Why do I get a spurious WM_MOUSEMOVE message whenever Resource Manager is running?

devblogs.microsoft.com/oldnewthing/20160616-00

June 16, 2016

Raymond Chen

We saw some time ago that the operating system generates spurious `WM_MOUSEMOVE` messages as a side effect of mouse cursor recalculations. In a sense, the way to kick-start a mouse cursor recalculation is to pretend that the mouse moved. And if you see a continuous stream of spurious `WM_MOUSEMOVE` messages, then it suggests that there is some continuous activity that is triggering mouse cursor recalculation.

A customer discovered that their program receive spurious `WM_MOUSEMOVE` messages at a rate of approximately one every two seconds. They were able to narrow down the conditions and found that it occurred when the Hyper-V Manager or Resource Monitor were running. What is it about those two programs that is causing all the spurious mouse activity?

The common thread is that these program have a continuously updating window that refreshes (you guessed it) every two seconds. At each refresh, the programs do a `WM_SETREDRAW` to disable redrawing in their window, then they update all their stuff, and then they do another `WM_SETREDRAW` to re-enable redrawing.

We saw some time ago that the window manager provides a default implementation of the `WM_SETREDRAW` message which makes the window pseudo-invisible when redraw is disabled, and then undoes all the weird fakery when redraw is re-enabled.

Undoing the weird fakery is where the spurious `WM_MOUSEMOVE` messages are coming from. Making a window pseudo-visible means, among other things, that if the cursor is inside the window that just became pseudo-visible, the window needs to be told, “Hey, you have the mouse cursor. Set the cursor to something appropriate.”

And that’s what’s triggering the spurious mouse move messages every two seconds.

If processing mouse-move messages is expensive for your program, you can watch to see if you receive a mouse move message that matches the coordinates of the previous message. If so, and your program state hasn’t changed between the two messages, then ignore the

spurious message. (Of course, if your program state *has* changed, then you will want to reprocess the message, because the answer to the question might be different the second time around.)

Raymond Chen

Follow