If vertical strips are better, why do toolbars use horizontal strips?

devblogs.microsoft.com/oldnewthing/20070129-02

January 29, 2007

Raymond Chen

If vertical strips are better, why do toolbars use horizontal strips? An early version of the toolbar control first made its appearance in Windows 3.0, and in those days, screen resolutions were low and toolbar buttons were small. Horizontal or vertical didn't really matter. Ten bitmaps, each 16 × 16, at 4-bit color, comes out to one kilobyte, much less than even a single 4KB page. (And what's this paging nonsense anyway? We're talking 16-bit Windows here!) When this code was ported to 32-bit Windows for Windows 95, the existing bitmap format was preserved in order to retain compatibility with the 16-bit clients of the toolbar control. Windows 95 included both the 16-bit and 32-bit versions of the common controls library, and they both built out of the same sources, with a little `#ifdef WIN32` action here and there to deal with the places where 16/32 source code compatibility didn't quite cover everything. The early prototypes of the Windows 95 shell were written in this mixed 16/32 model, so that the same program could be compiled either as a 16-bit program or a 32-bit program. The 16-bit version was used in the earlier builds because the 32-bit kernel, GDI, and USER were all being developed in parallel. The 16-bit version went out in the builds, and the 32-bit version was used by the development team as a proving ground for Windows 95's 32-bit computing environment. When the 32-bit environment was declared "good enough to self-host", the switch was thrown, and the 32-bit shell went into the builds instead of the 16-bit shell. All this is a long way of saying that source code compatibility between 16-bit code and 32-bit code was very important, not only within the Windows 95 shell team, but throughout the Windows 95 product, so that teams could port their 16-bit components to 32-bit in a gradual and evolutionary manner. If you were to introduce gratuitous subtle differences (like the orientation of bitmap strips in toolbars), you wouldn't make many friends. They would port their program to 32-bit Windows and the toolbars would all come out funny. "But it works fine when I compile it as 16-bit. Must be a bug in Windows 95. I'll wait for the next build and try again." With the introduction of version 4.70 of the common controls, the internal structure of the toolbars changed, and the bitmaps came to be stored as imagelists rather than as raw bitmaps. Imagelists store their images in a vertical strip, so once again all is well with the world. "Why not add a `TB_ADDVERTICALBITMAP` message?" Well, for one thing, you're looking for inefficiency at the wrong level. All that happens with the bitmap is that it gets stuffed into an imagelist.

Nobody actually draws out of it, so there's no point in optimize it for drawing. And for another thing, you're looking for inefficiency at the wrong level. If this even shows up on your performance traces, it means that you're spending your time loading images into toolbars. Isn't that a bigger cause for concern, that you're spending so much of your time setting up toolbars? Shouldn't you be spending your time doing productive work?

(Imagelists? Use your imagination.)

Raymond Chen

Follow