The effect of SetCursor lasts only until the next SetCursor

devblogs.microsoft.com/oldnewthing/20050525-27

May 25, 2005

Raymond Chen

Of course the effect of the `SetCursor` function for a thread lasts only until that thread changes the cursor to something else. Any moron knows that, right?

The tricky part is that the `SetCursor` may come from an unexpected place.

THe most common place people run into this is when they do something like this:

```
// Put up the hourglass
HCURSOR hcurPrev = SetCursor(hcurWait);
... do some processing ...
// Restore the original cursor
SetCursor(hcurPrev);
```

This puts up the hourglass during the processing. But if you pump messages (or if a function you call pumps messages), then the hourglass will go away and return to the normal arrow.

That's because when you pump messages, this opens the gates for messages like `WM_NCHITTEST` and `WM_SETCURSOR`. The latter in particular will typically result in the cursor changing, either to a cursor selected by the window itself or to the class cursor if the message makes it all the way to `DefWindowProc`.

If you want to keep the hourglass up even while pumping messages, you need to let the window know that "If you are asked to set the cursor, please put up an hourglass instead of what you would normally display as the cursor." That window would then have to alter its `WM_SETCURSOR` handling to take this setting into account.

```
case WM_SETCURSOR:
 if (ForceHourglass()) {
   SetCursor(hcurWait);
   return TRUE;
 }
 ...
```

Note that forcing the hourglass is only the tip of the iceberg. Even though the cursor is an hourglass, the window is still active and can receive other message, such as mouse clicks and keypresses. If your program is not ready to receive new input during this phase, you need to detect this case and not go into some recursive state if the user, say, impatiently clicks the “Compute!” button while you are still computing.

Raymond Chen

Follow