Why are icons multiples of 8 pixels in width?

devblogs.microsoft.com/oldnewthing/20050823-12

August 23, 2005

Raymond Chen

Icons are all multiples of eight pixels in width. It's not just because computer people like powers of two.

Back in the early days of Windows, video cards were monochrome or, if you were lucky, 16-color. These were all planar video modes, the mechanics of which were discussed earlier. Now imagine copying a bitmap to the screen where both the bitmap and the screen are planar. If the starting coordinates of the destination was an exact multiple of eight, then the bitmap could be copied via block transfer instructions. On the other hand, if the destination was not a perfect multiple of eight, you had to do a lot of fancy bit shifting to get it onto the screen.

This is the source of the `CS_BYTEALIGNCLIENT` window class style. With this style set, the window manager will try to position the window so that the x-coordinate of the client rectangle's upper left corner sits at a perfect byte boundary of video memory. If you were running at a 1bpp video mode (monochrome or 16 color), this meant that the x-coordinate was a multiple of eight. By positioning the window this way, a bitmap copied to the upper left corner of the client rectangle would be copied via fast block transfer instructions.

If you look at dialog box dimensions from Windows 95 or earlier, you'll find that they are nearly always a multiple of 32 DLUs in width. Since four horizontal DLUs equal one average character width, you had to keep your dialog width a multiple of 32 to ensure that the final dialog size was a multiple of eight.

Keeping bitmap widths such that they represented exact byte boundaries was important for performance on the machines of the day. Copying blocks of pixels around was typically performed in three major steps: A thin vertical strip from the left edge of the bitmap to the first byte boundary, then the bulk of the bitmap up to the last byte boundary, and finally a thin vertical strip from the last byte boundary to the right edge. If you kept your eyes open, you could actually see these three stages of drawing occurring. (Like I said, machines of the day weren't all that fast.) Keeping things byte aligned and at byte width meant that the two thin vertical strips had zero width and therefore could be optimized out.

Of course, in today's world of 32bpp displays, all these old considerations are largely irrelevant.

Raymond Chen

Follow