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

devblogs.microsoft.com/oldnewthing/20220921-00

September 21, 2022

Raymond Chen

Suppose your program changes state in a way that affects the mouse cursor. For example, maybe it changed from "busy" to "not busy", or it moved some windowless controls around. How can you get the system to recalculate the mouse cursor based on the new program state?

One thing not to do is just call `SetCursor` with the new cursor, because the mouse may not even be over a part of your window that uses the new cursor. Or it may not even be over your window at all.

Another thing not to do is to try to change the cursor by calling `SetClassLongPtr` with `GCL_HCURSOR`. This is using a global solution to a local problem, because changing the class cursor changes *all* windows of that class, not just the one you want to update. Besides, changing the class cursor doesn't result in the cursor changing. It just changes the default cursor the system uses the *next time* a cursor calculation occurs.

What you want to do is re-run the cursor calculation. So let's do that.

```
void RecalcWindowCursor(HWND window)
{
  POINT pt;
  if (GetCursorPos(&pt)) {
    HWND child = WindowFromPoint(pt);
    if (window == child || IsChild(window, child)) {
      UINT code = SendMessage(child, WM_NCHITTEST, 0, MAKELPARAM(pt.x, pt.y));
      SendMessage(child, WM_SETCURSOR, (WPARAM)child, MAKELPARAM(code,
WM_MOUSEMOVE));
    }
  }
}
```

We get the cursor position and see if the cursor is over our window (or one of our children). If so, then we need to recalculate the cursor, so we send the `WM_NCHITTEST` message to obtain the hit-test code for the cursor position. Then we generate a `WM_SETCURSOR` message for that hit-test code and send it to the child window, pretending that the mouse has

just moved. This kicks off the standard process of setting the mouse cursor: The child window forwards to its parent to see if the parent wants to override the cursor. If not, then the child window picks its own cursor, which defaults to the class cursor.

Bonus chatter: But what if the mouse moves after we called `GetCursorPos`? Well, if the mouse moved to some other part of the window, or to another window on the same thread, then the window that the mouse has moved to will get a new mouse-move message, which will trigger a recalculation of the mouse cursor at its new position. And if the mouse moved to a window from some other thread, then the cursor calculations we performed will have no visible effect: The cursor established by `SetCursor` applies only when the mouse is over a window belonging to the thread that called it.

Raymond Chen

Follow