Some notes on Win32 carets

devblogs.microsoft.com/oldnewthing/20240916-00

September 16, 2024

In Win32, the *caret* is a visual indication of where the next typed character will be inserted. It is traditionally a thin vertical bar.

Note that all defaults listed below are as of this writing. The values are not contractual, but I'm just providing them for illustration.

You can ask for the caret blink rate by calling the `GetCaretBlinkTime` function (default: 500ms). Note that the caret will stop blinking after a while (default: 5 seconds) to avoid running timers indefinitely if they have a period shorter than one minute. Furthermore, on Remote Desktop connections, caret blinking might be disable entirely. This avoids long-running timers on systems that may be hosting multiple simultaneous interactive users. When you have 100 users signed in, each with a blinking caret, even a timer that fires only once every 500ms is running at a high enough frequency to cause measurable energy consumption.

The caret blink rate also controls the default blink rate used by the `FlashWindowEx` function, and when the system flashes the title bar of a window when you click on its disabled owner, it does so (today) at 8 times the caret blink rate.

When you create a caret, you can let the system choose the dimensions by setting the width and height to zero. Today, this uses the system-defined window border width and height. Unfortunately, the window manager lets you down here: The recommended width is not the window border width but the caret width, which comes from `SystemParametersInfo` with `SPI_GETCARETWIDTH`. So those defaults don't help you much; they're the wrong values.

Another way to create a caret is to use a bitmap. In that case, it is your responsibility to keep the bitmap handle valid for as long as the bitmap caret is in use, *viz*. until the caret is destroyed, or it is replaced by another caret.

In all cases, the system draws the caret by the classic algorithm of XOR. This works okay if the background is white or black (or close to it). Not so great if the background is some other color, and definitely not so great if the background is gray (because inverted gray is still gray).

In the case where you set the caret to a bitmap, the system will still use XOR to draw it. If your bitmap is not a monochrome bitmap, then things will probably look kind of weird because XOR'ing two colors together tends to produce non-intuitive results.

I guess what you could do is XOR the background color with your desired caret color, and use that color as the color in your caret bitmap. Through the magic of XOR, when the caret bitmap is XOR'd with the background color, your desired caret color pops out.

background ^ (background ^ foreground) = foreground

This is really more of a hack based on a mathematical quirk than an intended design behavior. Bitmap carets were intended to be used with monochrome bitmaps.

There you go, a bunch of loosely-connected paragraphs on Win32 carets.