What do bitwise operations mean for colors?

devblogs.microsoft.com/oldnewthing/20061113-11

November 13, 2006

Raymond Chen

Someday, you're going to pass a raster operation to the `BitBlt` function that entails bit manipulation. Something like `SRCAND` perhaps, or possibly the dreaded `SRCINVERT`. These bitwise operations make perfect sense for monochrome bitmaps, since those are one bit per pixel anyway. But what does it mean for color bitmaps? What do you get when you "and" together forest green (#228B22) and hot pink (#FF69B4)? The bitwise operations are performed in the pixel space of the destination. If the destination is a non-palettized bitmap format (higher than 8bpp), then the pixel values are red, blue and green components packed together (in various ways depending on the format). Those values are "and"ed together as integers to produce the result. For example, in a 565-format bitmap, the color forest green is represented as the value 0x2464 (0y00100`100011`00100) and hot pink is 0xFB56 (0y11111`011010`10110). The bitwise "and" of these two values is 0x2044, which is a very dark purple. With palettized bitmaps, the results are much less predictable since the values in the bitmap are not colors but color indices. For example, if a pixel has the value 6, that means that the color of the pixel is determined by the entry in slot 6 of the bitmap's color table, and that color could be anything. There is no rule that even requires that color 0 be black and that the highest available color be white, though most bitmaps adhere to this by convention. On an 8bpp bitmap, then, the question of what you get when you "and" together forest green and hot pink is underdetermined. If the color table for example happened to put forest green in slot 6 and hot pink in slot 18, the result of the "and" operation would be 6 & 18 = 2, and the result pixel would therefore be whatever color was in slot 2 of the bitmap's color table. What does this mean for you, adventuresome blitter?

If you're going to use raster operations that involve bitwise operations, one of the pixels involved in the operation should be black or white (zero or all-bits-set) in order to obtain predictable results. You can then use identities like "x and black = black" and "x xor black = x" to predict the result, assuming the bitmap follows the convention for black and white noted above. But if you're going to be xor'ing with forest green, then the results could be anything.

Raymond Chen

Follow