TrackMouseEvent tracks mouse events in your window, but only if the events belong to your window

devblogs.microsoft.com/oldnewthing/20101206-00

December 6, 2010

Raymond Chen

Greg Williams wonders why `TrackMouseEvent` fails to detect mouse hover events when responding to DoDragDrop callbacks. "My suspicion is that `DoDragDrop` monopolizes the window so that a `WM_MOUSEHOVER` message is never posted, so it won't end up being useful." That's basically it, for the appropriate sense of the word "monopolize." The `TrackMouseEvent` monitors mouse events that take place in your window and notifies your window when events of interest occur. But this requires that the events actually take place in your window! The `DoDragDrop` function calls `SetCapture` so that it can carry out the task of following the mouse anywhere on the screen. Recall that mouse events normally are delivered to the window beneath the mouse, but `SetCapture` lets you say, "No, I want all mouse events to go to me for as long as the mouse button is held down. Mine! All mine!" This function is typically called when you are performing some sort of mouse drag operation so that the window can respond to mouse events even after the use has dragged the mouse out of the window. (Which, in many cases, is the only interesting case.) That's why `TrackMouseEvent` has no effect when you try to use it to detect mouse hovering during a drag/drop operation: The `TrackMouseEvent` function is not seeing any mouse events! They're all being stolen by `DoDragDrop`. The unfortunate consequence of this is that if you want to have a special behavior during drag/drop hover, you'll have to detect the hover manually by remembering the mouse position and timestamp and waiting for the appropriate amount of time to elapse without a significant mouse motion.

Exercise: But wait, since I don't get drag/drop events when the mouse is not inside my window, how can I simulate `WM_MOUSELEAVE`?

Raymond Chen

Follow