What was the ShowCursor function intended to be used for?

17 Dec 2009 7:00 AM 17

Back in the days when Windows was introduced, a mouse was a fancy newfangled gadget which not everybody had on their machine. Windows acknowledged this and supported systems without a mouse by having keyboard accelerators for everything (or at least that was the intent). But if the design stopped there, you'd have a dead cursor in the middle of your screen all the time, which you could move around if you had a mouse, which you didn't.

Enter the `ShowCursor` function.

The `ShowCursor` function takes a parameter that indicates whether you want to show or hide the cursor. (It would perhaps be more verbosely named `ChangeCursorShowState`.) If you call `ShowCursor(TRUE)` then the cursor show count is incremented by one; if you call `ShowCursor(FALSE)` then the cursor show count is decremented by one. A cursor is show on the screen if the cursor show count is greater than or equal to zero.

When Windows starts up, it checks if you have a mouse. If so, then the cursor show count is initialized to zero; otherwise, it is initialized to negative one. That way, you don't get an annoying immovable cursor on the screen if you don't have a mouse.

If a program entered a state where it wanted to show the cursor even on systems without a mouse, it would call `ShowCursor(TRUE)` when it entered the state, and `ShowCursor(FALSE)` when it left it. One such state might be when activating the keyboard interface for selecting a rectangular region in a document. Under these conditions, a program naturally is expected to move the cursor around in response to user actions, even if the user didn't move the physical mouse hardware.

But the most common reason for forcing the cursor to be shown is in order to show an hourglass cursor because it's busy. That's right, back in the mouseless days, code to display an hourglass cursor went like this:

```
HCURSOR hcurPrev = SetCursor(LoadCursor(NULL, IDC_WAIT));
ShowCursor(TRUE); // force cursor shown on mouseless systems
... perform long operation ...
ShowCursor(FALSE); // re-hide cursor on mouseless systems
SetCursor(hcurPrev);
```

Conversely, if a program entered a state where it wanted to hide the cursor even on systems with a mouse, it would call `ShowCursor(FALSE)` when it entered the state, and `ShowCursor(TRUE)` when it left it. For example, you might do this when showing a slide show.

Let's look at how this all worked out in practice. I use a table because people seem to like tables.

	Machine with mouse	**Machine without mouse**
Normal	0 (cursor shown)	-1 (cursor hidden)
Enter mode where cursor should be forced shown		
`ShowCursor(TRUE)`	1 (cursor shown)	0 (cursor shown)
Exit mode where cursor should be forced shown		
`ShowCursor(FALSE)`	0 (cursor shown)	-1 (cursor hidden)
Enter mode where cursor should be forced hidden		

`ShowCursor(FALSE)`	-1 (cursor hidden)	-2 (cursor hidden)
Exit mode where cursor should be forced hidden		
`ShowCursor(TRUE)`	0 (cursor shown)	-1 (cursor hidden)

Now that all systems come with a mouse as standard equipment, this historical information is not of much use, but there it is in case you were wondering. (And in a case of *everything old is new again*, the growing popularity of touch computing means that you once again have a class of computers with no mouse. So maybe this information is useful after all. Just a fluke, I assure you.)

Back in the old 16-bit days, this counter was a global state, along with other window manager states like the focus window and the input queue. During the conversion to Win32, the cursor show counter became a thread-local state. (Naturally, multiple threads could merge their cursor show counters by attachment.) Consequently, when a thread calls `ShowCursor` it affects the cursor show state only for windows that belong to that thread.

Blog - Comment List MSDN TechNet

Comments

Marquess
17 Dec 2009 7:18 AM
#

What if a program exited unexpectedly? Did the system reroll the previous changes to cursor state?

Nick
17 Dec 2009 7:18 AM
#

What happens if you called ShowCursor(FALSE) but then forgot to call ShowCursor(TRUE) and another app called ShowCursor(FALSE)... but DID remember to call ShowCurstor(TRUE)?

Nick
17 Dec 2009 7:23 AM
#

Sorry my bad, I missed the end bit about thread-specific cursor state.

:D

Nathan_works
17 Dec 2009 7:28 AM
#

Playing with our Surface machine here, I've always wondered why a cursor jumps out

every now and then.. (and getting it to go hide again ...)

me
17 Dec 2009 7:52 AM
#

Also remember that with Windows 2.03 on a 8086, outputting on the screen without visible cursor and outputting to the screen with visible cursor were very different from the performance perspective! I know some programs and programmers that used ShowCursor() to increase the speed of the output.

Anonymous
17 Dec 2009 9:25 AM
#

I can remember writing graphics programs under DOS with a mouse (and an installed mouse driver).

You needed to hide the mouse cursor before drawing to the screen, and then show the mouse cursor when you were done drawing. Otherwise, when you moved the mouse around there would be remnants of the mouse left behind where the application was drawing.

Adrian
17 Dec 2009 9:30 AM
#

This is very interesting. I never thought about those uses of the ShowCursor. I've always thought it was primarily for hiding the cursor when it shouldn't be shown, for example, during a slide presentation or video playback.

MarcT
17 Dec 2009 11:58 AM
#

This behavior of course led to the ubiquitous forum thread:

A: "How do I hide the cursor?"

B: "ShowCursor(FALSE)"

A: "I tried that, and it doesn't work"

C: "You have to call it in a loop, like this:

while(ShowCursor(FALSE)) {}"

B: "But [why that's bad]"

A: "Thanx, C, that fixed it!"

B: "Aaaaaugh!"

Nick
17 Dec 2009 12:54 PM
#

"And in a case of everything old is new again, the growing popularity of touch computing means that you once again have a class of computers with no mouse."

Good grief, even more of these dang 80s fashions are making themselves popular again. I'm not really looking forward to the shoulder pads and aviator sunglasses, but at least we'll get miniskirts back.

Oh, and music might be worth listening to again ;)

Anonymous Coward
17 Dec 2009 4:10 PM
#

And Windows XP is still very usable without a mouse. Really, it is. When I am typing long stretches of text (code, reports) I often get by just using keyboard shortcuts. The only thing that really sucks without a mouse is the web.

And of course there we get to yet another reason to have the cursor on: the user might be using mouse keys, for example because they're disabled and can't use the mouse, or like me because I sometimes need to nudge the mouse pointer one pixel and I simply don't have the dexterity.

By the way, I tried to turn off mouse keys and unplugged my mouse, but although Windows acknowledged that there was no mouse on the system anymore (ping-bong) it didn't hide the mouse pointer. It just sat there, unmovable.

> [*Dynamic unplug is a bad test, since Windows doesn't have a time machine. The initial cursor show value was calculated at boot time. -Raymond*]

prb
17 Dec 2009 5:57 PM
#

Whats the reason for using a counter rather than a simple boolean show\hide flag?

> [*Um, work through the scenarios above with a simple boolean and you'll see what goes wrong. (I can't believe I had to write that.) -Raymond*]

Marquess
17 Dec 2009 6:35 PM
#

@prb:

To quote Raymond: “What if two programs did this?” (back when it was a global state)

Worf
17 Dec 2009 10:41 PM
#

Windows CE often has a touch screen, but it really just emulates a mouse. Tapping the screen moves the cursor around.

Which, when you think about it, is kinda handy for giving visual feedback oh where you tapped.

Otherwise you have to provide other kinds of feedback - see Windows Mobile/Phone.

Then again, touch systems often provide a mouse, for the few times that a developer evaded some programming taxes...

Random832
18 Dec 2009 6:34 AM
#

How did the system handle switching between applications back when this was a global state - was this something handled in the default WM_ACTIVATE?

> [*If the system managed the cursor show count on a per-window basis then it wouldn't be a global state any more! The system did nothing about the cursor show count during switching, but if you think about it, that wasn't an issue most of the time anyway. (Hint: Co-operative multi-tasking.) -Raymond*]

Random832
18 Dec 2009 7:53 AM
#

Right, but that requires each application to know and react appropriately when it's being switched, which was why I asked if it was handled in DefWindowProc.

I guess I didn't see that as being 'ruled out' by your statement that the system doesn't manage it, since I didn't see it as part of "the system" if it did.

So each program that messed with the ShowCursor state (other than ones that never yield before reverting the change) had to have its own code to save it and restore it?

> [*Over 99.99% of the time, programs do not yield before reverting the change. (The only example I can think of in the 0.01% percent is Reversi, and possibly not even Reversi.) Given the constraints, it was probably a fair trade-off. -Raymond*]

Ken Hagan
18 Dec 2009 8:16 AM
#

@MarcT: Ow! That's painful to read even at 20 years distance.

Gabe
18 Dec 2009 10:04 AM
#

Random832 has a good point. Programs like screensavers or PowerPoint that need to keep the cursor hidden would have to hide it when they activate and show it again when they deactivate.

I think Raymond is saying, however, that 99.99% of all programs had no need to ever show/hide the cursor except when displaying the hourglass. Since the hourglass only applies while performing a long operation, you would just show the hourglass before starting the operation and hide it afterwards, at which point you would yield control.

> [*PowerPoint isn't even in the 0.01%, because it just does a SetCursor(NULL) in its WM_SETCURSOR handler to avoid having to "remember" to manage the cursor on activation changes. -Raymond*]