Logical consequences of the way Windows converts single-clicks into double-clicks

devblogs.microsoft.com/oldnewthing/20041015-00

October 15, 2004

Raymond Chen

First, I'm going to refer you to the MSDN documentation on mouse clicks, since that's the starting point. I'm going to assume that you know the mechanics of how single-clicks are converted to double-clicks.

Okay, now that you've read it, let's talk about some logical consequences of that article and what it means for the way you design your user interface.

First, some people design their double-click action to be something unrelated to the single-click action. They want to know if they can suppress the initial WM_LBUTTONDOWN of the double-click sequence.

Of course, you realize that that would require clairevoyance. When the mouse button goes down for the first time, the window manager doesn't know whether another click will come or not. (Heck, often the user doesn't know either!) So it spits out a WM_LBUTTONDOWN and waits for more.

Now suppose you're a program that nevertheless wants to continue with the dubious design of having the double-click action be unrelated to the single-click action. What do you do?

Well, one thing you could do is to do nothing on receipt of the WM_LBUTTONDOWN message aside from set a timer to fire in `GetDoubleClickTime()` milliseconds. [Corrected 10am.] If you get a WM_LBUTTONDBLCLK message within that time, then it was a double-click after all. If you don't, then it must have been a single-click, so you can do your single-click action (although a bit late).

This "wait a tick" technique is also necessary if you don't have a double-click action, but the second click causes trouble in conjunction with the first click. Why is this necessary? Because many users double-click everything. Here are some examples of where the "delayed action to avoid the second click" can be seen:

- The context menu that appears for taskbar notification icons. If the context menu appeared immediately upon the first click, then the second click would dismiss the context menu, leaving the user confused. "I clicked and something happened and then it went away." (Users don't say "I double-clicked"; they just say that they clicked. Double-click is the only thing they know how to do, so they just call it "click". For the same reason you don't say "I drove my blue car" if you have only one car.)
- If Explorer is in one-click mode, it waits to see if there is a second click, and if so, it ignores it. Otherwise, when people double-click, they launch two copies of the program. Furthermore, if you suppress the second click but don't wait a tick, then the program they launched gets stuck **behind** the Explorer window, since the user clicked on Explorer after launching the program.
- The XP style Start button ignores the second click. Otherwise, when people double-click the Start button, the first click would open the Start menu and the second click would dismiss it! (This is sometimes known as "debouncing".)

Let's demonstrate how you might implement click delay. Start with the scratch program and add the following:

```
void CALLBACK DelayedSingleClick(HWND hwnd, UINT,
                                 UINT_PTR id, DWORD)
{
    KillTimer(hwnd, id);
    MessageBeep(MB_OK);
}


void OnLButtonDown(HWND hwnd, BOOL fDoubleClick,
                   int x, int y, UINT keyFlags)
{
    if (fDoubleClick) {
        KillTimer(hwnd, 1);
        MessageBeep(MB_ICONASTERISK);
    } else {
        SetTimer(hwnd, 1, GetDoubleClickTime(),
                 DelayedSingleClick);
    }
}


    HANDLE_MSG(hwnd, WM_LBUTTONDOWN, OnLButtonDown);
    HANDLE_MSG(hwnd, WM_LBUTTONDBLCLK, OnLButtonDown);
```

Also, since we're messing with double clicks, we should turn them on:

```
    wc.style = CS_DBLCLKS;
```

When you run this program, click and double-click in the client area. Notice that the program doesn't react to the single click until after your double-click timeout has elapsed, because it's waiting to see if you are going to continue to click a second time (and therefore double-click instead of single-click).

Next time, we'll look at clicks beyond two.

Raymond Chen

Follow