What does CS_SAVEBITS do?

devblogs.microsoft.com/oldnewthing/20060428-00

April 28, 2006

Raymond Chen

If you specify the `CS_SAVEBITS` class style, then the window manager will try to save the bits covered by the window. But the real question is why, because that is your guide to using this power only for good, not for evil. When a window whose class specifies the `CS_SAVEBITS` class style is displayed, the window manager takes a snapshot of the pixels on the screen where the window will be displayed. First, it asks the video card to store the pixels in available off-screen video memory (fast). If no video memory is available, then the pixels will be stored in system memory (slower). If the saved pixels have not been discarded in the meantime (see below), then when the window is hidden, the saved pixels are copied back to the screen and validated; in other words, the pixels are marked as "good" and no `WM_PAINT` message is generated. What invalidates the saved pixels? Anything that would cause those pixels to be out of sync with what should be on the screen once the popup window is removed. Here are some examples:

- If the popup window moves, then the saved pixels are discarded, since putting those pixels back on the screen would put them in the wrong place.
- If an underlying window invalidates itself (most commonly via `InvalidateRect`), then the saved pixels are also discarded, because the underlying window has indicated that it wants to change its pixels.
- If any windows beneath the popup change size or position or z-order, then the saved pixels are of no use.
- If any windows are created or destroyed beneath the popup.
- If somebody calls `GetDC` for a window beneath the popup and starts drawing.

You get the idea. If copying the saved pixels back to the screen would result in an inconsistent display, then the saved pixels are discarded. So how do you use this power for good and not for evil? One consideration is that the region should cover a relatively small portion of the screen, because the larger the saved bitmap, the less likely it will fit into available off-screen video memory, which means the more likely it will have to travel across the bus in a video-to-system-memory blit, the dreaded "vid-sys blt" that game developers are well familiar with. In the grand scheme of vid/sys blts, "vid-vid" is the fastest (since the video card is very good at shuffling memory around within itself), "sys-sys" is next best (since the motherboard can shuffle memory around within itself, though it'll cost you CPU cache space), "sys-vid" is in

third place, and “vid-sys” is the worst: Programs write to video memory much more often than they read from it. As a result, the bandwidth between the video card and system memory is optimized for writing to video, not reading from it. But the primary concern for deciding when to use the `CS_SAVEBITS` window class style is not making the window manager go to all the trouble of saving the pixels, only to have to throw them away. A window that is a good candidate for the `CS_SAVEBITS` style is therefore one that does not move, covers a relatively small portion of the screen, and is visible for only a short time. That the window shouldn’t move is obvious: If the window moves, then the saved pixels are useless. The other two rules of thumb try to minimize the opportunity for another window to do something that invalidates the saved pixels. By keeping the window small in area and putting it on the screen for only a short time, you keep the “target” small both spatially and temporally. Consequently, the best candidates for `CS_SAVEBITS` are menus, tooltips, and small dialogs, since they aren’t too big, they don’t typically move around, and they go away pretty quickly.

(Some people appear to be under the mistaken impression that `CS_SAVEBITS` saves the bits of the window itself. I don’t know where people get this impression from since even a modicum of experimentation easily demonstrates it to be false. The Windows drawing model follows the principle of Don’t save anything you can recalculate.)

Raymond Chen

Follow