# MaskImage: an Exercise in Creating Anti-Aliased Pictures

by

Lawrence D’Oliveiro
Computer Services Dept
University of Waikato
Hamilton, New Zealand
Internet mail: ldo@waikato.ac.nz

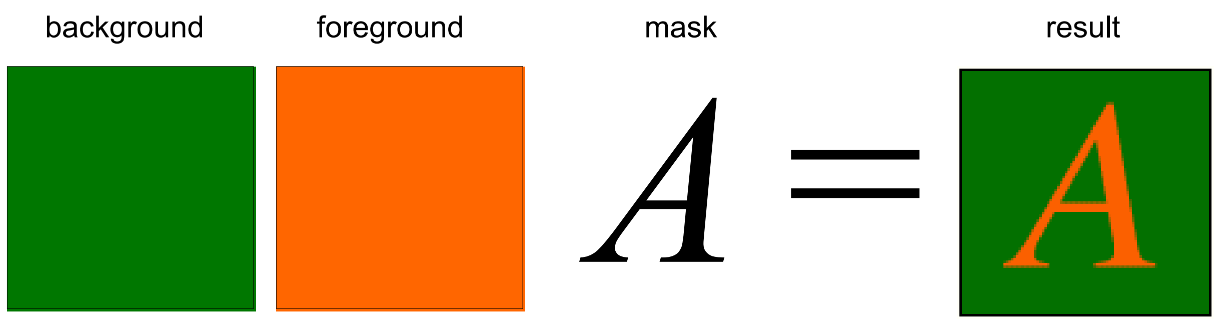
1991 December 26

*MaskImage* is a utility for superimposing one image on top of another through a mask, with nice smooth-looking anti-aliased edges at the mask boundaries. The technique for doing this on the Mac is described in detail in the article “Scoring Points with TrueType”, in issue 7 of Apple’s *develop* magazine.

Basically, *anti-aliasing* is a technique for smoothing the edges of graphical objects by interpolating mixtures of the two adjacent colours all along the boundaries. The result looks very nice on a colour screen, but don’t try printing it out on a black-and-white device (e g a LaserWriter)—it just looks all fuzzy. Also, the anti-aliasing computation is quite time-consuming to do, which is probably why you don’t see it done more often on the Mac. However, you could use it to pretty up logos and other selected images, for on-screen presentation purposes.

To use this utility, you must have a colour-capable Macintosh (LC, Classic II, PowerBook 140 or better). Your system version must be at least 6.0.5 (if your machine doesn’t require a newer version). If your machine is a Mac II, IIx, IIcx or SE/30, you must have the 32-bit QuickDraw extension installed (or be running System 7.0 or later). To actually *view* the images you’re generating in all their glory, you will need suitable display hardware (e g a colour screen and 8-bit-per-pixel or better display hardware), but this is not necessary just to *generate* the images.

To start with, you will need to get three images from somewhere: a foreground, a background, and a mask. The mask controls which part of the foreground image “shows through” the background. Here’s a simple example:



In this case, the background image is a solid green rectangle, the foreground image is a solid orange rectangle, and the mask is the letter “A” in 96-point Times Italic. The foreground and background can be any sort of image you like—if they are not raster images, they will be converted to a raster in the result. The mask should be a black-and-white image that can be scaled up to four times its initial size without introducing any “jaggies”: thus, I don’t recommend a raster image for the mask. For best results, it should consist of resizable geometric objects, or (as in this case) text drawn with scalable outline (TrueType or ATM) fonts.

If you’re using a drawing program to create the images, it will be easiest if you run the program together with HyperCard under MultiFinder. Within the drawing document, position the foreground, background and mask images the way you want them to appear in the composite image. Click on the graphics comprising the foreground image and copy them to the Clipboard. Switch to the MaskImage stack and click the “Paste Foreground Picture” button (*don’t* use HyperCard’s “Paste Picture” function). Go back to your drawing document and select the graphics that make up the foreground image. Copy these to the Clipboard. Back to the MaskImage stack again, and this time click the “Paste Background Picture” button. Back to your drawing document, and select and copy the graphics for the mask. One last time to MaskImage, and click the “Paste Mask Picture” button.

Now you’ve got all the pieces of the image into the stack, click the “Generate Result” button. This part of the operation may take a few seconds (like more than 10), so wait patiently until the button highlight goes off. Once it does, switch back to your drawing program and do a Paste. Voila!

If you didn’t get the position of some component quite right, simply reposition it, recopy it to the Clipboard, click the appropriate Paste button in MaskImage, and regenerate the result. MaskImage keeps all the component images around in memory until you replace them with new ones or you leave the stack, so you don’t have to recopy and paste the ones that don’t change.

And, if you haven’t figured it out already, you didn’t have to paste them into MaskImage initially in the order that I described.

If you don’t have enough memory to run your drawing program and HyperCard at the same time, you can copy the component images one by one from the drawing program into the Scrapbook desk accessory, quit the drawing program, start up HyperCard and open the MaskImage stack, paste the components from the Scrapbook one by one, generate the result, paste it back into the Scrapbook, quit HyperCard, restart your drawing program, and paste the result in from the Scrapbook.

**Notes:**

I did all my testing with Canvas 2.1.1 (not having got my 3.0 upgrade yet). Canvas maintains the relative positions of objects when they've been copied to the Clipboard, so the alignment between foreground, background and mask doesn't get thrown off. However, there may be other drawing programs around that don't do this.

Some drawing programs won't let you paste in graphics that they can't generate themselves. For example, when you paste a colour raster image into MacDraw II, it converts it to a black-and-white one. If your drawing program has a "paste as picture" function, which lets you bring in a picture without trying to interpret it as a collection of editable objects, you could try using it. Otherwise, get another drawing program.

The composite image is generated at a pixel depth of 16 bits per pixel, which I thought was a decent compromise between image size and quality. You can specify another depth (say 32 bits per pixel) by adding another argument in the call to the "MaskImage" XFCN, but I haven't tested this. If your machine's display has 8 bits or less per pixel, you may notice that the composite image draws relatively slowly. This is because it uses QuickDraw's "ditherCopy" mode, which gives a truer approximation to the actual image colours, at the expense of some graininess and (of course) drawing speed.

This stack requires HyperCard 2.0 or later, even though it doesn't actually take advantage of any of the features introduced with HyperCard 2.0.

**Why a HyperCard Stack?**

Let's face it, I'm a hacker. I write lots of little utilities to perform useful functions (useful to me, at least) like this one. I can't always be bothered to take the time to design a nice user interface for all of them. Which is a pity if I want to give them to others.

HyperCard, however, comes with a ready-made set of user-interface tools. And it comes with this truly wonderful ability to let you add new functions to it, in the form of XCMDs and XFCNs. Thus I can spend my time writing the core part of the code without worrying about the user interface, and then design the interface as a separate step, without worrying about how that impacts on the core part of the code.

*And* if you don't like my interface, you can take apart my stack and build a new one. Or you can just tinker. Feel free to browse, if you're curious. I'm even enclosing the source code for the main part of the MaskImage XFCN, in case you can't get hold of the *develop* article. If you have any comments, you can contact me at the address at the start of this document.

**Late Addition: Ramp Generation**

I stuck this in as an afterthought, and haven't put together a proper interface for invoking it yet. If you want to experiment with this, have a look at the "GenerateRamp" procedure in the card script. What this does is let you generate a graduated fill between any two colours, going either horizontally or vertically. The result could be used as the foreground or background with the MaskImage function, or just pasted into a drawing you're creating, if your drawing program doesn't include a function for generating ramps more conveniently.