Why does a maximized window have the wrong window rectangle?

devblogs.microsoft.com/oldnewthing/20120326-00

March 26, 2012

Raymond Chen

Commenter configurator wonders why the maximum size for a form is the screen size plus (12,12). Somebody else wonders why it's the screen size plus (16,16). Let's start by rewinding the clock to Windows 3.0. When you maximize a window, the default position of the window is such that all the resizing borders hang "off the edges of the screen". The client area extends from the left edge of the screen to the right edge of the screen, and also goes all the way to the bottom. It doesn't go all the way to the top, since it needs to leave room for the caption, but the resizing border that sits above the caption area is not visible either. The reason for this should be obvious: Since the window is maximized, there's no point wasting screen real estate on the resizing borders. You want the client area to be as large as possible; that's why you maximized the window. The result of this window positioning is that the window rectangle itself is slightly larger than the screen. The parts that "hang off the edges of the screen" are not visible because, well, they're off the screen. (Of course, if your window had a maximum size smaller than the screen, then those borders stay visible.) The size of these borders might not be 12 pixels, mind you. This is how things stood for a long time. Even the introduction of multiple monitors in Windows 98 didn't affect the way maximized windows were positioned. Multiple monitors, however, altered one of the assumptions that lay behind the positioning of maximized windows, namely the assumption that edges beyond the screen were not visible. I mean, they weren't visible on the screen that held the maximized window, but they *were* visible on the adjacent monitor. As a result, when you maximized a window, its borders appeared as a sliver on the adjacent monitor. Why didn't Windows get rid of the sliver when multiple monitors were introduced? You probably know the reason already: Because there are applications which relied on the sliver. For example, an application might detect that it is maximized by checking whether its edges hang off the screen, rather than checking the `WS_MAXIMIZED` style. Why would they do it that way? Probably because they fumbled around until they found something that seemed to work, sort of like the people who detect whether the mouse buttons are swapped by calling `SwapMouseButton` instead of `GetSystemMetrics(SM_SWAPBUTTON)`. (Or maybe because they wanted to treat as "logically maximized" windows which the user had manually resized to be larger than the screen.) The introduction of the Desktop Window Manager in Windows Vista gave the window manager team a chance to solve the problem without impairing compatibility: The Desktop Window

Manager controls how windows appear on the screen, which can be different from the actual window properties. For example, the Desktop Window Manager typically animates a window into position when it becomes visible, yet if an application calls `GetWindowRect`, it will just see the window at its normal position with no animation.

This decoupling of logical and physical characteristics permits all sorts of visual tricks. The visual trick relevant here is the removal of the overhang borders from a maximized window. The borders are still there: If you call `GetWindowRect`, you will get the same coordinates you always did. But they don't appear on the screen. The sliver is gone.

Raymond Chen

Follow