Appendix D:
Frequently Asked Questions

What exactly is an alpha channel?

First of all, a *channel* is a color or data component of a pixel. It often refers collectively to a particular color component for all pixels in an image. A screen grab on a color NeXT has three channels: red, green and blue. An *alpha* channel or coverage component indicates how transparent each pixel of data is. For example, the **alphaMage.tiff** picture shipped with Pixel Magician has a totally transparent background (alpha = 0.0), a totally opaque central figure (alpha = 1.0) and a partially transparent shadow (0.0 < alpha < 1.0). The trans- parency can be observed by Sovering the image with different background colors. The transparency data can be used to achieve interesting effects when combining images with one another. By itself, an image with transparency looks no different from a normal opaque image. The only way to ªseeº the transparency is to change the background behind the picture. Just think of the TV ad comparing a real aquarium with a TV picture of an aquarium. It is impossible to tell the difference (especially on a low-resolution device like a TV) until the pitchman walks behind the aquarium.

I select Retain Alpha Channel when I save my image but the final result still has no alpha channel.

You are probably Sovering your image onto the background when you want to Copy it instead.

Sovering an image onto a background is like painting the image onto a canvas the color of the background. The canvas can be seen wherever the image is not totally opaque. If you use a solid background, however, the final resultÐthe painted image *with* the canvas, is no longer transparent. When you save the Sovered image, you are saving the painted canvas. If the canvas is opaque, then the final result (the painted canvas) is also opaque, and the saved image will not have an alpha channel. Of course, you could choose to paint onto a totally transparent canvas (like a piece of clear glass). The final result would then be exactly as transparent as your image. Theoretically, this is just like Copy.

Copying an image preserves the alpha channel *as is*. Nothing is done with it when the image is saved. Of course, to view the image on the screen, some- thing needs to be done with the alpha channel. On color NeXTs, copied images are Sovered onto black backgrounds (so the background of the **alphaMage.tiff** picture shipped with Pixel Magician appears black), but on monochrome NeXTs and when printed on the NeXT printer, they are Sovered onto white backgrounds (the background of **alphaMage.tiff** will appear white). This is for purposes of display only. When the image is saved, only the image data itself is saved (only the paint and not the canvas using the Sover analogy). When using Copy, the user-selected background color will only be seen around the edges of a rotated image.

Now why is Copy *theoretically* the same as Sovering onto a transparent background? The results of Copy and Sover are computed using different mathematical formulae. Copy uses the very simple *final image = original image* formula and is faster than Sover, which requires two additions and a multi- plication to be performed for each pixel in the image. For normal pictures, these two formulae will yield the same result, but for certain non-compliant, illegal pictures, the two methods may give different results. In this case, one picture may be more useful than the other.

What does not checking Retain Alpha Channel do? When this switch is not checked, the result after applying the Copy or Sover or other composite operator is examined and, if an alpha channel remains, it will be manually removed from the final result. Theoretically, because Pixel Magician treats the color data as premultiplied, Copying a picture and not checking Retain Alpha Channel is equivalent to Sovering the image onto a black background. Again, with certain non-compliant pictures, different results may be obtained using the two methods. The Retain Alpha Channel switch can also be useful as a shortcut to producing opaque results when compositing with operators such as Dover and PlusL.

When I make an image with transparency paletted, I can't change the background color. Why is this?

Since Pixel Magician changes the bit depth of a file before performing any compositing operations, making a file paletted and compositing it with the background color probably cannot be accomplished in one step with the results you expect. When the file is paletted, the alpha channel is lostÐ

Pixel Magician does not fully support paletted files with transparency data. Most compositing operators then have no effect.

To work around this, break the task into smaller steps. First, composite the picture with the background color of your choice and save the file. Make sure the resulting picture has no alpha channel. If necessary, you can force Pixel Magician to strip the alpha channel when you save the file. Then convert this intermediate file to the paletted bit depth of your choice. Since the intermediate file is unimportant, using a format like TIFF or PXM is best. These formats preserve the most information about your file and are the formats read most quickly by Pixel Magician.

For example, suppose you want to send the supplied **alphaMage.tiff** picture to a friend with a PC who can only read 8 bit PCX files. Since PCX does not support an alpha channel, you will have to do something with it to remove it. One possibility is to place the image over a background color of your choice using the Sover operator. Of course, you can use other composite operators for other effects. Save or convert this file to an intermediate file, *e.g.,* **/tmp/alphaMage.pxm.** Now convert **/tmp/alphaMage.pxm** to an 8 bit PCX file either by using the Convert window or by opening **/tmp/alphaMage.pxm**, changing it to 8 bit paletted with the Image Inspector and saving it as a PCX file.

This same tactic can also be used with other complicated operations. If Pixel Magician is not giving you the results you expect, break the task into little subtasks and use intermediate files to force the operations to occur in exactly the order desired.

Why do my rotated pictures appear skewed?

Uncheck the Proportional switch in the Scale panel. For more details on why skewing occurs, see the section on Rotating in Appendix B, ªNotes on Scaling, Rotating and Dots Per Inch.º

If I convert a TIFF into a GIF and later convert it back to a TIFF, why does the final result look worse than the original?

Pixel Magician can read and display TIFFs with up to 8 bits each of red, green and blue data for each pixel in the image (assuming you have a NeXTdimension). Thus, TIFF images can be displayed with up 24 bits of visible color data (16,777,216 colors). Of course, the monitor does not have enough pixels to show all these colorsÐonly about a million colors can be seen at any one time, but Pixel Magician can be used to pan around an image larger than the physical screen. GIF, however, is capable of storing only 256 colors total for the entire image. When a file with many colors is converted into a GIF file, the choice of which 256 colors to use in the GIF is obviously very important. There is no best way to make this choice, but Pixel Magician gives the user a choice of many popular and powerful palette-selection algorithms.

When the GIF file is converted back into a TIFF, no matter how good the color selection algorithm was, some colors will have been lost so the new TIFF will not look as good as the original. The lost colors are nowhere in the GIF file and can not miraculously reappear when the GIF is converted back into a TIFF.

This problem can also occur with color NeXTstations, although it should not look as severe, and with other 24/32 bit formats such as TGA, PICT, PCX and Sun Raster.

Exactly the same reasoning can be employed to explain why a color file saved to a black and white image (such as the MACPaint format requires) doesn't suddenly acquire the colors of the original color image when the black and white image is converted back to a color format.

What are JPEG and LZW, and how do they differ?

LZW (Lempel-Ziv, Welch) refers to an adaptive coding technique for non-lossy compression. When an image (or other data) is LZW compressed for efficient storage and later uncompressed, the resulting image will be identical to the original. LZW in one form or another is the basis for most high performance non-lossy compression algorithms and is used in the GIF and TIFF file formats and in popular utility programs such as the standard Unix compress program, PKZIP and ARC. LZW works best when the data has long uniform stretches such as occur in line art, faxes, ray-traced images and pictures created using computer drawing and painting programs. For more information, see ªA Technique for High Performance Data Compressionº by Terry A. Welch, *IEEE Computer*, vol. 17 no. 6 (June 1984).

JPEG (Joint Photographic Expert Group) is a lossy compression technique good for continuous tone images. Lossy means that once an image has

been JPEG compressed, it will no longer look as good as the originalÐdata will have been lost. The amount of data lost can be controlled by the Q-Factor. Because data is actually lost, much higher compression ratios can be achieved than is possible through non-lossy algorithms. JPEG is also distinct from most non-lossy compression techniques in that it tries to take advantage of the two-dimensional nature of images. Stated simply, the lossy JPEG algorithm is a DCT (discrete cosine transform) followed by quantization and then Huffman or other encoding of the the resulting coefficients. JPEG works best on scans of moderately complex ªrealº data such as photographs. It is especially good for regions with a steady change in shade. It is not good for line art, faxes or pictures created with painting programs because the DCT will result in Gibbs phenomenon. Complex images also will not encode well if they have strong high frequencies resulting from important image data (as opposed to noise). The best that can be hoped for in this case is that JPEG might preserve the general texture of the high-frequency areas. For more information, see ªOverview of the JPEG Still Picture Compression Algorithmº by G. Wallace, *Electronic Imaging East `90*.

Note 1: JPEG is a truecolor algorithm only. It is not usable with palettes.

Note 2: There is also a lossless JPEG algorithm, but this is not what most people mean when they refer to *the* JPEG algorithm.

NeXTmail Note: Since NeXTmail tars, compresses (with Unix compress) and uuencodes attachments, and since compressing a compressed file with the BSD compress program can at best make it only a little smaller and usually makes it substantially larger, it is not a good idea to send LZW compressed images using NeXTmail. Unless substantial image compression with acceptable loss can be achieved with the JPEG

algorithm, it is probably best not to compress images that are to be sent over NeXTmail.

Why doesn't my software read files I produce with Pixel Magician?

An enormous variety of image software exists on the market today. With most formats, the lack of a standard library means that each program must write, from scratch, all the routines necessary to read and write that format. Many, if not most, image programs are content merely to make sure that they will read files that they themselves write (some software will not even do this)!

Pixel Magician realizes that a program advertising to read a certain format may in fact read only a very limited class of files within that format. For example, an image processing program that claims to accept TGA files may in fact only accept truecolor TGA files since most sophisticated transformations (convolutions, maximal entropy methods, wavelet transforms, shock filters and so forth) are best performed with truecolor data. On the other hand, a PC painting program that claims to read TGA files may in fact only accept paletted files because of memory requirements. The TIFF specification states that probably no program has been written that completely implements the specificationÐthe version 6.0 specification is already well over a hundred pages of fairly dense material, and it is still in progress!

The only solution is for Pixel Magician to give the user the ability to write many different flavors of each file format. If you can't get your software to read Pixel Magician's files, experiment with the different options available. With many programs on systems without virtual memory, the problem may simply be that the image is too large. In this case, see if the program will read a smaller file produced by Pixel Magician. If so, try using Pixel Magician to reduce the number of colors in your image, thereby making the image smaller. If nothing works, send us a copy of the file in question and the name of the program you are trying to use with Pixel Magician. Technical support will be happy to help you with your problem.