tables (CLUTs). A CLUT is a small table of colors (with 256 or 16 entries, in this case) chosen from any of the 16.7 million colors available in the Millions mode. To display one of these colors, the Macintosh gives the display hardware a color index, say, from 0 to 255, which is then used to pick a corresponding color from the CLUT to display on your screen.



Indexed color modes allow you to choose a fixed number of specific colors, one of which is specified for each pixel through an index. Both the 256 color (top) and 16 color (bottom) modes work this way.

This way of doing things gives you tremendous flexibility in choosing your colors. For example, on a 256-color system, you can design a color set with 256 levels of grayscales for your display just as easily as you can design a color set with 256 colors chosen at random from the 16.7 million available. Of course, what you would obviously prefer is the ability to choose the optimal set of 256 colors for displaying a given image, and this is exactly what JPEGView's two-pass color reduction feature does for you.

Finally, I should admit that the 4-color mode is also technically a CLUT mode, even though I called it a monochrome mode before. The reason I grouped it in with the black & white mode is that your Macintosh requires every display to be capable of showing black and white, which leaves only 2 colors that are truly free to be chosen. Since you can't even get the essential red, green, and blue with those 2 colors, it is best simply to choose two intermediate grayscales to give you the highest-quality image, leaving you with a monochrome display.

JPEGView and color sets

If at least one monitor in your system setup is operating in one of the CLUT modes—either with 16 colors or with 256 colors—then you can use JPEGView's **Colors** menu to select a new color set for the current image. There are currently four different color sets available: the **System Colors** (Y), the **Grayscales** (G), the **Image Colors** (J), and the color set produced with **Two-Pass Color Reduction** (T).

*Selecting **Two-Pass Color Reduction, when available,** will nearly always give you the highest quality image.*

The first two options—the **System Colors** and the **Grayscales**—are color sets that you're probably familiar with already, though you may not have known it. The former is simply the set of colors that you're used to seeing when you're in the Finder, while the latter is what you see when you switch your monitor into Grayscales mode in the Monitors control panel. To see what sort of color selection is provided by these or any other sets of colors in JPEGView, you can make the Colors floating window visible. As you switch color sets within JPEGView, the colors displayed in the floating window will change to reflect the new set of colors.

Some images that you load with JPEGView, in particular GIF images, contain a color set internally, defining which colors should be used to display the image. If JPEGView finds such a color set within an image, it will allow you to choose those colors via the **Image Colors** item in the **Colors** menu. If no color set information is provided with the image, this menu item will not be available.

Finally, for images that contain Thousands and Millions of colors, JPEGView provides the ability to generate a custom color set for the image, produced by examining the distribution of colors within the image. This is known as two-pass color reduction (sometimes called color quantization), for the simple reason that JPEGView must first scan through the entire image once to determine the proper color set, and then the image must be scanned a second time during the actual display process. For the other color sets available, the display process only requires one pass through the image.

Two-pass color reduction will very nearly always give you far better results than any of the other color options. The main downside to this feature is that it simply requires more time to display an image. JPEGView leaves the default choice up to you; see the *Preferences Settings* chapter for more details on how you can control this behavior. In addition, once you have performed two-pass color reduction, JPEGView allows you to save the calculated color set information with the image, so that you don't need to perform the calculation again. The chapter *Saving Images* explains how you can do this.

Dithering—how and why it works

No matter which color set is selected, you also have control over whether or not to dither the resulting image via the **Dithering** item in the **Colors** menu (B):

*Dithering can be toggled on and off through the **Dithering** item in the **View** menu.*

Dithering is a technique that "simulates" colors that aren't available in the current color set by modifying the colors of the surrounding pixels. As a simple example of how dithering accomplishes this feat, consider a random pixel in an image, whose color is supposed to be purple. If

the current color set contains a color that exactly matches that purple, that particular color is chosen, and there are no problems; dithering has no net effect.

But imagine a color set had no purples at all, but did have shades of red and blue. Without dithering, that purple pixel would appear as a red pixel, since red is the closest match we can get to purple with the current (red and blue) set of colors; unfortunately, this won't look very purple in the end. Now turn dithering on. Dithering finds the closest match—red—just as before, but then it looks at the difference between this closest match and the original color we wanted. In this case, dithering would notice that there should have been more blue in this pixel, and, to compensate for this loss, it would add a little bit of blue to the neighboring pixels.

Now imagine that one of these neighboring pixels, whose blue component has just been increased, was originally the same purple as the original pixel. With the adjustment, it is now a rather bluish-purple color. So, when it comes time to find the closest match to this new color, we will choose blue rather than red, because blue is closer to bluish-purple than red is. Again, we calculate the difference between the chosen blue color and the original color we wanted (bluish-purple), and see that some red was missing. Dithering will thus add a little bit of red to pixels surrounding this one, and proceed on to the next pixel, continuing this for all the pixels in the image.

In the end, an area of solid purple—which cannot be displayed as such with our example blue and red color set—will appear as a pattern of alternating blue and red dots, which will actually give the appearance of purple if you don't look too closely. In this way, dithering approximates colors by finding what was missing in the current pixel and distributing that difference to the surrounding pixel. The advantage of dithering is that it effectively simulates more colors than your monitor can actually produce. The disadvantage is that the image comes out a little less distinct, and can sometimes suffer from dithering artifacts. The next section discusses this latter problem in more detail.

Scaling, dithering, and display quality

The Macintosh's 32-bit QuickDraw, which is the highly-optimized system responsible for all drawing operations that occur on your system, supports the ability to automatically perform dithering and scaling operations on an image. This functionality makes it very easy to incorporate such features into an application program, and for most operations, it does an excellent job. However, there are some limitations in QuickDraw's dithering that only become apparent when displaying images on 256-color screens using two-pass color reduction.

The problem is that the dithering algorithm used by QuickDraw works best when the available color set is not very well matched to the colors that are in an image, e.g., when an image is displayed with only 16 colors, or with 256 colors using the system colors. When the available colors are a good match for the image colors, however, QuickDraw's dithering can produce dithering artifacts, or strangely discolored areas in the image.

In order to fix this problem of dithering artifacts, JPEGView implements its own dithering algorithm, which remedies the problem for most images. Unfortunately, in order to support its own dithering, JPEGView must also support its own scaling, since the two operations are intricately coupled. Although JPEGView's high-quality dithering algorithm itself is fast—almost as fast as QuickDraw's built in dithering—JPEGView does not know how to scale an image as fast as QuickDraw. Because of this, JPEGView supports two forms of scaling, in addition to QuickDraw's own routines.

Selecting which set of routines to use for drawing is accomplished through the **Quality** menu, available under the **Colors** menu. Three options are available: **Very High** quality (Shift-V), **High** quality (Shift-H), and **Normal** quality (Shift-N).

*The **Quality** submenu, in the **Colors** menu, allows you to trade off display time for image quality*

These options are only useful (and only available) if the image is drawn on a screen with Thousands or more colors available, or on a screen with 256 colors if the image is drawn using either two-pass color reduction or the image colors. In all other situations, the normal QuickDraw drawing is used.

What JPEGView calls Normal quality drawing is simply QuickDraw's standard routines, which give fast results with the possibility of the dithering artifacts mentioned above on 256-color screens. Normal quality is also used when the current color set does not match the image's colors very well, since QuickDraw's dithering does a fine job in this situation.

High quality drawing combines JPEGView's improved dithering with a fast, but relatively coarse, scaling algorithm. For most purposes, this fast scaling does a good job, and even gains you a small speed increase over the QuickDraw scaling you're used to. High quality drawing is only available on 256 or Thousand-color screens; those with Millions of colors available won't notice any dithering difference, and the scaling is visually a little worse than standard QuickDraw scaling, making High quality into a net loss over Normal quality for such users.

Very High quality drawing takes JPEGView's improved dithering and adds a slower, but very accurate scaling routine. For those with an eye for detail, the scaling used in Very High quality drawing is often better than standard QuickDraw scaling, making this quality level useful for those with Thousands and Millions of colors as well. Note that when an image is not scaled, High quality and Very High quality are identical, since the difference lies only in the scaling algorithm used.

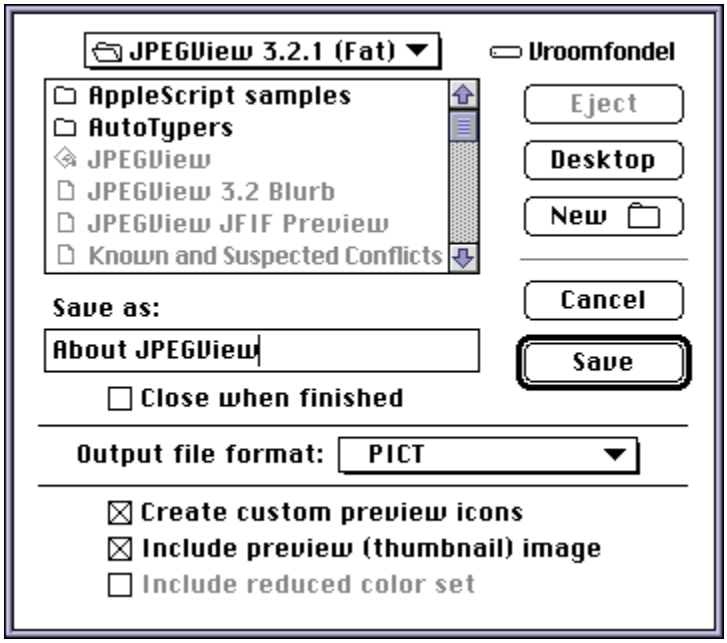
To help see the difference that JPEGView's improved dithering can make, try opening the "Parrots JPEG" image that was included with JPEGView. View it using both Normal and High quality on a 256 color screen with two-pass color reduction turned on; the difference should be clear. Note that "Parrots JPEG" is an extreme example of the problems encountered with QuickDraw's dithering; don't expect such dramatic improvements on all images!

Saving Images

Although JPEGView has been designed primarily as an image viewer, it also sports the ability to convert image files between the standard JFIF format and the Macintosh's own JPEG-compressed PICT format, as well as options for creating previews and icons, for pre-calculating two-pass color reduction, and for saving cropped image files. Note that only JPEG-compressed files may be saved with JPEGView; other compression schemes are supported only for viewing. You will know whether you are able to save the current image by checking the **Save As** item under the **File** menu; if it is dimmed, then JPEGView cannot save the active image.

File formats in the Save dialog

When you choose the **Save As** item from the **File** menu, you are presented with a standard save file dialog, with a few extra options appended to the bottom:



The Save dialog, with all of JPEGView's wonderful options.

Anything that looks familiar works as usual. The remaining options allow you to choose the output file format or to include additional data with the image file. Additionally, by checking the

"Close when finished" box, you can instruct JPEGView to immediately put away the image after it has finished saving it, thereby avoiding any long window updates that might follow.

If you are saving an image that has been compressed using JPEG, the file format popup menu will offer you several choices, allowing you to save your image in one of the three output file formats that JPEGView currently supports: JPEG-compressed PICT, JFIF, and PICT Resource.

When saving any JPEG-compressed image, you get three choices for the output file format.

More details concerning the properties of the PICT and JFIF file formats can be found in the *File Formats and File Types* chapter. The oddball format available here is the PICT Resource format, which is mainly useful for saving JPEG-compressed PICT images in a useful format for programmers.

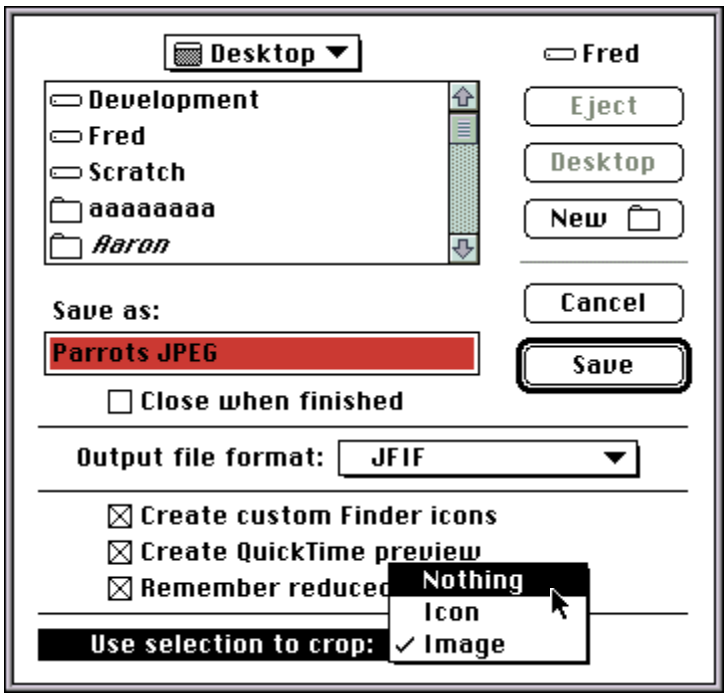
Also, please note that PICT images which are "banded", that is, made up of several small JPEG images sewn together, cannot be saved as a JFIF file. Banded PICT images are sometimes produced by QuickTime whenever memory space is low. Since the JFIF image format was designed to store only a single image, a banded PICT cannot be converted to JFIF without decompressing the composite image and recompressing it as a single image. JPEGView does not support this sort of functionality.

If the image you are saving has not been JPEG-compressed, then you will have to save it in its original format, and there will be no popup menu for you to choose from. However, you will still have the luxury of choosing to add QuickTime previews and/or custom color Finder icons to the image for future ease-of-use.

Saving modified images

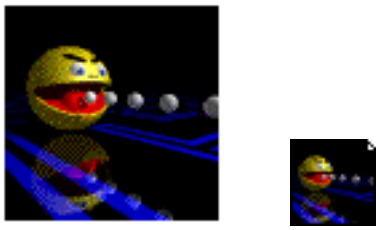
In addition to simply allowing you to save the image data out to a file, JPEGView also allows you to save cropping information, QuickTime preview images, custom Finder icons, and color reduction with your image.

There are two ways to get JPEGView to save the cropping information with an image. The most straightforward is simply to crop the image in JPEGView, and then choose **Save As** from the **File** menu with the cropped image frontmost; when saved, the cropping is retained. Alternatively, you can select the portion of the image you wish to crop to, and then issue the **Save As** command. When you do this, the popup menu at the bottom of the dialog will be enabled, allowing you to tell JPEGView to "Use selection to crop Image".



If you choose to save an image while a selection is still active, the popup menu at the bottom of the Save dialog will enable you to crop either the image or the image's icon to include only the selected area.

If you have checked the "Include preview (thumbnail) image" box, JPEGView will create a preview image. This preview image is simply a smaller representation of the full image, allowing you to quickly see what the image is without having to decompress the whole thing first. QuickTime automatically supports viewing of these preview images in the Open dialog for all formats except JFIF. The ability to view JFIF previews can be added to QuickTime by placing the *JPEGView JFIF Preview* extension is your Extensions folder. See the chapter *Viewing Images* for more details on viewing JFIF previews.



An example of a preview and a custom preview icon, side by side.

JPEGView also supports the creation of custom icons for saved images. This option extracts a portion of the image (exactly which portion is selectable in the Preferences; see the chapter *Preferences Options* for more details) and shrinks it down to the size of a Finder icon, adding this information to the saved file. The resulting document will then appear in the Finder not with the standard JPEGView icons (which are pretty cool to begin with), but with this postage stamp-sized icon instead. If you have a color system, this might enable you to guess which image you would see if you double-clicked the icon.

The full complement of JPEGView's document icons.

Sometimes it is convenient to be able to specify a smaller portion of the image from which to take the icon data. JPEGView allows you to do this by simply selecting the subsection of the image you want, and then choosing “Icon” from the “Use selection to crop...” popup menu at the bottom of the Save dialog. This will instruct JPEGView to use the selected area of the image for the icon, while leaving the entire image intact (to crop the image as well as the icon, choose “Image” from the popup menu).

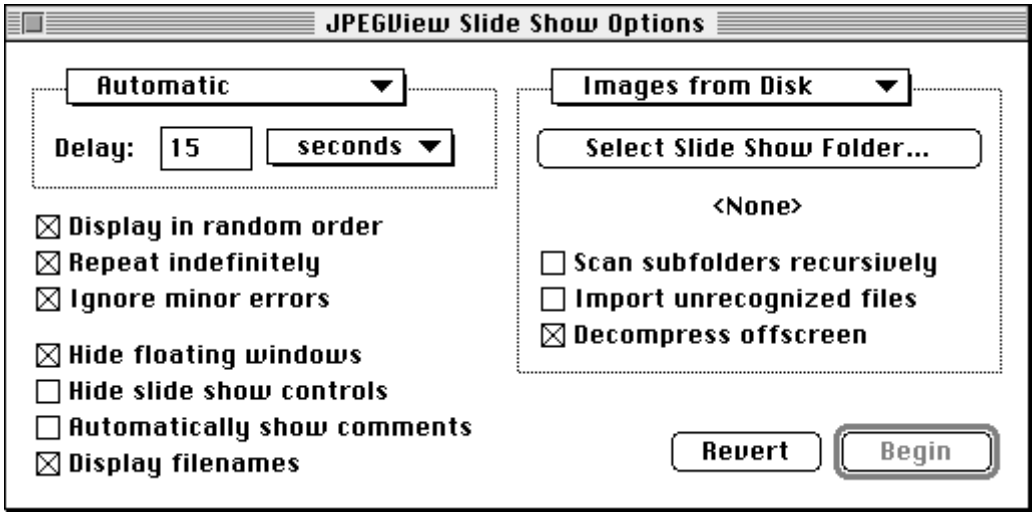
JPEGView’s color reduction information can also be included whenever you save a JPEG-compressed image. What this means is that when opening this image in the future, JPEGView and certain other image viewers can detect and make use of this information without having to slog through the time-consuming color reduction. When you open the image with JPEGView specifically, this information serves to define the Image Colors, which JPEGView will automatically use.

The final result

Once you have entered a filename and have chosen your desired options, click on the Save button to perform the operation. If you chose to create a preview or icons, a progress dialog will appear to keep you updated on the status of JPEGView’s work; otherwise, JPEGView will perform the save operation quickly and quietly.

Slide Show

One of JPEGView's more powerful features is its ability to display an entire folder's worth of images in a slide show. Unfortunately, everybody seems to want their slide show to operate differently than everyone else's, which is the reason why JPEGView presents you with so many options when you bring up the **Slide Show Options** (via the **File** menu or the I key combination). Fortunately, though, you never need to mess with most of them!



The JPEGView Slide Show Options dialog, with its overwhelming selection of options.

Slide shows for beginners

The concept of a slide show is simple: first, you need to choose a group of images to view; and second, you need to decide who controls the JPEGView projector. In regard to the first decision, JPEGView allows you to choose one of two possible sources for your images: either from memory or from disk. The simplest way to see a slide show is to open up several images in JPEGView and choose "Images from Memory" from the pop-up menu on the right-hand side of the Slide Show Options dialog. This tells JPEGView that it should choose its slides from among those that you've opened.

Selecting the source of the slide show images from the pop-up menu on the right side of the options dialog.

Your other option is to instruct JPEGView to choose from image files that reside on your disk. Running in this mode of operation requires an extra step, in addition to choosing "Images from Disk" from the rightmost pop-up menu. Inside the box governed by this pop-up menu is a large button which reads "Select Slide Show Folder". Clicking on this button will bring up a dialog box which will allow you to select a folder; its operation is pretty self-explanatory. Once a folder has been selected, its full path will be displayed below the button.

An alternative—and simpler—method of selecting the slide show folder is available if your system is drag-aware (running System 7.5 or an earlier System 7 release with the *Macintosh Drag and Drop* extension and Finder 7.1.2 or later). Rather than navigating through a dialog to choose the slide show folder, you can simply drag any folder or alias from the Finder into the Slide Show Options window. As you do this, the window will become highlighted, and when you release the mouse button, the folder will change to the one you dragged in.

After choosing the source of the slide show images, you must then decide who retains control of the switching between slides, either you or JPEGView. The pop-up menu on the left-hand side of the Slide Show Options dialog allows you to control this aspect of the slide show. Choosing "User-controlled" from the pop-up menu tells JPEGView to wait for you to explicitly command it before proceeding on to the next slide.

Selecting how the slide show is controlled from the left-side pop-up menu.

Your other choice is to give JPEGView complete control of things, by selecting "Automatic" from the leftmost pop-up menu. Doing this enables the two items in the box below the pop-up menu, allowing you to specify the minimum time interval between successive slides. Use the newly-enabled pop-up menu to choose the units—either seconds, minutes, or hours (?!)—and type the appropriate number in the box to its left.

Once you've selected these basic options, you're ready to proceed. Click the "Begin" button to start up the slide show immediately, or simply close the window to save your changes but postpone the beginning for later. If, for some reason, the "Begin" button is dimmed, it means that JPEGView can no longer find the folder you selected; to remedy the situation, either choose another folder or insert the disk that the folder resided on. The other button, "Revert", restores the settings to what they originally were when you first opened the Slide Show Options dialog.

Finally, the slide show can be initiated directly from the **File** menu by choosing **Begin Slide Show**, or by typing I while holding down the Shift key. This causes the slide show to be run with the last settings you chose in the options dialog.

Help! What's going on?!

Once begun, JPEGView's slide show proceeds inexorably forward until you manage to stop it (or, optionally, until it finishes cycling through all the images). As each image is displayed, the next image is prepared in the background. Because this preparation can sometimes take longer than the automatic slide show delay, it is useful to know whether JPEGView is still preparing the next image or just waiting for the delay to pass.

While it is busy preparing the next image for display, JPEGView provides three indicators that you can use to tell that it is busy. The first is the familiar spinning busy cursor, visible at all times in user-controlled slide shows, or visible by moving the mouse in automatic slide shows. In addition, JPEGView displays a small white progress dot in the upper-right corner of the screen where

the images are displayed. As things progress, the little dot rotates around to indicate that everything is still chugging away.

The third indicator is provided in the status line of the Slide Show Controls floating window, if it is currently visible on the screen. If it isn't immediately visible when the slide show begins, simply click the mouse button anywhere outside of a floating window to make it appear. A second click will make it disappear again, clearing it out of the way of any image displayed underneath the controls. You can control whether the controls window is visible through an option in the Slide Show Options dialog; see below for more details.

The Slide Show Controls floating window provides point-and-click control of the slide show plus a status line.

The status line of the Slide Show Controls floating window serves as a running commentary of JPEGView's current operations. Here is what you can expect to see on this line:

- Loading: *file*. JPEGView is reading the file named *file* from disk.
- Reducing: *image*. JPEGView is performing two-pass color reduction on *image*.
- Rendering: *image*. JPEGView is currently drawing *image* offscreen.
- Drawing: *image*. JPEGView is drawing *image* onscreen.
- Ready. JPEGView has finished its preparations for the next image and is waiting for you to manually advance (user-controlled slide show only).
- Delaying... *seconds*. JPEGView has finished its preparations for the next image and is killing time before displaying it, with seconds *seconds* remaining (automatic slide show only).
- Paused. The slide show is currently paused.

Let me off this crazy thing!

As JPEGView's slide show progresses, you have the opportunity to control its progress, either through the push-button controls available in the Slide Show Controls floating window, or through keyboard shortcuts provided to mimic these controls. Currently, there are four push-button controls available: forward, reverse, stop, and pause. Exactly what happens when you activate any one of these controls is described below.

 ***Forward***. This button is generally used to advance the slide show forward by one slide whenever JPEGView has finished preparing the next slide (that is, when the status line is displaying either "Ready" or "Delaying..."). If you press this button while JPEGView is still busy, it will activate and stay depressed until JPEGView is finished with the next slide, at which point it will advance to it immediately. To make matters simple, just think of the forward button as meaning "advance to the next slide as soon as you're ready."

To perform this operation from the keyboard, or when the Slide Show Controls are not visible, simply hit any key other than those that are reserved for the other functions below. The space bar, right arrow, and Enter keys are popular choices.

 ***Reverse***. Use this button to back up one slide in the slide show, relative to the slide that is currently displayed on the screen. Note that the reverse function may be disabled if the first slide in the list is being displayed (since there is nowhere to back up to!) You can also use the left arrow key on the keyboard to back up in the same way.

 ***Stop***. Click this button to stop the slide show immediately, leaving the last image displayed on screen. When you do this, the Slide Show Controls floating window will disappear, and any previously visible floating windows will be shown again. The keyboard equivalents for this function are the Escape key (esc) or . (that is, type the "." (period) key while holding down the key).

 ***Pause***. Use this button if you want to temporarily pause the slide show. When first activated, the Slide Show Controls floating window is automatically made visible, and the menu bar is displayed. You can then switch to other applications, end the slide show, modify some (not all) of the slide show options, or do whatever else you care to do before resuming the slide show. When you switch back to JPEGView, simply click the depressed pause button again to resume the slide show where you left off. Note that JPEGView may need to reload and reprocess the next image, depending on exactly where you interrupted processing.

JPEGView also supports an extension to the paused slide show which allows you to not only pause but also hide all traces of the slide show. If you hold down the Option key as you click the pause button, JPEGView will automatically hide all of its slide show windows and send itself into the background. To resume after this happens, simply bring JPEGView back to the front by selecting it from the application menu on the far right side of the menubar. JPEGView will make its windows visible again, and you will be back in standard pause mode; clicking on the pause button will resume where you left off.

To pause the slide show from the keyboard, you can use the up arrow key. Hiding and pausing the slide show can be accomplished by holding down the Option key as you type the up-arrow.

Finally, please note that the response to your commands—both mouse-based and keyboard-based—will very often not be immediate, so give JPEGView a couple seconds before getting violent.

Playing with the options

Unless you tell it otherwise, JPEGView takes all your selected slides, mixes them up in a random order, and then displays them over and over again until you interrupt the slide show. However, using JPEGView's Slide Show Options dialog, you can customize the operation of the slide show to suit your own purposes. Below is a list of all the available options, along with a description of what each option's effects are.

Display in random order (default setting is *on*). Turning this option on tells JPEGView that you would like to see the slides in a completely random order, rather than in their natural order. Turning it off leaves the slides in their natural order. For a memory-based slide show, the natural order is the order you see the files in the **Window** menu; for a disk-based slide show, this natural order is the order they were found on disk, usually alphabetical within each folder.

Repeat indefinitely (default setting is *on*). Checking the box next to this item will cause JPEGView to cycle through the slide show until interrupted by the user. If the "Display in random order" checkbox is also checked, the slides will be randomly reordered before each cycle. If

this box is left unchecked, JPEGView will end the slide show after a single cycle and display an alert telling you it has finished.

If you turn off the "Repeat Indefinitely" option, JPEGView will alert you once it has displayed all the slides.

Ignore any minor errors (default setting is *on*). With this box checked, JPEGView will quietly ignore any minor errors that occur when loading slide show images. "Minor" errors include corrupt image warnings, low memory conditions, and the like. It is possible that leaving this option on can cause some images not to be displayed during a slide show; if this is the case, you should turn this option off and see which errors are causing the problem.

Hide floating windows (default setting is *on*). Turning this option on automatically hides any visible floating windows for the duration of the slide show. Otherwise, the floating windows will remain visible, floating above the images as they are displayed. Note that floating windows enjoy special status during a slide show: they can be shown, hidden, and moved about without interrupting the slide show or proceeding onto the next slide. To toggle the visibility of a floating window, simply use the appropriate -key shortcut; to drag a floating window, just drag it as usual, but note that this will only work when the cursor is a standard arrow shape!

Hide slide show controls (default setting is *off*). Use this option to control whether the Slide Show Controls floating window is displayed at the start of the slide show. If this box is checked, no window is displayed; however, access to the window is still enabled through mouse clicks during the slide show. Leaving this box unchecked displays the Controls floating window immediately when the slide show begins.

Automatically show comments (default setting is *off*). Checking this option gives control over the visibility of the Comments floating window to JPEGView, meaning that it will automatically appear whenever comments are found for the current image. If this option is left off, the Comments floating window remains under user control.

Display filenames (default setting is *on*). Turning this option on displays the name of each image file in the lower right-hand corner of the image. If there is enough room either below the image or to the side of the image, the name is displayed there; otherwise, the file name obscures the corner of the image. With this option off, there will be no indication of which file is currently displayed.

If there is not enough room next to or below an image, JPEGView will display the filename over the bottom-right corner.

Options for disk-based slide shows

The following options are only meaningful for slide shows based on disk images. If you have selected a slide show from memory, these options will all be unavailable.

Scan subfolders recursively (default setting is *off*). If you check this option, then JPEGView will not only display all the files in the folder you've selected, but will also scan any folders contained inside the selected folder, as well as folders within those folders, etc. In fact, you could even select the Desktop with this option checked, and JPEGView would happily scan all the folders on all mounted drives for images. (I don't really advise this, however!) Turning this option off means that the search is limited to files within the selected folder only.

If you turn on "Scan subfolders recursively", JPEGView will display a progress dialog while it's scanning.

Import unrecognized files (default setting is *off*). Checking this option causes JPEGView to add to its list of slide show images any files it finds in the selected folder, no matter what type of file the Finder says it is. As with the **Import** feature in the **File** menu, this is for the most part only useful to those with CD-ROM drives that have lots of incorrectly typed files. Leave this option off if you want to only open images whose file types are correct.

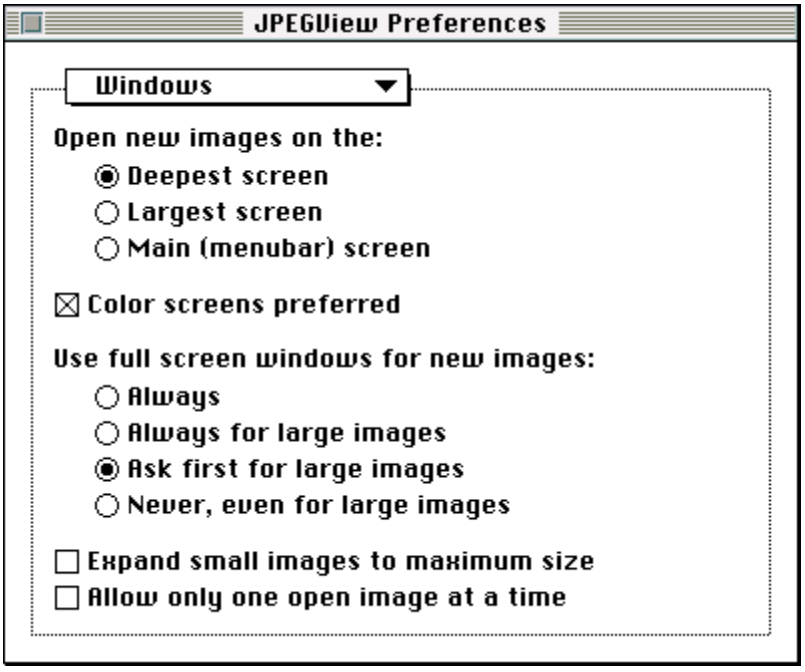
Decompress offscreen (default setting is *on*). With this box checked, JPEGView attempts to completely process the next image offscreen—including both color reduction and drawing— before removing the current image. The success of this option is highly dependent on how free memory you have in your system. If there is not enough memory to draw the next image offscreen, it will have to be decompressed and drawn onscreen, meaning that more free memory will give you smoother transitions between slides. If you leave this option off, JPEGView will decompress and draw every image onscreen (though color reduction is always done offscreen during a slide show).

Preferences Settings

JPEGView provides a number of options for controlling how it handles images by default. These are collectively known as the preferences options, and can be accessed via the **Preferences** menu item in the **File** menu. The different settings are grouped into five categories—Windows, Drawing, Bitmaps, Files, and Miscellany—and you can select which set to modify via the pop-up menu near the top of the Preferences dialog. As you make changes to the options, they take effect immediately; close the preferences dialog to save the changes to disk. Each category and its corresponding options are described in detail below.

Windows preferences

The settings in this category allow you to control JPEGView's use and placement of windows when opening new images.



The Preferences dialog displaying Windows preferences.

Open new images on the... (default is *Deepest screen*). These settings allow you to choose which monitor JPEGView uses for opening new images:

Deepest screen. This is the screen that is capable of displaying the most colors.

Largest screen. This is the screen that has the largest number of pixels.

Main (menubar) screen. This is the screen that has the menubar.

Of course, if you have only one screen, this option doesn't affect anything; it is primarily for choosing among multiple monitors on a "two-headed" system.

Color screens preferred (default is *on*). In conjunction with the screen selection above, this option lets you choose color screens over black & white/grayscale screens whenever a conflict occurs. For instance, if you chose *Largest screen* above, and had two monitors connected which were the same size, this option would let you choose either the color or the grayscale monitor first (assuming of course that they weren't both color or both grayscale!)

Use full screen windows for new images... (default is *Ask first for large images*). This setting controls how JPEGView handles the automatic use of full screen windows:

Always. Use a full screen window for every image opened, no matter what its size.

Always for large images. Use a full screen window for any image taller than the height of your screen.

Ask first for large images. Ask to use a full screen window for any image taller than the height of your screen.

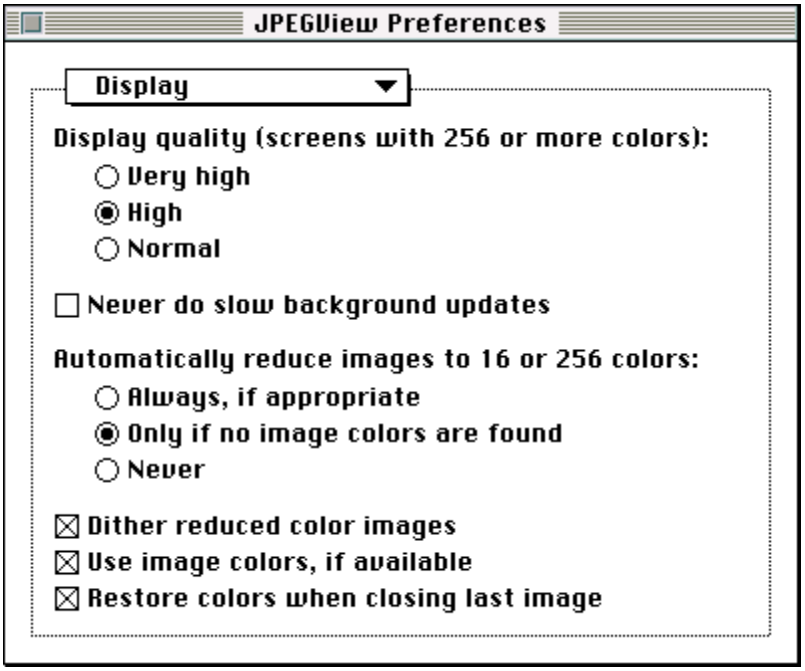
Never, even for large images. Never use a full screen window for any image opened, no matter what its size.

Expand small images to maximum size (default is *off*). This option lets you choose the normal size of any image smaller than the screen it is displayed on (including small images produced from cropping). With this option off, these images are simply displayed at their original size, with no scaling. Turning this option on automatically expands these images to their largest possible size on the screen whenever they are created.

Allow only one open image at a time (default is *off*). This option allows you to insist that there be only one image open at any given time. With this option turned on, JPEGView will automatically close all open images before loading in any additional images. This helps keep your desktop clear of multiple images and saves you the trouble of closing every image manually before opening a new one. Leaving this option off allows you to open multiple images simultaneously without any hassle.

Display preferences

These preferences options control exactly how JPEGView chooses colors and handles drawing images when you open them.



The Preferences dialog displaying Display preferences.

Display quality (screens with 256 or more colors)... (default is *High*). This setting controls the drawing quality for newly-opened images:

Very high. Selects the highest-quality display, using JPEGView's custom dithering and high-resolution scaling algorithms.

High. Selects a high-quality display, using JPEGView's custom dithering and faster scaling algorithms.

Normal. Selects normal-quality display, using the QuickDraw's built-in dithering and scaling algorithms.

For more information on these quality settings, check out the *Colors and Color Reduction* chapter. Note that these quality settings only affect color displays with 256 or more colors, and that high-quality only works on 256 and Thousand-color displays. For this reason, the quality setting reported by JPEGView may actually be lower than what you set here.

Never do slow background updates (default is *on*). This setting lets you disable slow window updates when JPEGView is not the active application. When this option is checked, obscured image windows that have no offscreen bitmap will remain displayed as a pattern of dots until you bring JPEGView to the front again. Turning this option off will allow these updates to proceed normally, possibly delaying other applications.

Automatically reduce images to 16 or 256 colors... (default is *Only if no image colors are found*). These settings let you control when JPEGView performs two-pass color reduction on newly-opened images:

Always. Performs two-pass color reduction on every image where it is applicable, even if the image has its own color set.

Only if no image colors are found. Performs two-pass color reduction on an applicable image only if the image has no color set defined with it.

Never. Never performs two-pass color reduction on any image.

The use of two-pass color reduction is limited to “high-color” images, that is, images containing Thousands or Millions of colors, and then only when they are displayed on screens with 256 or 16 colors.

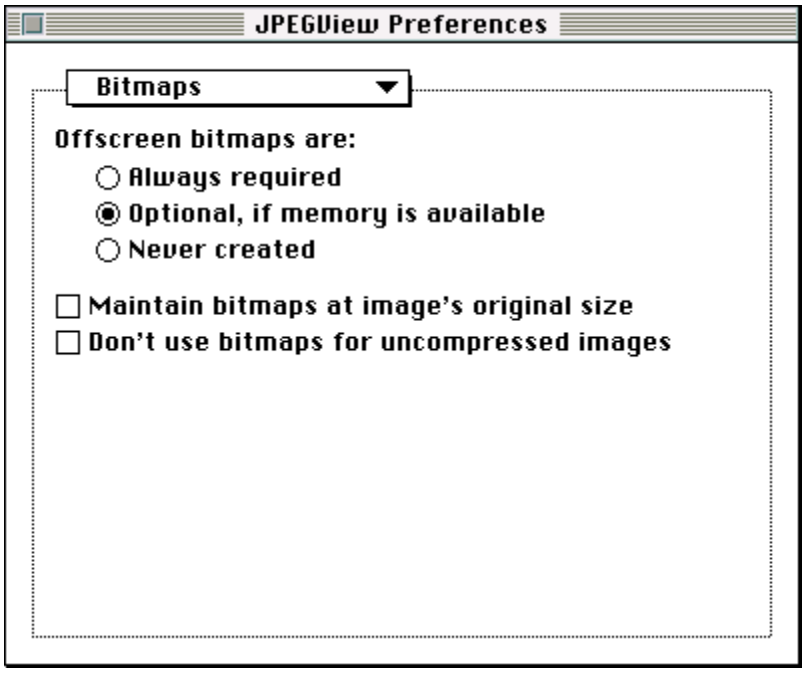
Dither reduced color images (default is *on*). This option gives you control over JPEGView’s dithering whenever two-pass color reduction is used on an image. Because two-pass color reduction usually does such a good job of choosing colors, you may find that dithering is unnecessary. If this box is checked, images displayed with two-pass color reduction will always be dithered. If this box is left unchecked, then these images will be left undithered.

Use image colors, if available (default is *on*). This checkbox controls JPEGView’s use of image colors. When image colors are found stored with the image data, JPEGView will automatically use them for displaying the image if this option is checked. Otherwise, the standard system colors or two-pass color reduction (whichever is appropriate) will be used.

Restore colors when closing last image (default is *on*). This option lets you instruct JPEGView to automatically restore the system colors whenever the last open image is closed. With this option off, closing the last image may leave the system in a somewhat discolored state. But turning this option on will tell JPEGView to manually reset the colors after closing the last image. The main disadvantage to this is that everything on the desktop will need to be redrawn in the new color set.

Bitmaps preferences

The items in this section control how JPEGView maintains its offscreen bitmaps, which are used for fast window updating.



The Preferences dialog displaying Bitmaps preferences.

Offscreen bitmaps are... (default is *Optional, if memory is available*). These settings control JPEGView’s creation of offscreen bitmaps.

Always required. Requires offscreen bitmaps for all new images. If an image can be loaded, but no bitmap can be created, JPEGView will not allow you to open the image and will give you an out of memory error. Further, once the bitmaps are created, they cannot be deleted to make room for other operations to proceed.

Optional, if memory is available. Makes offscreen bitmaps an optional feature, contingent on there being enough memory to create them. If more memory is needed for another operation, existing bitmaps can be deleted and later recreated.

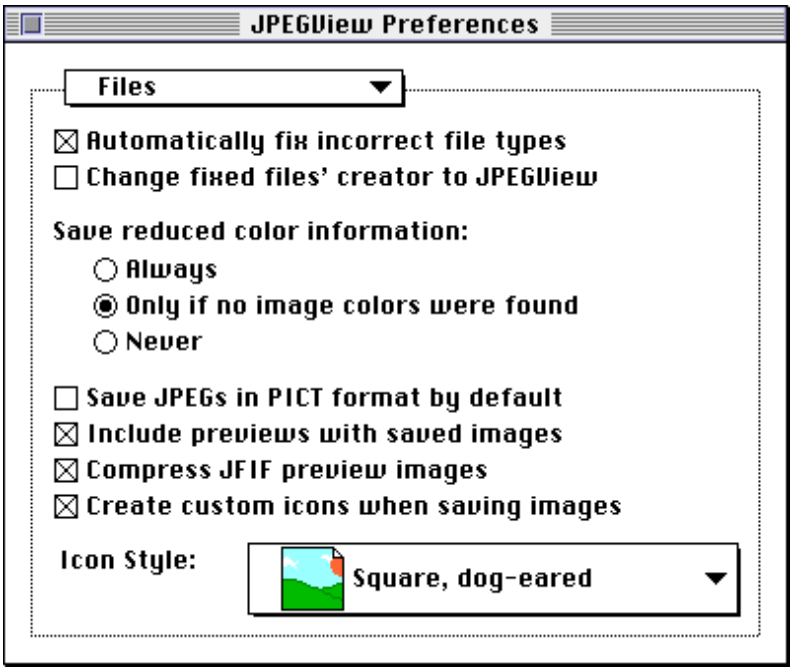
Never created. Eliminates the use of offscreen bitmaps altogether. Not recommended for most users.

Maintain bitmaps at image's original size (default is *off*). This option lets you choose to create all offscreen bitmaps at the image's original size. Normally, JPEGView's bitmaps are just large enough to display what's on your screen, meaning that screen updates are fast but that operations such as resizing, cropping, and preview and icon creation require that the image be decompressed again to make use of the full size and color resolution of the original. With original size bitmaps enabled, however, these operations become very fast because the full-resolution image data is already available. The main downside is that window updates of scaled images are slowed down because the scaling and dithering operations need to be performed to draw the correctly-sized image from the full-size original.

Don't use bitmaps for uncompressed images (default is *off*). This checkbox controls whether JPEGView creates offscreen bitmaps for images that are uncompressed. Because uncompressed images are usually quick to display, it can save memory if you check this box to tell JPEGView that such images don't need the benefits of offscreen bitmaps. Otherwise, uncompressed images will be treated just as any other image, as far as offscreen bitmaps are concerned.

Files preferences

These options control automatic file type fixing and default options in the Save dialog.



The Preferences dialog displaying Files preferences.

Automatically fix incorrect file types (default is *on*). This option is used to enable JPEGView's automatic file type fixing feature, which is simply a quiet means of keeping file types straight without your intervention. With this option checked, for example, opening a JPEG file that is incorrectly identified as a GIF will cause JPEGView to automatically change the file's type to JPEG. Without this option, the file types would remain confused, and could affect the ability of other applications to correctly identify your images.

Change fixed files' creator to JPEGView (default is *off*). This option is an extension of the previous option which causes the creator of any file whose type was automatically fixed to be changed to JPEGView. In practice, what this means is that the image file's icon will become a JPEGView icon, and that the Finder will in the future open JPEGView whenever that image file is double-clicked. On the other hand, leaving this option off will preserve the file's original creator, meaning that images created in other applications will still be associated with their original applications.

Save reduced color information... (default is *Only if no image colors were found*). These settings control how the "Include reduced color set" checkbox is set up when the Save dialog is opened an output format which supports reduced color sets is selected.

Always. Ensures that the "Include reduced color set" checkbox is always selected.

Only if no image colors were found. Selects the "Include reduced color set" checkbox only if the image had no image colors stored in the original data.

Never. Always leaves the "Include reduced color set" option unchecked.

Save in PICT format by default (default is *on*). This option lets you control which format will be chosen by default whenever an image file is saved. Turning this option on causes the PICT format to always be selected when the Save dialog is presented. Otherwise, the format selected in the Save dialog is the same format as that of the original file.

Include previews with saved images (default is *on*). This checkbox controls the initial setting of the "Include preview (thumbnail) image" box in the Save dialog. If this option is enabled, then the corresponding checkbox in the Save dialog will be checked by default whenever the output file format is PICT or JFIF. Otherwise, the "Include preview (thumbnail) image" box will be left off when the Save dialog is presented.

Compress JFIF preview images (default is *on*). This setting controls the use of compression when creating preview images for JFIF files. Turning this option on means that JFIF previews are compressed using JPEG before being stored in the image data; if this option is off, JFIF previews are left uncompressed. Although compressed previews are a more recent addition to the JFIF standard, they are stored in such a way that any JPEG decoder that follows the rules should be able to handle them. For compatibility's sake, however, this option lets you create the old-style previews as well. Also note that PICT previews are *always* compressed by QuickTime, so there is no corresponding option for them.

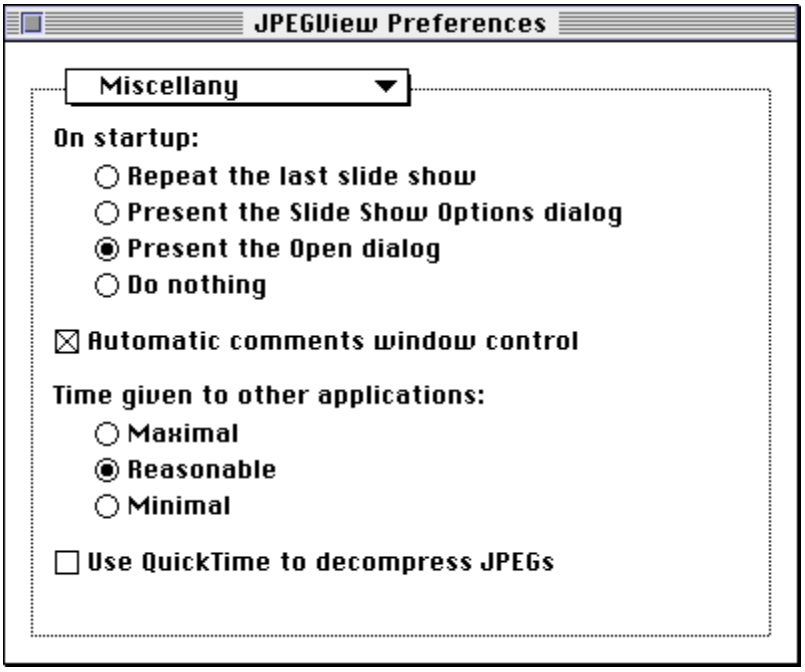
Create custom icons when saving images (default is *off*). This option sets up the "Create custom preview icons" checkbox when the Save dialog is presented. If this option is on, the "Create custom preview icons" option will be automatically selected in the Save dialog; otherwise, the corresponding option in the Save dialog will be initially turned off.

Icon Style (default is *square, dog-eared*). This popup menu allows you to choose your preferred style for custom Finder icons. If you choose square icons, then JPEGView will extract the largest square area from the center of the icon and shrink that down into an icon; in contrast, proportional icons are scaled to dimensions proportional to the original image. In addition, if

you choose a dog-eared style, then JPEGView will fold over the upper-right corner of the icon's image.

Miscellaneous preferences

These preferences settings control the JPEGView's behavior in various other situations.



The Preferences dialog displaying Miscellaneous preferences.

On startup... (default is *Present the Open dialog*). These settings control what actions JPEGView takes when it initially starts up. Note that these actions only apply if you launch JPEGView directly by double-clicking the JPEGView icon. If you launch JPEGView by double-clicking a document icon or by dropping a document icon onto the JPEGView icon, these actions will not be taken.

Repeat the last slide show. Causes the last slide show to be immediately repeated without your intervention.

Present the Slide Show Options dialog. Opens the Slide Show Options dialog, giving you the opportunity to set up a slide show right away.

Present the Open dialog. Opens the Open dialog, allowing you to select an image file right away.

Do nothing. Does nothing at all.

Automatic Comments window control (default is *on*). This option determines who retains control of the Comments floating window. Enabling this option means that JPEGView is in ultimate control of the Comments floating window. In this situation, JPEGView will automatically make the comments visible whenever the active image contains comments, and will automatically hide the Comments window if no comments are available. Turning off this option returns full control of the Comments floating window to you.

Time given to other applications… (default is *Reasonable*). These settings let you specify how much time JPEGView gives to other applications while it is decompressing an image for display.

Maximal. Selects the maximum amount of time JPEGView is willing to give up. Note that this will slow down decompression and display a great deal if there is a lot of other activity going on in your system.

Reasonable. Selects what JPEGView considers to be a reasonable amount of time to give to other applications.

Minimal. Selects the minimum amount of time JPEGView can give to other applications. This will slow down any other activity on your system with the benefit of faster decompression and display times when JPEGView is the active application.

Use QuickTime to decompress JPEGs (default is *on* for 680x0 Macintoshes, and *off* for Power Macintoshes). This option controls whether or not JPEGView uses QuickTime's built-in JPEG decompression to display JPEG images. On 680x0 Macintoshes, QuickTime is a big win if you have it installed, as it is much faster than the Independent JPEG Group's code; however, it is also less forgiving of unusual JPEG images, and occasionally crashes when attempting to decode corrupt data. On Power Macintoshes, the speed difference between the two is minimal, so you might as well use the more robust code.

File Formats and File Types

With so many different types of files floating around—GIF, JPEG, PICT, TIFF, BMP, MacPaint, Startup Screen, etc.—it becomes important to be able to recognize these different file types in the most efficient manner possible, so that the System and its applications aren't bogged down with the task of identifying what a file contains. This chapter discusses this problem and how the Macintosh and JPEGView cooperate to help make the situation as painless as possible.

Identifying different types of files

One obvious solution to the file type identification problem is to simply open up each file, look briefly inside at the contents, and determine the type of file based on what is seen inside. This method relies on the file's *format*, or the order in which data is stored in the file. For example, a JFIF file always begins with a certain sequence of characters which allows it to be easily identified.

Although this method may seem like the most logical way of handling things, it has some serious disadvantages. First, it's slow. Opening a single file and looking inside is not so time-consuming an operation. But when you try to look inside 10, 50, or 100 such files, this becomes a real drag on system performance. In addition, new file formats are always cropping up, and it would be impossible to keep up with the constant stream of new file types.

In response to these two problems, the Finder implements a scheme of file typing, where it assigns each file a four-character file type, which in theory uniquely identifies the contents of the file. This makes for very fast identification: we need only look at this file type to see what sort of file we're looking at—we don't even need to open the file in the first place! Thus, for JPEGView to quickly identify valid image files, it is useful to simply scan for the valid file types: JPEG, JFIF, PICT, GIFf (the extra "f" is added to make it exactly four characters), TIFF, BMPp, PNTG, and SCRN.

Unfortunately, things aren't quite so rosy. Although the Finder maintains a record of the file's type, it does no checking to make sure that this type actually describes the format of the file it purports to. You could give a GIF image a JPEG file type, and the Finder would be none the wiser. This means that applications need to be responsible for maintaining the accuracy of these file types, in order to preserve the usefulness of this file typing scheme.

Where this problem becomes particularly evident is when you are transferring files from another computer which has no file typing mechanism, either via modem or by using *Apple File Exchange*. Although some communications programs automatically check certain characteristics of the file

to help assign the correct type, you will more often than not discover that the nice batch of GIFs you just downloaded aren't correctly typed as GIFf, as they should be.

To help keep things in order, JPEGView provides two mechanisms for correcting file types. The first is the "Scan for Image Files" button in the Open dialog, which scans the current directory for valid image files and fixes the types of those image files whose types were incorrect to begin with. The second method used by JPEGView is the Preferences option to "Automatically fix incorrect file types," which will change incorrect file types for an image files you open with JPEGView. See the chapters *Viewing Images* and *Preferences Settings* for descriptions on how these mechanisms operate.

Making the icons appear

In addition to the file's type, the Finder also stores information about the file's "creator." When used in combination with the file type, a file's creator determines which icon the Finder displays for a given image. For example, a file with a file type of PICT and a creator of JPEGView will show JPEGView's icon for a PICT file. The Finder also uses the creator information to determine which application to launch when you double-click on the file. In the previous example, double-clicking on the PICT file would open JPEGView, which is identified as its creator.

When you fix a file's type in the Open dialog by clicking on the "Scan for Image Files" button, JPEGView also changes the creator of the file to JPEGView. This allows you to just double-click on these fixed-up files to open them with JPEGView. In contrast, the option to quietly and automatically fix incorrect file types for images that you open with JPEGView will only change the type of the file, leaving the option of changing the creator up to your setting of the "Change fixed files' creator to JPEGView" box in the Preferences dialog.

Using the AutoTypers

In order to allow you to easily change the type and creator of a file "by hand," the JPEGView distribution contains a folder with a number of tiny drag-and-drop applications in it. These are the *AutoTypers*—small, single-purpose applications whose sole function is to set the file types and creators of a bunch of files, all in one fell swoop. To use these utilities, simply select the files you wish to modify in the Finder, and then drag the icons onto the icon of the appropriate AutoTyper, releasing the mouse button.

Drag documents onto an AutoTyper icon to set the Finder's file type and creator information by hand.

This operation, known as "drag and drop", tells the Finder to open the AutoTyper and pass it all the files you "dropped" when the mouse button is released. The AutoTyper will then set the creator of each dropped file to JPEGView, and set the type of each dropped file to whichever type is appropriate for the specific AutoTyper. After a second or so, you should see the icons of the files you dropped change to the appropriate JPEGView icons.

JPEGView ships with a total of seven AutoTypers — one for each file format supported. See the next section for a description of each of these formats.

File types supported by JPEGView

JPEGView supports a total of seven file types for viewing: JFIF/JPEG, PICT, GIF, TIFF, BMP, MacPaint, and Startup Screen. The first of these is the JFIF (JPEG File Interchange Format), which is the current standard for exchanging JPEG-encoded image files between different computers. Nearly all JPEG-compressed images that you will find publicly available are in this format. Overall, JFIF is a very bare-bones format, storing essentially the raw JPEG data and perhaps a preview image, but nothing more, and is therefore the smallest of the supported file formats.

The second format—the PICT format—is the Macintosh's standard way of storing data, and has been around in a simpler form ever since the first Macintoshes were produced. With the QuickTime extension installed, any application that recognizes and reads a standard PICT can also read a JPEG-compressed PICT, which is one reason why it is so useful to store files in this format on the Macintosh. Additionally, QuickTime supports a number of other types of compression through QuickTime. Because it contains more information than a JFIF/JPEG file, you can expect PICT files to be slightly larger in size.

The GIF format, originally developed by CompuServe in 1987, is an aging but still popular way of exchanging files between different computers. With the advent of JPEG, which can store much more color information in a much more tightly compressed file, this format is gradually becoming obsolete, but for the moment you will probably find GIF files more easily and find many more of them (and many more poor quality ones) than you will JPEGs.

The TIFF format, developed and maintained by Aldus, is a common but extremely verbose means of storing just about any type of graphic in any order on any system. Because it is so complex, the TIFF specifications divide TIFF functionality into baseline and extended features. JPEGView's TIFF reader supports all of the features demanded by a TIFF baseline reader; in addition, LZW compression with and without prediction is supported as an extension. TIFF files are generally rather large compared with their GIF and JPEG counterparts.

Microsoft's BMP image format seems to be the analog to PICT for Windows machines. Strangely, though, whoever designed the BMP specification must have been from another planet, as everything about BMP is backwards. Image data is stored from bottom to top, and pixels are stored in blue/green/red order, rather than the usual red/green/blue. This means that you will sometimes see BMP images drawn from bottom to top if memory is tight. Compression in BMP files is comparable to compression in standard PICT files, i.e., not terribly great, so BMP images are usually pretty large.

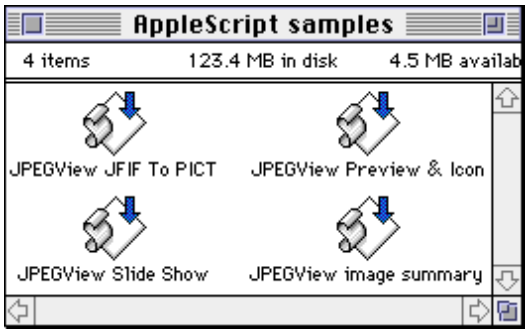
MacPaint images are, oddly enough, a relic from the old MacPaint days of the original Macintoshes that still seems to be with us today. MacPaint files always store black and white images that are in theory one printed page in size. Compression is about the same as PICT images containing the same data.

Finally, the Startup Screen format is the standard file format for startup screens on the Macintosh. There are two flavors of startup screens—black and white, or "classic" Startup Screens, and the newer Mac II, or "PICT0" Startup Screens—both of which are handled transparently by JPEGView. Classic Startup Screens are uncompressed, while PICT0 Startup Screens have the same characteristics as uncompressed PICT images.

Scripting JPEGView

Although JPEGView by itself is a useful tool for image viewing and rudimentary conversions, it becomes even more powerful when used in conjunction with AppleScript. This is because JPEGView is "scriptable", meaning that you can write scripts in AppleScript that can perform any operation you can do with the keyboard or mouse in JPEGView. JPEGView is also "recordable", meaning that you can use the *Script Editor* application to record your actions while running JPEGView, and then play them back later.

JPEGView includes with its original distribution package four potentially useful "droplets", or drag-and-drop script applications, which also serve as basic examples of how to write JPEGView scripts.



The AppleScript samples folder in the JPEGView distribution folder contains four example AppleScript "droplets", or drag-and-drop script applications.

JPEGView includes with its original distribution package four potentially useful "droplets", or drag-and-drop script applications, which also serve as basic examples of how to write JPEGView scripts. The function of each script is described briefly below.

JPEGView Convert to PICT: dropping a bunch of files on this icon will fire up JPEGView and convert each JPEG-compressed image in the bunch to JPEG-compressed PICT format with previews, custom icons, and reduced color sets.

JPEGView Preview & Icons: by dropping files onto this application, you can get JPEGView to create previews and icons for all valid images.

JPEGView Image Summary: using drag and drop with this application, JPEGView will open each image and add a summary line to a new untitled window in the Scriptable Text Editor.

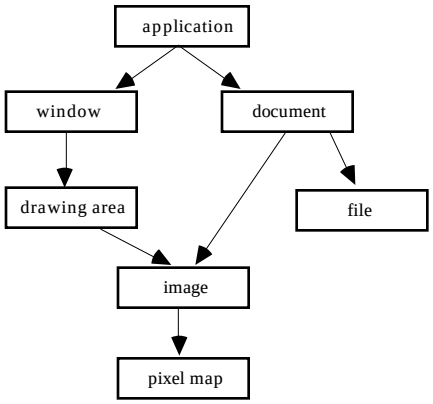
JPEGView Slide Show: dragging and dropping onto this icon will start up a JPEGView slide show of the dropped folders and files.

Of course, you can easily modify these scripts using the *Script Editor* to make them do more precisely what you want. This is probably the best way to get your feet wet in working with the AppleScript-JPEGView combination.

This chapter explains how AppleScript and JPEGView work together to give you scripting capabilities. It does not tell you how to use AppleScript; rather, it assumes some basic understanding of how AppleScript works and of how scripts are written. If you don't have AppleScript, you can find out how to get it in the *JPEGView Questions and Answers* chapter.

The JPEGView Object Hierarchy

In order to write scripts for JPEGView, it is important to understand how JPEGView organizes the objects it manipulates. Here is a crude graphical representation of the JPEGView object hierarchy:



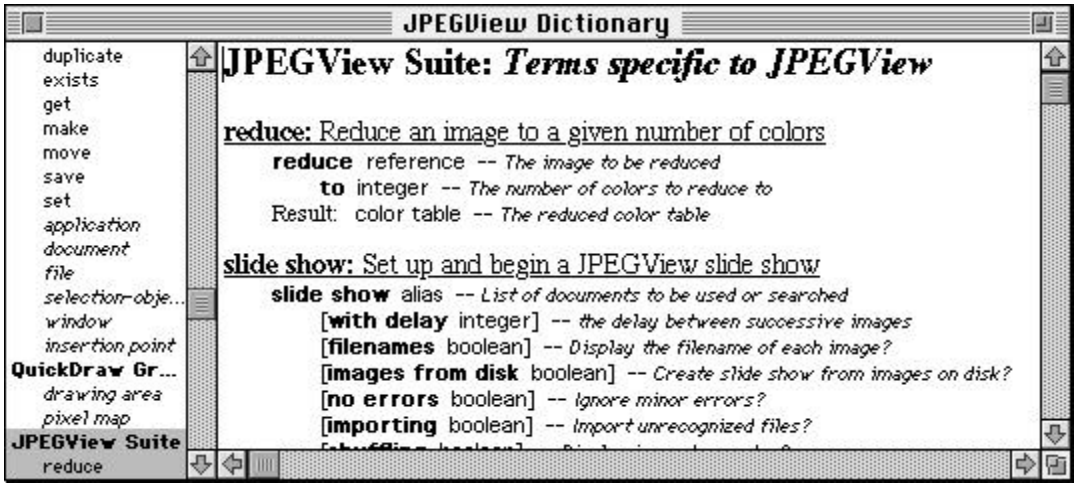
A simple schematic of JPEGView's AppleScript object hierarchy.

At the top of the hierarchy is the application, which serves as the "container" for all the other objects in JPEGView. Inside the application there can be any number of windows and documents. Be careful not to confuse image objects with document objects, however! A document is some abstract object that is being viewed or edited in the application; an image is what the document contains, i.e., the graphic image *inside* the document.

Window objects in general fall into two categories: those that are displaying a document, and those that are not. Windows that are displaying documents contain exactly one document object (since only one document can appear at a time in a given window) and one drawing area object. Windows that are not displaying a document have no documents and no drawing areas associated with them.

Aside from the application object, which can contain many documents and windows, most of JPEGView's objects only contain a single element. That is, windows can only contain one drawing area (if any at all); drawing areas, documents, and files can contain only one image; documents can contain only one file; and images can only contain zero or one pixel map. Although it may have seemed more logical to combine everything into a single object, the current setup fits more smoothly into Apple's definitions for the standard AppleEvent suites.

The best way to see what commands and objects JPEGView supports is to open its dictionary using the *Script Editor's* **Open Dictionary** command:



Opening the JPEGView dictionary from within the Script Editor.

This will produce a list of four suites, or groups, of objects and commands: the Required Suite, the Standard Suite, the QuickDraw Graphics Suite, and the JPEGView Suite. The Required and Standard suites are common to all AppleScript-aware applications, and won't be described here apart from the extensions and limitations imposed by JPEGView.

The QuickDraw Graphics Suite provides definitions for pixel maps and drawing areas, two objects used in JPEGView. However, JPEGView only implements a subset of the full definition of these classes, so be careful if you've seen them in other applications. Finally, the JPEGView Suite contains two events for reducing images to a particular number of colors and for running the slide show; only the latter event is interesting for scripting purposes.

Once you've glanced over the classes, try recording some random actions in JPEGView. This will help you get a feel for how JPEGView handles and manipulates objects. Almost everything you do in JPEGView will appear as a script command in the *Script Editor's* window.

Finally, note that JPEGView only supports the "simple grammar" at this time; that is, you can only refer to objects by number or by name. "Whose" clauses and conditionals are not supported, although the generic selectors such as *every*, *any*, *first*, *last*, etc., will work fine.

Commands in the Required Suite

JPEGView supports all four events defined in Apple's Required Suite; these are the open, print, quit, and run commands.

open: Open the specified object(s). This command is used in JPEGView to open image files. The object it expects is a list of aliases to the files you want to open. If you leave the remaining parameters alone, the image will be opened according to the current preferences settings; however, you can control almost every aspect of the opening procedure through the use of the parameters specified below. Any omitted parameters will take on the values from the preferences settings.

on deepest color / deepest grayscale / largest color / largest grayscale / main—allows you to specify which monitor to open the image on

using full screen always / always if large / on request if large / never—allows you to specify how JPEGView handles the use of full screen windows

expansion true / false—controls whether small images are expanded to maximum size

visible true / false—controls whether the image window is initially visible

with colors image or reduced or system / reduced or image or system / image or system / reduced or system / system—controls the initial colors used for image drawing, in order of preference; that is, "image or reduced or system" means that image colors are preferred, followed by reduced colors, and then standard system colors

with quality very high / high / normal—specifies the quality to be used in drawing the opened images

bitmaps true / false—controls whether bitmaps are required for the images

comments true / false—specifies whether the Comments floating window is automatically displayed for images that have comments

type fixing true / false—controls whether file types are automatically fixed

changing creators true / false—controls whether creators are changed along with file types

no errors true / false—tells JPEGView whether or not it should display error messages; defaults to false

Note that since JPEGView only accepts lists of aliases as parameters to the *open* command, you must manually coerce files or filenames to aliases before passing them to the *open* command. So, be careful to use "open {file as alias}" and not just "open {file}".

print: Print the specified object(s) or file(s). Specifying either files or document objects instructs JPEGView to start printing the appropriate images.

quit: Quit application. Quits JPEGView.

run: Sent to an application when it is double-clicked. This function performs the startup action specified in the preferences; normally only used by the Finder when it launches JPEGView.

Commands in the Standard Suite

The Standard Suite provides the basic commands for manipulating objects in JPEGView, and JPEGView supports all of Apple's recommended functions in this suite.

close: Close an object. In JPEGView, only windows and documents can be closed. Closing a document also closes its corresponding window. The only additional non-standard parameter is:

restore colors true / false—controls whether colors are automatically restored after the last image is closed

count: Return the number of elements of a particular class within an object. This function is applicable in the standard way to every object in the JPEGView object hierarchy.

data size: Return the size in bytes of an object. This function is supported in the standard way for every object and for every property of every object in the JPEGView object hierarchy.

delete: Delete an element from an object. This command only works for windows, documents, and pixel maps. For windows and documents, it provides the same functionality as the *close* command; for pixel maps, it disposes of any offscreen bitmap associated with an image.

duplicate: Duplicate object(s). In JPEGView, only documents and windows can be duplicated, and in the case of windows, only windows that contain documents can do it. In addition, the *du-*

plicate command supports a **visible** parameter, whose function is identical to the **visible** parameter in the *open* command from the Required Suite.

exists: Verify if an object exists. This is supported in the standard way for all objects in the JPEGView object hierarchy.

get: Get the data for an object. This function is supported in the standard way for all objects and for all properties of objects in the JPEGView object hierarchy.

make: Make a new element. Only windows can be made in JPEGView, and then only static windows, which include the three floating windows, the About box, the Help window, and the Preferences and Slide Show Options dialog. To create a given new window, simply specify its contents property in the "with properties" parameter.

move: Move object(s) to a new location. Both documents and windows can be moved in JPEGView. This allows you to move windows (document windows in the case of documents) in front of or behind other windows in the application.

save: Save an object. In JPEGView, only documents and windows containing documents can be saved. In addition to the standard parameters, JPEGView adds the following three optional parameters:

preview true / false—specifies whether to create a preview of the document

reduced colors true / false—specifies whether to save a reduced color set with the document

icons true / false—specifies whether to create custom icons for the document

set: Set an object's data. This command works in JPEGView only for data contained in the properties of an object. The data of an actual object cannot be directly set.

Objects classes in the Standard Suite

application: the JPEGView application. This object sits at the top of the object hierarchy and always exists. It can contain any number of windows and any number of documents. The application properties supported by JPEGView are as follows:

clipboard (list of anything)—non-modifiable—since JPEGView does not support copying and pasting, this property always returns an empty list

frontmost (boolean)—non-modifiable—this property is true if JPEGView is the frontmost application

name (string)—non-modifiable—the name of the application, i.e., "JPEGView"

version (integer)—non-modifiable—the version number, specified as (major version x 256) + (minor version); for example, version 3.0 is (3 x 256 + 0) = 768

document: a document. This object serves as the container for all images that are manipulated in JPEGView. Documents always contain exactly one file and one image. The supported document properties are:

modified (boolean)—non-modifiable—this property is true if the image has been modified; in JPEGView, this property always returns false

name (boolean)—modifiable—this is the name of the document

selection (boolean)—non-modifiable—this property points to a selection object whose **bounds** property describes the current selection in this document

file: a file. This object specifies information about the file used to store a document. File objects always contain exactly one image in JPEGView. Supported file properties include:

stationery (boolean)—non-modifiable—this property is true if the file is a stationery pad document; since JPEGView does not support stantionery pad documents, this will always return false

name (boolean)—modifiable—this contains the name of the file, specified as a full pathname; for example, a file named "Example.PICT" in the folder "Temporary" on the volume "My Hard Drive" would have a pathname of "My Hard Drive:Temporary:Example.PICT"

window: a window. Window objects describe properties of the visible (and invisible) windows on the screen. Windows that contain a document will contain exactly one document object and one drawing area object. All other windows will contain no elements. The window properties supported in JPEGView are:

bounds (bounding rectangle)—modifiable—this property describes the position and size of the window on the screen; if you want to change the size of a window that contains an image, change the **bounds** property of the *drawing area* instead

closeable (boolean)—non-modifiable—this property is true if the window has a close box; only the Preferences and Slide Show Options dialogs are missing a close box

titled (boolean)—non-modifiable—this property is true if the window has a title bar; this is true of all of JPEGView's windows

index (integer)—non-modifiable—this is the position of the window in the list; for instance, the frontmost window would return 1, the second window would return 2, etc.

floating (boolean)—non-modifiable—this property is true if the window is a floating window; JPEGView's Statistics, Comments, and Colors windows all qualify as floating

modal (boolean)—non-modifiable—this property is true if the window is a modal window; none of JPEGView's windows are modal

resizable (boolean)—non-modifiable—this property is true if the window is resizable; in JPEGView, only image windows are resizable

zoomable (boolean)—non-modifiable—this property is true if the window can be zoomed; image windows and the Statistics floating window can be zoomed in JPEGView

zoomed (boolean)—modifiable—this property is true if the window is zoomed to a non-standard state

name (boolean)—modifiable—this is the title of the window

selection (boolean)—non-modifiable—this property points to a selection object whose **bounds** property describes the current selection in this window

visible (boolean)—modifiable—this property is true if the window is currently visible on the desktop

full screen (boolean)—modifiable—this property is true if the window is a full-screen window

contents (image / full screen image / image statistics / image comments / screen colors / help text / slide show options / preferences / about box / slide show controls / about help)—non-modifiable—this property describes briefly what the window is displaying

Object classes in the QuickDraw Graphics Suite

drawing area: a drawing area in a window. This class serves to represent the area of a window inside which images are drawn. In JPEGView, members of the drawing area class always contain exactly one image. The supported properties of the drawing area object include:

bounds (bounding rectangle)—modifiable—this property describes the area inside of which the image is drawn; for normal windows, this coincides with the bounds of the window itself; for full-screen windows, this represents the area that the image is drawn in; set this property if you wish to change the size of the displayed image

colors (system colors / grayscales / image colors / two pass color reduction)—modifiable—this property specifies the current color set used for drawing the image; if two pass color reduction is requested and the color reduction algorithm has not yet been applied to the image, it will be done immediately

color table (color table)—modifiable—this property contains the current full color table of the image; in most cases, it is easier to work with the colors property instead

scale (fixed)—modifiable—this property describes the current scaling factor used in drawing the image; set this as an alternative to manipulating the bounds property directly

quality (very high / high / normal)—modifiable—this property specifies the current drawing quality applied to the image

transfer mode (no dithering / dithering)—modifiable—this property specifies how the image is drawn, either with dithering or without

pixel map: a pixel map. This object is used to describe the offscreen buffer that JPEGView uses for fast window updating. The two pixel map properties that are supported are:

bounds (bounding rectangle)—non-modifiable—this is the size of the offscreen buffer

pixel depth (small integer)—non-modifiable—this is the depth, or number of colors specified as *n*, where 2^n is the number of colors available

Commands in the JPEGView Suite

The commands in the JPEGView suite provide access to two of JPEGView's more important features: two-pass color reduction, and slide shows.

reduce: Reduce an image to a given number of colors. This command, which applies to image objects only, allows you to calculate an optimal color set reduced to a specified number of colors. The result is a color table object which can be used to set the color table property of a drawing area object. If you want to set an image's colors to the standard reduced color set in JPEGView, set the **colors** property of the *drawing area* instead. Normally, this command is not used in scripts.

slide show: Set up and begin a JPEGView slide show. This powerful function allows you to specify a folder or a set of files to view in a JPEGView slide show, along with a number of options that control how the images are to be displayed. The options parameters are, naturally, optional, with their values taken from the last set of user preferences if not specified by the command. These options are:

with delay *seconds*—specifies the delay between successive images, in seconds

filenames true / false—controls whether filenames are displayed along with each image

images from disk true / false—specifies whether the slide show is disk-based or memory-based

no errors true / false—specifies whether error messages are suppressed during the slide show

importing true / false—controls whether all files found are tried, or whether only files whose file types are correct according to the Finder are opened for display

shuffling true / false—controls whether the slides are shuffled before each repetitition of the slide show

looping true / false—specifies whether the slide show repeats indefinitely (loops)

recursive scan true / false—controls whether a disk-based scan for files scans subfolders recursively

user control true / false—specifies whether the slide show is user-controlled or automatic

offscreen drawing true / false—controls whether the next slide is prepared offscreen

auto comments true / false—controls whether the Comments floating window appears automatically when comments are found in an image

hidden windoids true / false—controls whether floating windows (windoids) are automatically hidden at the start of the slide show

hidden controls true / false—controls whether the Slide Show Controls floating window is hidden at the start the slide show

screen saver true / false—specifies whether to run in screen saver mode; in this mode, JPEGView ensures that it is always the frontmost application, and immediately aborts the slide show and quits whenever the mouse is moved or clicked, or a key is pressed

Object classes in the JPEGView Suite

image: an image graphic. Image objects in JPEGView describe the fundamental information about the image itself, not about the way in which the image is drawn on your screen. Images can contain zero or one pixel maps, depending on whether the image has an associated offscreen buffer. The properties supported by the image class are:

bounds (bounding rectangle)—modifiable—this property specifies the current bounding rectangle of the image; to crop an image, set this property to a rectangle smaller than the original bounds property

comments (string)—non-modifiable—this property contains the comments field, if any, found with the image

display time (integer)—non-modifiable—this property specifies the amount of time needed to decompress and draw the full image, in milliseconds; it is only set after the image has been decompressed and drawn the first time

format (type class)—non-modifiable—this property tells you which format the image is in, according to the file type codes described in the *File Formats and File Types* chapter, i.e., PICT, JPEG, or GIFf.

compression (type class)—non-modifiable—this property specifies a code indicating the type of compression used on the image

has image colors (boolean)—non-modifiable—this property is true if the image has its own internally-defined color table

has reduced colors (boolean)—non-modifiable—this property is true if the image has a color set produced by two-pass color reduction

image colors (color table)—non-modifiable—this property contains the full image color set

compressed (boolean)—non-modifiable—this property is true if the image is compressed

aborted (boolean)—non-modifiable—this property is true if the last decompression of the image was interrupted in some way

banded (boolean)—non-modifiable—this property is true if the full image is actually several smaller horizontal strips connected together

corrupt (boolean)—non-modifiable—this property is true if the image data has been found to be invalid in some way

cropped (boolean)—non-modifiable—this property is true if the image has been cropped in any way

original bounds (bounding rectangle)—non-modifiable—this property contains the full-size original bounds of the image

pixel depth (small integer)—non-modifiable—this property specifies the number of colors in the image as *n*, where 2^n is the total number of colors in the image

reduced colors (color table)—non-modifiable—this property contains the full reduced color set

length (integer)—non-modifiable—this property contains the length of the image, in bytes

The following classes are not really members of the JPEGView object hierarchy; rather, they are really just placeholders for the appropriate file format that you can specify with the **save** command.

picture: an image in PICT format..

JFIF: an image in JFIF format.

startup screen: an image in SCRN format.

GIF: an image in GIF format.

TIFF: an image in TIFF format.

BMP: an image in BMP format.

MacPaint: an image in MacPaint format.

General JPEG Questions and Answers

[These questions and answers are taken almost verbatim from the text of the JPEG Frequently Asked Questions list, maintained by Tom Lane. A lot of the information contained here does not pertain very specifically to JPEGView, but rather to the current state of JPEG standards and to the Independent JPEG Group's free software. The latter consists of two general conversion programs: cjpeg, which converts files to JPEG; and djpeg, which converts files from JPEG. (*JPEG Convert*, a Macintosh version of cjpeg/djpeg, is available from many archive sites.) Note that references to compression quality factors in this section refer to cjpeg and djpeg, which use a scale running from 0 to 100, rather than from 0.00 to 4.00 as does Apple's QuickTime. For more information about JPEG in general or the free JPEG software in particular, contact the Independent JPEG Group at jpeg-info@uunet.uu.net.]

What is JPEG?

JPEG (pronounced "jay-peg") is a standardized image compression mechanism. JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard.

JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes. It works well on photographs, naturalistic artwork, and similar material; not so well on lettering, simple cartoons, or line drawings. JPEG handles only still images, but there is a related standard called MPEG for motion pictures.

JPEG is "lossy," meaning that the decompressed image isn't quite the same as the one you started with. (There are lossless image compression algorithms, but JPEG achieves much greater compression than is possible with lossless methods.) JPEG is designed to exploit known limitations of the human eye, notably the fact that small color details aren't perceived as well as small details of light-and-dark. Thus, JPEG is intended for compressing images that will be looked at by humans. If you plan to machine-analyze your images, the small errors introduced by JPEG may be a problem for you, even if they are invisible to the eye.

A useful property of JPEG is that the degree of lossiness can be varied by adjusting compression parameters. This means that the image maker can trade off file size against output image quality. You can make *extremely* small files if you don't mind poor quality; this is useful for applications like indexing image archives. Conversely, if you aren't happy with the output quality at the default compression setting, you can jack up the quality until you are satisfied, and accept lesser compression.

Why use JPEG?

There are two good reasons: to make your image files smaller, and to store 24-bit-per-pixel color data instead of 8-bit-per-pixel data.

Making image files smaller is a big win for transmitting files across networks and for archiving libraries of images. Being able to compress a 2 Mbyte full-color file down to 100 Kbytes or so makes a big difference in disk space and transmission time!

If your viewing software doesn't support JPEG directly, you'll have to convert JPEG to some other format for viewing or manipulating images. Even with a JPEG-capable viewer, it takes longer to decode and view a JPEG image than to view an image of a simpler format such as GIF. Thus, using JPEG is essentially a time/space tradeoff: you give up some time in order to store or transmit an image more cheaply.

It's worth noting that when network or phone transmission is involved, the time savings from transferring a shorter file can be greater than the extra time needed to decompress the file.

The second fundamental advantage of JPEG is that it stores full color information: 24 bits/pixel (16 million colors). GIF, the other image format widely used on the networks, can only store 8 bits/pixel (256 or fewer colors). GIF is reasonably well matched to inexpensive computer displays—most run-of-the-mill PCs can't display more than 256 distinct colors at once. But full-color hardware is getting cheaper all the time, and JPEG images look much better than GIFs on such hardware. Within a couple of years, 8-bit GIF will seem as obsolete as black-and-white MacPaint format does today. Furthermore, for reasons detailed below, JPEG is far more useful than GIF for exchanging images among people with widely varying display hardware. Hence JPEG is considerably more appropriate than GIF for use as an image exchange standard.

A lot of people are scared off by the term "lossy compression". But when it comes to representing real-world scenes, *no* digital image format can retain all the information that impinges on your eyeball. In comparison with the real-world scene, JPEG loses far less information than GIF. The technical meaning of "lossy" has nothing to do with this, though; it refers to loss of information over repeated compression cycles, a problem that you may or may not care about. (If you do, see below.)

When should I use JPEG, and when should I stick with GIF?

JPEG is not going to displace GIF entirely; for some types of images, GIF is superior in image quality, file size, or both. One of the first things to learn about JPEG is which kinds of images to apply it to.

Generally speaking, JPEG is superior to GIF for storing full-color or gray-scale images of "realistic" scenes; that means scanned photographs and similar material. Any continuous variation in color, such as occurs in highlighted or shaded areas, will be represented more faithfully and in less space by JPEG than by GIF.

GIF does significantly better on images with only a few distinct colors, such as line drawings and simple cartoons. Not only is GIF lossless for such images, but it often compresses them more than JPEG can. For example, large areas of pixels that are all *exactly* the same color are compressed very efficiently indeed by GIF. JPEG can't squeeze such data as much as GIF does without introducing visible defects. (One implication of this is that large single-color borders are quite cheap in GIF files, while they are best avoided in JPEG files.)

Computer-drawn images (ray-traced scenes, for instance) usually fall between photographs and cartoons in terms of complexity. The more complex and subtly rendered the image, the more likely that JPEG will do well on it. The same goes for semi-realistic artwork (fantasy drawings and such).

JPEG has a hard time with very sharp edges: a row of pure-black pixels adjacent to a row of pure-white pixels, for example. Sharp edges tend to come out blurred unless you use a very high quality setting. Edges this sharp are rare in scanned photographs, but are fairly common in GIF files: borders, overlaid text, etc. The blurriness is particularly objectionable with text that's only a few pixels high. If you have a GIF with a lot of small-size overlaid text, don't JPEG it.

Plain black-and-white (two level) images should never be converted to JPEG; they violate all of the conditions given above. You need at least about 16 gray levels before JPEG is useful for gray-scale images. It should also be noted that GIF is lossless for gray-scale images of up to 256 levels, while JPEG is not.

If you have a large library of GIF images, you may want to save space by converting the GIFs to JPEG. This is trickier than it may seem—even when the GIFs contain photographic images, they are actually very poor source material for JPEG, because the images have been color-reduced. Non-photographic images should generally be left in GIF form. Good-quality photographic GIFs can often be converted with no visible quality loss, but only if you know what you are doing and you take the time to work on each image individually. Otherwise you're likely to lose a lot of image quality or waste a lot of disk space…quite possibly both. There's more information below about GIF to JPEG conversion.

How well does JPEG compress images?

Very well indeed, when working with its intended type of image (photographs and suchlike). For full-color images, the uncompressed data is normally 24 bits/pixel. The best known lossless compression methods can compress such data about 2:1 on average. JPEG can typically achieve 10:1 to 20:1 compression without visible loss, bringing the effective storage requirement down to 1 to 2 bits/pixel. 30:1 to 50:1 compression is possible with small to moderate defects, while for very-low-quality purposes such as previews or archive indexes, 100:1 compression is quite feasible. An image compressed 100:1 with JPEG takes up the same space as a full-color one-tenth-scale thumbnail image, but it retains much more detail than such a thumbnail.

Gray-scale images do not compress by such large factors. Because the human eye is much more sensitive to brightness variations than to hue variations, JPEG can compress hue data more heavily than brightness (gray-scale) data. A gray-scale JPEG file is generally only about 10%-25% smaller than a full-color JPEG file of similar visual quality. But the uncompressed gray-scale data is only 8 bits/pixel, or one-third the size of the color data, so the calculated compression ratio is much lower. The threshold of visible loss is often around 5:1 compression for gray-scale images.

The exact threshold at which errors become visible depends on your viewing conditions. The smaller an individual pixel, the harder it is to see an error; so errors are more visible on a computer screen (at maybe 70 dots/inch) than on a high-quality color printout (300 or more dots/inch). Thus a higher-resolution image can tolerate more compression…which is fortunate considering it's much bigger to start with. The numbers quoted above are typical for screen viewing. Also note that the threshold of visible error varies somewhat across images.

What's all this hoopla about color quantization?

Most people don't have full-color (24 bit per pixel) display hardware. Typical display hardware stores 8 or fewer bits per pixel, so it can display 256 or fewer distinct colors at a time. To display a full-color image, the computer must choose an appropriate set of representative colors and map the image into these colors. This process is called "color quantization". (This is something of a misnomer; "color selection" or "color reduction" would be a better term. JPEGView calls it "color reduction".)

Clearly, color quantization is a lossy process. It turns out that for most images, the details of the color quantization algorithm have much more impact on the final image quality than do any errors introduced by JPEG itself (except at the very lowest JPEG quality settings). Making a good color quantization algorithm is a black art, and no single algorithm is best for all images.

Since JPEG is a full-color format, converting a color JPEG image for display on 8-bit-or-less hardware requires color quantization. The speed and image quality of a JPEG viewer running on such hardware are largely determined by its quantization algorithm. You'll see great variation in image quality among viewers on 8-bit displays, much more than occurs on 24-bit displays.

On the other hand, a GIF image has already been quantized to 256 or fewer colors. (A GIF does have a specific number of colors in its palette, and the format doesn't allow more than 256 palette entries.) GIF has the advantage that the image maker precomputes the color quantization, so viewers don't have to; this is one of the things that make GIF viewers faster than JPEG viewers. But this is also the *disadvantage* of GIF: you're stuck with the maker's quantization. If the maker quantized to a different number of colors than what you can display, you'll either waste display capability or have to quantize again to further reduce the number of colors (which results in much poorer image quality than if you had quantized once from a full-color image). Furthermore, if the maker didn't use a high-quality color quantization algorithm, you're out of luck—the image is ruined.

For this reason, JPEG promises significantly better image quality than GIF for all users whose machines don't match the image maker's display hardware. JPEG's full color image can be quantized to precisely match the viewer's display hardware. Furthermore, you will be able to take advantage of future improvements in quantization algorithms (there is a lot of active research in this area), or purchase better display hardware, to get a better view of JPEG images you already have. With a GIF, you're stuck forevermore with what was sent.

A growing number of people have better-than-8-bit display hardware already: 15-bit "hi-color" PC displays, true 24-bit displays on workstations and Macintoshes, etc. For these people, GIF is already obsolete, as it cannot represent an image to the full capabilities of their display. JPEG images can drive these displays much more effectively.

In short, JPEG is an all-around better choice than GIF for representing images in a machine-independent fashion.

What are some rules of thumb for converting GIF images to JPEG?

Converting GIF files to JPEG is a tricky business—you are piling one set of limitations atop a quite different set, and the results can be awful. Certainly a JPEG made from a GIF will never be as good as a JPEG made from true 24-bit color data. But if what you've got is GIFs, and you need to save space, here are some hints for getting the best results.

With care and a clean source image, it's often possible to make a JPEG of quality equivalent to the GIF. This does *not* mean that the JPEG looks identical to the GIF—it probably won't on an 8-bit display, because the color quantization process used to display the JPEG won't exactly match the GIF's quantization. (See previous item for more about that.) But given a good viewer, the JPEG will look as good as the GIF. Some people claim that on 24-bit displays, a carefully converted JPEG can look better than the GIF source, because dither patterns have been eliminated. (More about dithering in a moment.)

On the other hand, JPEG conversion *will* degrade an unsuitable image or one that is converted carelessly. If you are not willing to take the amount of trouble suggested below, you're much better off leaving your GIF images alone. Simply cranking the JPEG quality setting up to a very high value wastes space (which defeats the whole point of the exercise…) and some images will be degraded anyway.

The first rule is never to convert an image that's not appropriate for JPEG. Large, high-visual-quality photographic images are usually the best material. And they take up lots of space in GIF form, so they offer significant potential space savings. (A good rule of thumb is not to bother converting any GIF that's much under 100 Kbytes; the potential space savings isn't worth the hassle.)

The second rule is to look at each JPEG, to make sure you are happy with it, before throwing away the corresponding GIF; this will give you a chance to re-do the conversion with a higher quality setting if necessary. Also compare the file sizes—if the image isn't suitable JPEG material, a JPEG file of reasonable quality may come out *larger* than the GIF.

The third rule is to get rid of the border. Many people have developed an odd habit of putting a large single-color border around a GIF image. While useless, this is nearly free in terms of storage cost in GIF files. It is *not* free in JPEG files, either in storage space or in decoding time; and the sharp border boundary can create visible artifacts ("ghost" edges). Furthermore, when viewing a bordered JPEG on an 8-bit display, the quantizer will think the border color is important because there's so much of it, and hence will waste color palette entries on the border, thus actually reducing the displayed quality of the main part of the image! So do yourself a favor and crop off any border before JPEGing.

Gray-scale images usually convert without much problem. When using cjpeg, be sure to specify -gray. (By default, cjpeg treats GIFs as color files; this works but wastes space and time for gray-scale data.) Quality settings around the default (75) are usually fine.

Color images are much trickier. Color GIFs of photographic images are usually "dithered" to fool your eye into seeing more than the 256 colors that GIF can actually store. If you enlarge the image, you will find that adjacent pixels are often of significantly different colors; at normal size the eye averages these pixels together to produce the illusion of an intermediate color value. The trouble with dithering is that, to JPEG, it looks like high-spatial-frequency color noise; and JPEG can't compress noise very well. The resulting JPEG file is both larger and of lower image quality than what you would have gotten from JPEGing the original full color image (if you had it). To get around this, you need to "smooth" the GIF image before compression. Smoothing averages together nearby pixels, thus approximating the color that you thought you saw anyway, and in the process getting rid of the rapid color changes that give JPEG trouble. Proper use of smoothing will both reduce the size of the compressed file and give you a better-looking output image than you'd get without smoothing.

With the free JPEG software (or programs based on it), a simple smoothing capability is built in. Try "-smooth 10" or so when converting GIFs. Values of 10 to 25 seem to work well for high-quality GIFs. Heavy-handed dithering may require larger smoothing factors. (If you can see regular fine-scale patterns on the GIF image even without enlargement, then strong smoothing is

definitely called for.) Too large a smoothing factor will blur the output image, which you don't want. If you are an image processing wizard, you can also do smoothing with a separate filtering program, but appropriate use of such tools is beyond the scope of this article.

Quality settings around 85 (a bit higher than default) usually work well when converting color GIFs, assuming that you've picked a good smoothing factor. You may need to go higher if you can't hide the dithering pattern with a reasonable smoothing factor. Really badly dithered GIFs are best left as GIFs.

Don't expect JPEG files converted from GIFs to be as small as those created directly from full-color originals. You won't be able to smooth away all of the dithering noise without ruining the image, and this noise wastes space. Typically, a good-quality converted JPEG will be 1/2 to 1/3rd the size of the GIF file, not 1/4th as suggested above. (If the JPEG comes out much more than half the size of the GIF, this is a good sign that the image shouldn't be converted at all.)

The upshot of all this is that "cjpeg -quality 85 -smooth 10" is probably a good starting point for converting color GIFs. But if you care about the image, you'll want to check the results and maybe try a few other settings. Blindly converting a large GIF library at this or any other setting is a recipe for disaster.

Does loss accumulate with repeated compression/ decompression?

It would be nice if, having compressed an image with JPEG, you could decompress it, manipulate it (crop off a border, say), and recompress it without any further image degradation beyond what you lost initially. Unfortunately *this is not the case*. In general, recompressing an altered image loses more information. Hence it's important to minimize the number of generations of JPEG compression between initial and final versions of an image.

It turns out that if you decompress and recompress an image at the same quality setting first used, little or no further degradation occurs. (Counterintuitively, this works better the lower the quality setting. But you must use *exactly* the same setting, or all bets are off.) This means that you can make local modifications to an image without material degradation of other areas of the image.

Unfortunately, cropping doesn't count as a local change! JPEG processes the image in small blocks, and cropping usually moves the block boundaries, so that the image looks completely different to JPEG. You can take advantage of the low-degradation behavior if you are careful to crop the top and left margins only by a multiple of the block size (typically 16 pixels), so that the remaining blocks start in the same places.

The bottom line is that JPEG is a useful format for archival storage and transmission of images, but you don't want to use it as an intermediate format for sequences of image manipulation steps. Use a lossless format (uncompressed PICT, TIFF, etc) while working on the image, then JPEG it when you are ready to file it away. Aside from avoiding degradation, you will save a lot of compression/decompression time this way…

Why all the argument about file formats?

Strictly speaking, JPEG refers only to a family of compression algorithms; it does not refer to a specific image file format. The JPEG committee was prevented from defining a file format by turf wars within the international standards organizations.

Since we can't actually exchange images with anyone else unless we agree on a common file format, this leaves us with a problem. In the absence of official standards, a number of JPEG program writers have just gone off to "do their own thing", and as a result their programs aren't compatible with anybody else's.

The closest thing we have to a standard JPEG format is some work that's been coordinated by people at C-Cube Microsystems. They have defined two JPEG-based file formats:

- JFIF (JPEG File Interchange Format), a "low-end" format that transports pixels and not much else.
- TIFF/JPEG, aka TIFF 6.0, an extension of the Aldus TIFF format. TIFF is a "high-end" format that will let you record just about everything you ever wanted to know about an image, and a lot more besides. TIFF is a lot more complex than JFIF, and is generally less transportable, because different vendors have often implemented slightly different and incompatible subsets of TIFF.

Both of these formats were developed with input from all the major vendors of JPEG-related products; it's reasonably likely that future commercial products will adhere to one or both standards.

JFIF has emerged as the de-facto standard. JFIF is simpler than TIFF and is available now; the TIFF 6.0 spec has only recently been officially adopted, and it is still unusably vague on some crucial details. Even when TIFF/JPEG is well defined, the JFIF format is likely to be a widely supported lowest common denominator; TIFF/JPEG files may never be as transportable.

In the Macintosh world, all images are supposed to appear in Apple's standard PICT format. QuickTime's JPEG compressor actually produces a JFIF-compatible data sequence with a PICT "wrapper" around it. Conversion between PICT/JPEG and JFIF is thus a fairly simple matter of adding or removing the "wrapper". JPEGView and several other Macintosh programs can do this for you. But non-Macintosh computers don't know anything about PICT, so be sure to convert a PICT/JPEG file to JFIF format before giving it to anyone who uses a non-Macintosh computer.

Just to muddy the waters, there are a couple of proprietary JPEG-based file formats to worry about. One that you are likely to find on some BBS systems is Handmade Software, Inc.'s "HSI" format. There's no way to read HSI files except by using HSI's software—which runs only on PCs. About all you can do is try to persuade the BBS operator to convert to the standardized, public-domain JFIF format.

Isn't there a lossless JPEG?

There's a great deal of confusion on this subject. The JPEG committee did define a truly lossless compression algorithm (i.e., one that guarantees the final output is bit-for-bit identical to the original input). However, this lossless mode has almost nothing in common with the regular lossy JPEG algorithm, and it offers much less compression. At present, very few implementations of lossless JPEG exist.

Lossless JPEG typically compresses full-color data by around 2:1. Lossless JPEG works well only on continuous-tone images; it does not provide useful compression of palette-color images or low-bit-depth images. (The JBIG standard is considered superior to lossless JPEG for images of less than 6 bits/sample.)

Cranking a regular JPEG implementation up to its maximum quality setting *does not* get you a lossless image. Even at the maximum possible quality setting, regular JPEG is not lossless, be-

cause it is subject to roundoff errors in various calculations. Roundoff errors are nearly always too small to be seen, but they will accumulate if you put the image through multiple cycles of compression.

Many implementations won't even let you get to the maximum possible setting, because it's such an inefficient way to use regular JPEG. With the IJG JPEG software, for example, you have to say not only "-quality 100" but also "-sample 1x1" to eliminate all deliberate loss of information. The resulting files are far larger and of only fractionally better quality than files generated at more reasonable settings. If you really need lossless storage, don't try to approximate it with regular JPEG.

JPEGView Questions and Answers

Why can't my friends read the JPEG files I give them?

The problem with transferring JPEG files from the Macintosh to another system is twofold. First, most Macintosh JPEG files are stored as QuickTime PICTs, rather than as standard JFIF files. So, if you're running into difficulties, first make sure you use JPEGView to save any JPEG image as a JFIF image. The second problem has to do with file transfers; if you save custom icons or custom color sets with your JFIF image, it creates resources in the JFIF file that might cause problems when transferring those files. To be absolutely safe, make sure that you turn off custom icons and custom color sets before saving your JFIF file (previews are still okay, however).

Why doesn't the JPEGView JFIF Preview icon show up when I start my system?

The JPEGView JFIF Preview extension isn't really a standard extension, in that it doesn't execute any code at startup. Rather, it is a QuickTime component, which means that QuickTime looks for it in the Extensions folder, and automatically loads and uses it whenever it needs to view a JFIF preview. Since it doesn't execute anything at startup, the extension doesn't display an icon at the bottom of your screen. However, it is still working to provide access to JFIF previews.

Why doesn't JPEGView have scroll bars?

JPEGView doesn't use scroll bars for two very good reasons: first, with scroll bars you have to have enough memory to buffer the entire decompressed image at full resolution, which makes the memory requirements phenomenally high for 24-bit JPEG images; and second, JPEGView implements a **Crop & Zoom** feature which provides the same functionality as scroll bars but with several important advantages. The section in the *Hints and Tips* chapter entitled "Make good use of the **Crop & Zoom** feature" explains these advantages in more detail.

Why does JPEGView crash when loading corrupted JPEG images?

This problem with terrible crashes was a result of loose error checking in the QuickTime 1.0 JPEG decoder. This is not—I repeat—not my fault! The solution is quite simply to upgrade to the latest

version of QuickTime (version 1.6.1 as of December 1993), which is far more robust in handling damaged JPEGs. And yes, there are some images out there that are bound to crash the new version on occasion. Nobody's perfect!

Where can I get the latest version of QuickTime?

Apple's plan for QuickTime seems to include making it as widely available as possible. If you have ftp access, just ftp to ftp.apple.com, login as "anonymous", and give your email address as a password. The latest version should be in the /dts/mac/sys.soft/quicktime directory. Otherwise, you may want to check with your local Apple dealer (bring a blank floppy with you), who might be willing to copy it for you. Also, check your local BBS systems.

Where can I get a copy of AppleScript?

Unlike QuickTime, AppleScript does not come absolutely free. At present, the only two viable avenues for obtaining AppleScript are by ordering it from APDA, in which case it will cost you $20 plus tax and shipping, or by picking up the very recent and very well-written The *Tao of AppleScript*, written by Derrick Schneider and published by Hayden Books. This latter option is probably the best for those just getting started with AppleScript, as it includes not only a good introduction to the AppleScript commands and functionality, but also a copy of the AppleScript extension and some scripting software for $25.

Why can't JPEGView convert between formats other than JFIF and QuickTime?

Why can't JPEGView create new images?

Why can't JPEGView map my image onto a complex three-dimensional object?

JPEGView is called JPEGView for a very good reason: it was written as an image viewer, not as an image conversion package. Its ability to convert between JFIF and QuickTime PICT format is simply there as a convenient means of adding/removing the QuickTime PICT header; no additional compression or decompression of the image is involved. Its features in the Save dialog—saving cropped PICTs and saving reduced color sets—are all there for the purpose of improving the on-screen viewing of JPEG images. And its ability to add previews and icons to images is implemented to allow you to see in advance what you will be viewing on your screen. One of the primary reasons for adding GIF support was to lessen the need to convert all GIF files to JPEGs. All of this adds up to sharpen the focus of JPEGView as an application; I think you'll find that its special features, low memory requirements, and overall stability are a direct result of keeping its feature set under control.

Why are my windows filled with strange patterns of dots?

Because JPEGView has marked that window for slow updating, meaning that the image is going to need to be decompressed again before it can be displayed. See the section on "Fast and Slow Window Updates" in the *Viewing Images* chapter for a precise description of what's going on.

Why don't some of my JPEGView documents have icons?

If you upgraded from a previous version of JPEGView, which did not handle some of the more recent types of image files, then the Finder doesn't realize that the new version supports them. To solve this, simply rebuild your desktop by holding down the Command () and Option keys during startup.

Why does JPEGView run out of memory during a slide show?

When JPEGView runs a slide show, it needs to have enough memory to hold two compressed images in memory simultaneously, in addition to any memory needed for color reduction and for the JPEGView code itself. The reason for this is that while the current slide is displayed, JPEGView will prepare the next image for display, in order to help minimize the delay between slides. Thus, to ensure that you have enough memory to run your slide show, make sure you give JPEGView enough memory to hold the two largest compressed images in memory, plus about 500k of overhead.

Why are the progress percentages so erratic with PICT images?

This is one of QuickTime's "features": you tell it what subroutine to call for progress updates, and it's supposed to call it regularly. Unfortunately, this doesn't always happen as regularly as I'd like, but there's nothing I can do to fix it. C'est la vie. QuickTime 1.0 is particularly notorious for this problem; QuickTime 1.5 and later shows a marked improvement, but things could still be better.

What is banding?

Banding is a technique used by Apple's QuickTime to allow you to compress large JPEG images in a small amount of memory. Essentially, it involves dividing the image vertically into several strips, or "bands," and compressing them separately. Unfortunately, this also makes them rather difficult to reassemble, except in the context of the QuickTime PICT format. For this reason, banded images cannot be converted to JFIF files.

Where can I get JPEG pictures?

On the Internet, you can now find JPEG pictures via anonymous FTP at the Washington University archives (wuarchive.wustl.edu). Simply log in as "anonymous", give your email address as a password, and begin poking around. Additionally, those with Usenet access should check out the alt.binaries.pictures newsgroups, which are becoming more and more dominated by full-color JPEG images every day.

Where can I get GIF pictures?

If you're on the Internet, you're in luck: check out the GIF archives of wuarchive.wustl.edu via anonymous FTP (give your email address as the password). There are also quite a lot of BBS systems that carry GIF files. Look around and you're bound to find some!

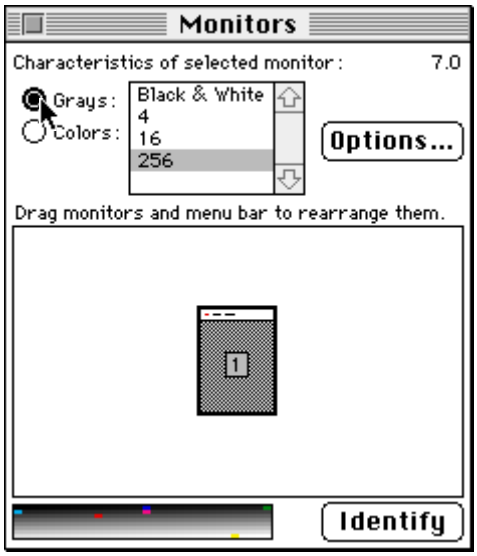
Hints and Tips

Here are some useful techniques you can use to get the most out of JPEGView.

You can sacrifice quality and system response time for speed.

There are several techniques you can use to make JPEGView open and display images more quickly; however, as with all things, there are tradeoffs involved. If you don't mind sacrificing a little bit of picture quality, you can gain a great deal of speed. First, if you're working on a color system, set the default display quality to *Normal* (under the "Drawing" preferences); although JPEGView's new dithering routines give superior quality, they are somewhat slower than the built-in routines that Apple provides. Returning to normal quality will make scaling and dithering of images go faster.

If you are on a 256-color system, you can also gain a lot of speed by turning off two-pass color reduction (choose *Never* under the "Automatically reduce images to 16 or 256 colors" options in the "Drawing" preferences). Your images will look much grainier, but they only need to be decompressed once to be displayed. Alternatively, if you don't mind working in grayscales, you can open the Monitors controls panel, and select "Grays":



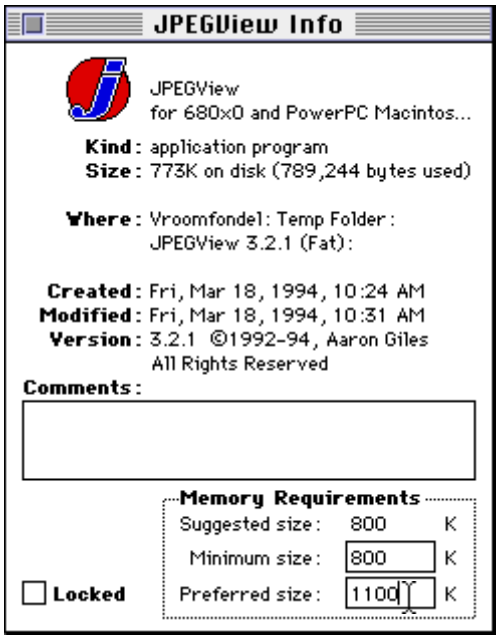
Selecting "Grays" in the Monitors control panel means that you never have to use two-pass color reduction.

JPEGView never does two-pass color reduction when displaying images on a grayscale device, and you get excellent quality with 256 grays.

Finally, if you don't mind slowing other applications down to a halt, you can set the "Time given to other applications" option to "Minimal" (under the Miscellany preferences). This causes JPEGView to hog more of the processor time for drawing, making it work faster overall.

If you have memory, use it!

Increase JPEGView's memory as much as you can afford to (using the **Get Info** box from the Finder). JPEGView comes initially set to use only 1024K of memory; although this is adequate for most basic operations, there is a great advantage to be gained by increasing the available memory.



*Using the **Get Info** command from within the Finder, change the "Preferred Size" to give JPEGView more memory.*

When JPEGView has enough free memory, it will store an off-screen copy of the uncompressed image displayed on your screen, which will be used to redraw obscured parts of the window instantly. If there is not enough free memory for this extra copy, JPEGView will have to decompress the image again before it can update an obscured window.

In addition, if you use two-pass color reduction and give JPEGView enough memory for a 24-bit image the size of your screen, the image will only need to be decompressed once. To give you an idea of how much memory is needed for optimal performance, here is a table of minimum recommended memory sizes for various monitor setups. To determine how much memory you should give to JPEGView, take the value in the table appropriate for your monitor and add to that the compressed size of the largest image you would like to view.

For 256 colors with no color reduction:

- 12" monitors: 500K
- 13" monitors: 600K
- 16" / full-page monitors: 850K

- 21”/two-page monitors: 1300K

For 256 colors with color reduction, or for Millions of colors:

- 12” monitors: 1100K
- 13” monitors: 1600K
- 16”/full-page monitors: 2500K
- 21”/two-page monitors: 4300K

Note again that these memory values are not hard minimums; rather, they are the minimum memory needed to show optimum performance on single images. The default allocation of 1024K is adequate for viewing almost any image less than 500K compressed, albeit with some speed degradation.

Also, note that if you have virtual memory, you should never give JPEGView more memory than the amount of *real, physical* RAM you have in your system. Otherwise, your hard drive is going to go nuts while JPEGView tries to access more memory than you have!

Make good use of the Crop & Zoom feature in the View menu.

There is a lot of power hidden in this innocuously titled feature! In fact, the primary reason for it being there, believe it or not, is that it is a replacement for scroll bars. With most image viewing utilities on the Macintosh, you are initially shown only a portion of the image, which you then must scroll around in, looking for whatever details you’re interested in. If you’re lucky, maybe it will let you automatically scale the image to fit on your screen.

JPEGView takes exactly the opposite approach: you are given the full image first, scaled to fit on your screen, with the option of zooming in on any part of that image. To do this, you simply click and drag a selection rectangle on your image, and then select **Crop & Zoom** from the **Edit** menu. Additionally, when used in conjunction with the **Select Screen Area** option in the **Edit** menu, you can automatically select a screen-sized chunk of the image to view at full resolution by simply dragging the selection rectangle anywhere in your image. This is fully illustrated in the *Special Effects* chapter.

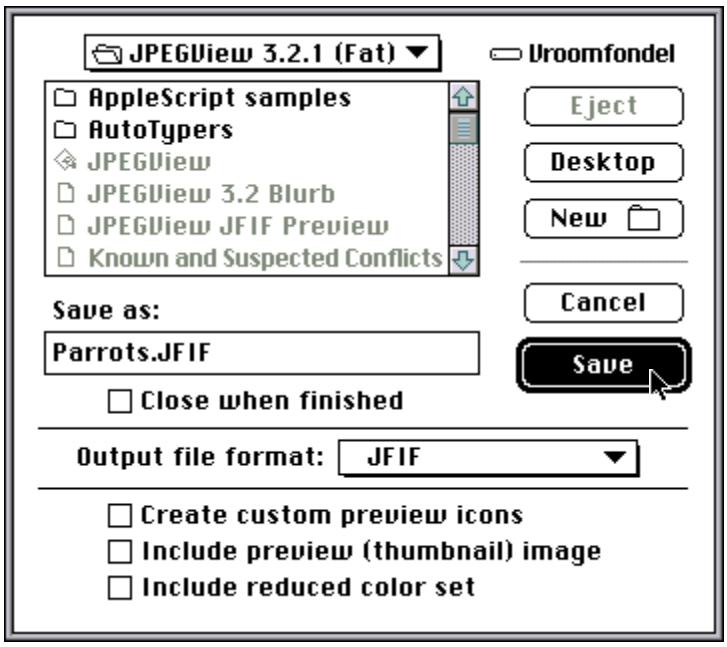
This approach has three important advantages: first, you gain back all the screen area taken up by those annoying scroll bars in the window; second, you are not required to have enough memory to buffer the entire decompressed image at full resolution; and finally, rather than scrolling around helplessly wondering what you are looking for, you can see with your own eyes the full image and move the selection rectangle over the interesting portions. It may not be the method you’re used to, but I believe it is far more powerful than the standard scroll bars.

Create a folder of aliases to use for your slide show.

This technique allows you to choose exactly which images you want to display in your slide show without having to reshuffle your image collection. Simply make aliases of all the images you want to have in your slide show, drag them all into a separate folder, and run the slide show from that folder. JPEGView will automatically resolve the aliases and display the images as you would expect.

Make sure you convert JPEGs to JFIF format before posting and trading them!

When swapping JPEG pictures with others—and especially when uploading JPEGs to public access sites!—always make sure you save the image as a JFIF file; otherwise, only Macintosh users will be able to read the file and you're liable to make some others upset! You should also save JFIF images *without* custom icons and reduced color information—this extra information tends to confuse some file transfer utilities!



Increasing your chances for success. When exchanging images with others on other types of computers, turn off custom previews and preview images, and save the image as a JFIF file.

Be careful if you are converting your GIFs to JPEGs!

A heated and never-ending debate has been filling the Usenet over whether converting GIF files to JPEGs is a good idea. Those opposed to the idea claim that JPEG was not designed to work with 256 (or fewer) color pictures and that JPEG only makes bad pictures look worse. Proponents feel that the space savings gained more than makes up for any quality degradation, which becomes minimal for high-quality GIFs. So, who's right and who's wrong? Unfortunately, it depends on the GIF image. For line drawings, cartoons, and GIFs with very few colors (16 or less), conversion to JPEG is usually a bad idea, as GIF compresses such images very well and JPEG will likely create some unwanted "noise" in the image. For high-quality, 256-color GIF images, JPEG compression does pretty well and introduces little visible loss in image quality. For other images, a judgement can only be made on an image-by-image basis. The conversion issue is by no means clear-cut.

Contacting the Author

JPEGView is postcardware, meaning that if you find yourself using JPEGView regularly, I am asking you to send me a postcard. So don't be shy, just scribble your name, address, an email address if you have one, and any comments you wish (or none, if you prefer) on a postcard and slip it in your local mailbox addressed to me at:

Aaron Giles
P.O. Box 2967
San Rafael, CA 94912
U.S.A.

What do you get for your postcard? Well, if you give me an email address that I can access through the Internet, I plan on providing updates and information through a mailing list I've set up. You will likely receive information about new versions before everyone else. Also, postcards provide encouragement; if I know many people are using JPEGView and find it useful, I have far more incentive to improve it in the future. So… if you'd like to see an even more powerful image viewer in the future, a postcard would certainly help!

Any user of JPEGView 3.0 or later also has the option of becoming a fully registered user. Registered users who send in a one-time registration fee of US$20 receive a printed, bound copy of the full JPEGView documentation, along with an official JPEGView 3.3 release disk, containing the complete JPEGView 3.3 release plus a few of my favorite JPEG images. My current U.S. Mail address can be found in the chapter *Contacting the Author*, later on in this documentation.

Overseas users who wish to avoid the hassle and fees of international money orders are asked to simply send the equivalent of US$20 in cash, plus about US$3 extra for postage (everything is sent airmail). Although this is not recommended by the post office, you can usually get away with it as long as you do well to mask the fact that there is cash enclosed, i.e., make sure it is wrapped in sufficiently opaque paper.

Note that registration is *optional*; the only "payment" that is required for individual use of JPEGView is a postcard.

Finally, a very simple site license is available for universities and companies that wish to make JPEGView 3.3 an officially supported product—and it typically only costs you the low low price of a few individual registrations (US$50 to be exact)! Contact me via email (agiles@aol.com) or U.S. Mail (the address above) for full details, or just go ahead and send in the cash (most people seem to do it anyway!) Purchase orders are welcome.

Organizations and individuals who are interested in distributing JPEGView with their products should contact me first before doing so. I'm usually pretty flexible about licensing, especially for "good causes", so if you're interested, please don't hesitate to get in touch with me about it!

I would also love to hear any comments, suggestions, bug reports, etc., from anyone using JPEGView. To get them to me, you can either include them with your postcard (preferred), or, if you've already sent your postcard, you can send them to me via email on the Internet at:

agiles@aol.com

or through email on America On-Line, using the address:

AGiles

If you send me bug reports that appear to be QuickTime-related, I will forward them to the QuickTime engineers at Apple.

Acknowledgments

JPEGView was—for the most part—written, conceived, produced, and documented by Aaron Giles. Behind the scenes, however, were quite a number of other rather important people you should know about.

Dirty Work

First I want to thank all those other programmers who were nice enough to do some of the dirty work for me. Their efforts in particular are responsible for making JPEGView work so smoothly:

Troy Gaul wrote *Infinity Windoid*, the code that draws the way cool floating windows everyone likes so much.

Ramon Felciano wrote *Mercutio*, the code which draws the menus and handles lots of weird keypresses.

The **Independent JPEG Group**, headed by **Tom Lane**, wrote the original color reduction algorithm, which was used as the basis for JPEGView's color reduction, in addition to the JPEG decompression code which is used as an alternative to QuickTime.

Ross Williams wrote LZRW1, the data compression technique used to squeeze the on-line help down to a reasonable size.

Prerelease Hell

A big part of putting together a new release of JPEGView is the serious amount of testing that is needed to make sure everything works on all (most?) machines. To this end, I would like to acknowledge all the alpha and beta testers who have contributed toward testing this JPEGView release:

Bob Andrews, Jim Brunner, Justin Collins, Shawn Connelly, John Coolidge, Ramon Felciano, Mike Fetzer, Troy Gaul, Stephen Haase, Paul Jacoby, Paul Jensen, Dave Johnson, Hugh Johnson, David Johnston, Mark Krueger, Tom Lane, Richard Lim, Tim Lowe, Robert Mah, Tom McDougal, Sam Neal, Tim North, Victor Norton, Brian Olson, Dan Pendergrass, Steve Poole, Charlie Reiman, Axel M. Roest, Eric Sack, Francois Schiettecatte, Bart Sears, James N. Stricherz, Steve Unger, Chris Webster.

Friends in High Places

A few special words of thanks are in order for several Apple employees who have made all the difference in helping me make JPEGView a success.

Mark Krueger was the guy who originally noticed JPEGView and helped me change it from a really basic hack into a robust QuickTime application.

Tim Lowe has been instrumental in helping me adopt and adapt to new Apple technologies—especially AppleScript—faster than I would have ever been able to without him.

And finally, thanks to **Jim Pelkey** and **Linda Haas**, for organizing the PowerPC developer's kitchen, and for making it such a great success.

Others Who Have Made a Difference

Tom Lane wrote the second-largest help chapter, *General JPEG Questions & Answers*, and deserves some additional praise for founding and heading the Independent JPEG Group, whose free source code has made the widespread use of JPEG images possible.

John Coolidge, I recently discovered, is responsible for sorting and maintaining the burgeoning JPEG image archives at wuarchive.wustl.edu. Here's hoping he can find the time (or the help!) to finally get all those pictures sorted!

More than 950 registered JPEGView users have so far taken the time to send in their postcards and letters, and believe me, the response has kept me going — keep 'em coming!

Finally, I want to thank my wonderful wife **Shanti**, for being (relatively) understanding and even supportive of my programming addiction / obsession.

Program History

Version 3.3 (May, 1994)

- Added support for dragging selected portions of images to other applications.
- Added drag-and-drop support for comments and statistics text.
- Added the ability to drag folders from the Finder into the Slide Show Options dialog.
- Progress displays are now updated more frequently.
- Added support for the Independent JPEG Group's code as an option.
- Added proportional and non-dog-eared icon styles.
- Changed the scale AppleScript property to accept integers to set scaling percentage.
- Removed the file format popup menu when there is no choice of formats available.
- Added the ability to use the current selection for cropping icons and / or images.
- Rewrote high quality drawing to be much faster; made very high the new default.
- Restored the ability to save cropped images.
- Rewrote the marching ants algorithm to be more visible onscreen.
- Opening PICTs at something other than 72 dpi now expands them to full size.
- Removed the unnecessary use of the Time Manager for timing image drawing.
- Fixed the problem where small image files would be incorrectly reported as invalid.
- Unresolvable aliases in a slide show folder no longer abort the scan in that folder.
- Recompiled the JPEGView JFIF Preview into a true fat component.
- Color startup screens are now displayed properly if a B&W version is present.
- Revealing part of a window during two-pass color reduction no longer aborts it.
- You are no longer asked to insert a disk if the last slide show was on a removable drive.
- The internal patch for QuickTime with virtual memory has been improved immensely.
- Scaling large images down to icon size should no longer leave occasional black bands.

Version 3.2 (March, 1994)

- Added support for TIFF baseline and LZW-compressed images.
- Added support for BMP images, compressed and uncompressed.
- Added support for MacPaint and Startup Screen images.
- Added rudimentary printing support.
- Added copying to the clipboard as a (memory-intensive) option.
- The last slide in a single-pass user-controlled slide show is now displayed.
- Cleaned up code for reversing within the slide show to work better.
- Re-enabled high quality drawing; JPEGView 3.1 always drew in very high quality.
- Colors are now restored when hiding the slide show.
- Hiding the slide show no longer crashes the machine.
- Better system checking for clean exits on 68000-class machines.
- Changed compiler options to work around Gatekeeper Aid conflict.
- Loosened up GIF decoding slightly to work with GIFs from some broken encoders.
- Fixed crash when reducing 16-bit color pictures.
- Real aliases are now stored for the slide show folder.
- Incorporated the QuickTime VM extension directly into JPEGView.
- Added the ability to save any image in its own format with previews and icons.
- Menubar area now gets updated instantly when toggling the menubar.
- Lots of internal changes and cleanup for the PowerPC.
- Improved floating window handling.

Version 3.1 (December, 1993)

- Added the ability to work in the background when saving and drawing images.
- Redesigned the preferences and slide show options dialogs, making them modeless.
- Added a progress dialog and the option to cancel while scanning for slide show images.
- Added the slide show controls floating window, allowing point-and-click control.
- Added the ability to back up in the slide show, and the new “pause and hide” feature.
- Recoded the drawing functions to fix incorrect displays on mixed color / grayscale systems.
- Fixed a bug that would crash the slide show under EvenBetterBusError.
- Fixed the bug that prevented the “Repeat last slide show” preference from working.
- Coded around the conflict with Connectix PowerBook Utilities.
- Sped up saving when creating both previews and icons for an image.
- Added the ability to save and recognize color reduction for JFIF images.
- Added support for multiple-image / animated GIF files and for GIF89 comments.
- Added better recognition of variant PICT formats.
- Added automatic width control of the comments window.

Version 3.0 (September, 1993)

- Full AppleEvent support, providing scripting and recording with AppleScript.
- Improved custom dithering, with user-selectable quality settings.
- Statistics window now floats above others and has a zoom box.
- Added support for reading and displaying comments from JPEG and PICT images.
- Slide shows can now render the next image offscreen before display.
- Displaying filenames is supported in slide shows.
- An indicator of slide show progress has been added.
- Added extended balloon help for nearly everything.
- Interrupting the slide show now gives you options to continue, end, or change options.
- Custom icons can now be created when saving images.
- Time is now given to background applications during decompression.
- Selections can be made and tracked in full-screen windows.
- Select screen area works more precisely.
- User-resizing is possible now by dragging the lower-right corner of an image.
- Added the ability to customize the use of offscreen bitmaps.
- Added a sparse pattern of dots to indicate pending slow updates.
- Added decompression time to the Statistics floating window.
- A simple new Colors floating window has been added.
- Moved cropping functions to the Edit menu.
- Fixed the bug that would occasionally cause Out Of Memory errors.
- Fixed the problems with offscreen bitmaps containing black areas.
- Added custom spinning cursors for long operations.
- Made the restoration of colors after closing the last image an option.
- Removed the distinction between JFIF and JPEG file typing.
- Moved all strings out of the code and into resources.
- Single-character keyboard shortcuts have been added for alerts and dialogs.
- Dialogs and alerts now appear centered on the monitor which currently has the mouse pointer.
- The preferences and slide show dialogs are now movable.
- Error messages and alerts have become a little more user-friendly.
- The progress dialogs now tell you which image is being worked on.
- The help text is now stored in a compressed format.
- Measures are explicitly taken to prevent screen savers from taking over during a slide show.
- JFIF previews are now supported via the JPEGView JFIF Preview extension.
- JFIF previews can now be compressed.
- “Automatically Fix File Types” option now works better under A/UX.

- Temporary memory is now used in some situations.
- The Help window can now be sized and zoomed.
- Dithering is now always on by default, unless you uncheck the preferences item “dither reduced color images”.
- Color accuracy problems when copying from offscreen buffers are gone.
- You’re not allowed to start the slide show until you’ve selected a valid folder.
- Changed the About box.

Version 2.0 (November, 1992)

- Extensively cleaned up and reorganized the Version 1.1 code.
- Added the Colors menu, allowing color palette selection.
- Added a dithering option that works.
- Added an IJG-based color reduction scheme for 8 and 4 bit systems.
- Improved the workings of offscreen bitmaps.
- Fixed full screen windows to include the corners of all monitors.
- Reorganized and expanded the Statistics window.
- Added identification of IJG-produced JFIFs, with estimates for the quality.
- Added identification of banding in PICT images.
- Added fast GIF decoding and viewing capabilities.
- Fixed problems with the progress percentage; added scaling indicator.
- Added options to automatically scan and fix file types within JPEGView.
- Added support for non-JFIF JPEG images, and identification of Adobe JPEGs.
- Reworked and reorganized the preferences options.
- Added support for many new preferences relating to the new features.
- Broke the on-line help into several sections and added a modeless interface.
- Improved slide show to handle color reduction during pauses between slides.
- Fixed much of the flakiness in handling large uncompressed PICT images.
- Added the ability to load and save custom palettes in PICT images.
- Removed support for compression of PICTs.
- Added select screen area option.
- Added the ability to drag the active selection rectangle.
- Modified offscreen bitmap handling to do long updates only after 5 seconds idle time.
- Fixed dialog handling to allow quick updates behind dialog boxes.
- Added the Import option.
- Added the ability to detect QuickTime-created JFIFs.
- Provided stronger support for PICTs at resolutions other than 72 dpi.

Version 1.1 (July, 1992)

- Added support for the automatic Slide Show.
- Added Next and Previous items to the Windows menu.
- Added options for specifying startup actions.
- Added the ability to select PICT as the default output file format.
- Added a warning option when the offscreen bitmap cannot be created.
- Added support for opening any image in a full screen window automatically.
- Changed the way files are saved to prevent data loss when an error occurs.
- Fixed a bug that caused the wrong image to sometimes be displayed in full screen.
- Fixed the problem that led to occasional crashes on startup.
- Put to rest the remaining problems with Open-Wide.

Version 1.0 (June, 1992)

- Rewrote I/O routines to handle file identification and conversions more cleanly.
- Added the ability to open any type of PICT.
- Added rudimentary JPEG compression options for uncompressed PICTs.
- Added support for opening and saving compressed PICT0 resources.
- Added rudimentary support for specifying JPEG compression quality in Preferences.
- Added the ability to produce thumbnail sketches in JFIF files.
- Added full screen support, and options for automatically using it.
- Added explicit support for multiple monitors.
- Added View menu, and options for cropping, uncropping, zooming, and rescaling.
- Added Statistics window and options for controlling it.
- Added creation of offscreen bitmaps for faster window updates.
- Made the Help dialog bigger and better, and redesigned the About box.
- Added Balloon Help.
- Fixed a conflict with the Open-Wide control panel.
- Improved memory management.

Version 0.9 (March, 1992)

- Initial release.