Why is there no support in the window manager for mouse button chording?

devblogs.microsoft.com/oldnewthing/20090413-00

April 13, 2009

Raymond Chen

Commenter Nekto2 asks why there is no mouse action associated with "click both buttons at the same time".

The window manager doesn't fire a special event for *both mouse buttons held down simultaneously* like it does for double-clicks. As with higher-order clicks, mouse chording is something that you have to put together yourself from the basic mouse events that the window manager generates. Add these lines to our scratch program:

```
void OnButtonDown(HWND hwnd, BOOL fDoubleClick,
                  int x, int y, UINT keyFlags)
{
  if ((keyFlags & (MK_LBUTTON | MK_RBUTTON)) ==
                      (MK_LBUTTON | MK_RBUTTON))
  {
    MessageBeep(IDOK);
  }
}


// Add to WndProc
    HANDLE_MSG(hwnd, WM_LBUTTONDOWN, OnButtonDown);
    HANDLE_MSG(hwnd, WM_RBUTTONDOWN, OnButtonDown);
```

When you run this program, it beeps when both the left and right mouse buttons are pressed.

Note that the program does not require the two button presses take place simultaneously. Most people do not have the dexterity to push the two buttons at precisely the same instant in time. (Especially since Einstein taught us that simultaneity is relative anyway.)

Why don't more programs use chording?

Recall that the semantics of double-clicking should be an extension of the single-click so that your program can perform the single-click action immediately (providing positive feedback to the user that the click was recognized), and then continue to the double-click action if a second click comes in. For example, a common pattern is for the single-click to select the clicked-on item and the double-click to launch it. You can stop at the first click and the result still makes sense. For chords, you have to have *two* stopping points, one if the user left-clicks and stops, and another if the user right-clicks and stops. Coming up with a chord action that is compatible with both stopping points requires more effort.

Another reason why many people choose to avoid chords in their user interface design is that chording requires more dexterity, and many users simply don't have the fine motor control necessary to pull it off without accidentally invoking some other action (such as a drag). Chording is also cumbersome to emulate with MouseKeys, so you run afoul of accessibility issues.

Still, there's nothing technically preventing you from using chording in your program. If you want to code it up, then more power to you.

Raymond Chen

Follow