How can I trigger a recalc of the mouse cursor after I changed some of my internal application state?, follow-up

devblogs.microsoft.com/oldnewthing/20230525-00

May 25, 2023

Raymond Chen

Some time ago, I described how to trigger a recalc of the mouse cursor after changing internal application state. In the bonus chatter, I noted that the cursor you set with `SetCursor` is used only when the mouse is over a window belonging to the thread that called it. Commenter Stuart Ballard asked,

> Does the Bonus Chatter imply that in theory there's no need for the
>
> ```
> window == child || IsChild(window, child)
> ```
>
> check at all? If I understand rightly, all we're doing is sending a message saying "the mouse cursor moved". Is there any reason it'd be bad for an unrelated window to receive a message like that?

The danger here is that the window you find may belong to a window from another thread. If you removed the "is for me or my children" check, then you are going to ask that other window to do a cursor recalculation, and that has a few problems.

First, it reopens the race condition I discussed in the bonus chatter: If the mouse has moved since the time you called `GetCursorPos`, then you are going to ask that other window to recalculate based on stale data. And this time, the problem is real: That other window may have already processed the system-generated `WM_SETCURSOR` message, and then you follow up with a `WM_SETCURSOR` message with the wrong coordinates, leaving the cursor in an incorrect state.

There's also a race condition if the target window reconfigures itself between the `WM_HITTEST` message and the `WM_SETCURSOR` message, and you end up asking the window to set its cursor based on outdated information.

This was not a problem if the message was for a window belonging to the same thread, because you know that the window cannot reconfigure itself between the `WM_HITTEST` message and the `WM_SETCURSOR` message, nor can it process a newer mouse-move event:

Those actions would have to happen on the same thread, and that thread is busy running *your* code.

Another danger of removing the extra test is that the target window's thread may be hung. and trying to send a message to that window will cause your thread to hang, too.

So keep filtering the window before you ask it to recalculate the mouse cursor.

Bonus chatter: Does this mean that you can simplify the test to

```
GetWindowThreadProcessId(child, nullptr) == GetCurrentThreadId()
```

?

No.

Cross-thread window hierarchies are valid, and if there is a child window that belongs to another thread, you stil want to tell it to recalculate the cursor. There is no race condition here, however, because cross-thread window hierarchies result in in input queue synchronization, so the other thread won't receive cursor messages until the current thread has finished.