The evolution of the ICO file format, part 3: Alpha-blended images

devblogs.microsoft.com/oldnewthing/20101021-00

October 21, 2010

Raymond Chen

Windows XP introduced the ability to provide icon images which contain an 8-bit alpha channel. Up until this point, you had only a 1-bit alpha channel, represented by a *mask*. The representation of an alpha-blended image in your ICO file is pretty straightforward. Recall that the old ICO format supports 0RGB 32bpp bitmaps. To use an alpha-blended image, just drop in a ARGB 32bpp bitmap instead. When the window manager sees a 32bpp bitmap, it looks at the alpha channel. If it's all zeroes, then it assumes that the image is in 0RGB format; otherwise it assumes it is in ARGB format. Everything else remains the same as for the non-alpha version. Note carefully that *everything else remains the same*. In particular, you are still required to provide a mask. I've seen some people be a bit lazy about providing a meaningful mask and just pass in all-zeroes. And everything seems to work just fine, until you hit a case where it doesn't work. (Read on.) There are basically three ways of drawing an alpha-blended icon image.

1. `DrawIcon(DI_NORMAL)` : This is by far the most common way icons are drawn. In the alpha-blended case, this is done by blending the *image* with the destination according to the alpha channel.
2. `DrawIcon(DI_IMAGE)` : This draws the *image* portion of the icon image, completely overwriting the destination.
3. `DrawIcon(DI_MASK)` : This draws only the *mask* portion of the icon image, completely overwriting the destination.

The `DI_IMAGE` and `DI_MASK` flags let an application draw just one of the two images contained in an icon image. Applications do this if they want finer control over the icon-drawing process. For example, they might ask for the mask so they can build a shadow effect under the icon. The mask tells them which parts of the icon are opaque and therefore should cast a shadow. If you understand this, then you can see how people who set their *mask* image to all-zeroes managed to get away with it most of the time. Since most programs just use `DI_NORMAL` to draw icons, the incorrect mask is never used, so the error never shows up. It's

only when the icon is used by a program that wants to do fancy icon effects and asks for `DI_MASK` (or calls `GetIconInfo` and looks at the `hbmMask`) that the incorrect mask results in an ugly icon.

The ironic thing is that the people who incorrectly set the mask to all-zeroes are probably the same people who will then turn around and say, “When I try to use alpha-blended icons, the result is hideously ugly under conditions X and Y. Those Microsoft programmers are such idiots. More proof that Windows is a buggy pile of manure.” What they don’t realize is that the hideous ugliness was caused by their own error.

Raymond Chen

Follow