What was the ShowCursor function intended to be used for?

devblogs.microsoft.com/oldnewthing/20091217-00

December 17, 2009

Raymond Chen

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.

Raymond Chen

Follow