How does the window manager decide where to place a newly-created window?

devblogs.microsoft.com/oldnewthing/20121126-00

November 26, 2012

Raymond Chen

Amit wonders how Windows chooses where to place a newly-opened window on a multiple-monitor system and gives as an example an application whose monitor choice appears inconsistent. The easy part is if the application specifies where it wants the window to be. In that case, the window is placed at the requested location. How the application chooses those coordinates is up to the application. On the other hand, if the application passes `CW_USEDEFAULT`, this means that the application is saying, "I have no opinion where the window should go. Please pick a place for me." If this is the first top-level window created by the application with `CW_USEDEFAULT` as its position, and the `STARTF_USEPOSITION` flag is set in the `STARTUPINFO`, then use the position provided in the `dwX` and `dwY` members. Officially, that's all you're going to see in the documentation. Past this point is all implementation detail. I'm providing it here to satisfy your curiosity, but please don't write code that relies on it. (This is, I realize, a meaningless request, but I must go through the motions of making it anyway.) Okay, now let's dive into the various levels of automatic window positioning the window manager performs. Remember, these algorithms are not contractual and can change at any time. (In fact, they *have* changed in the past.) Just to make it harder to rely on this algorithm, I will not tell you which operating system implements the algorithm described below. From now on, assume that the application has specified `CW_USEDEFAULT` as its position. Also assume that the window is a top-level window. First we have to choose a monitor.

- If the window was created with an owner, then the window goes onto the monitor associated with the owner window. This tends to keep related windows together on the same monitor.
- Else, if the process was created by the `ShellExecuteEx` function, and the `SEE_MASK_HMONITOR` flag was passed in the `SHELLEXECUTEINFO` structure, then the window goes onto the specified monitor.
- Else, the window goes on the primary monitor.

Next, we have to choose a location on that monitor.

- If this is the first time we need to choose a default location on a monitor, or if the previous default location is too close to the bottom right corner of the monitor, then act as if the previous default location for the monitor was the upper left corner of the monitor.
- The next default location on a monitor is offset from the previous default location, diagonally down and to the right.
    - The vertical offset is chosen so that the top edge of the new window lines up against the bottom of the previous window's caption.
    - The horizontal offset is chosen so that the left edge of the new window lines up against the right edge of the caption icon of the previous window.

The effect of this algorithm is that if you open a bunch of default-positioned windows on a monitor, they line up in a pretty cascade marching down and to the right, until the cascade goes too far, and then they return to the upper left and resume cascading. Finally, after choosing a monitor and a location on the monitor, the selected location is adjusted (if possible) so that the window does not span monitors.

And that's it, the default-window-positioning algorithm, as it existed in an unspecified version of Windows. Remember, this algorithm has been tweaked in the past, and it will get tweaked more in the future, so don't rely on it.

Raymond Chen

Follow