The wrong way to check whether the mouse buttons have been swapped

devblogs.microsoft.com/oldnewthing/20071017-00

October 17, 2007

Raymond Chen

Back in the late 1990's, the window manager team received a bug that said that sometimes the mouse button state got messed up and the computer acted as if the buttons were stuck down. Further investigation revealed that it occurred only when one particular program was running, and only if the user had enabled mouse button swapping.

The reason is that the program in question detected whether the mouse buttons were swapped with a function like this:

```
// do not use this function
BOOL AreMouseButtonsSwapped()
{
 BOOL fWasSwapped = SwapMouseButton(FALSE);
 if (fWasSwapped) SwapMouseButton(TRUE);
 return fWasSwapped;
}
```

The `SwapMouseButton` function changes the button swap state and returns the old state. The way the program checked whether the buttons were swapped was by unswapping the buttons and using the return value to determine what the previous setting was, then re-swapping the buttons if the previous setting was "Yes, they were swapped."

If you started with the buttons swapped, running this function created a tiny window where the buttons were momentarily unswapped. And if you were unlucky enough to click the mouse during this window of vulnerability, the program saw one mouse button go down and a different button come up! Even though it was the same physical button each time, it was a different logical button, since the meanings of the buttons had changed.

The correct way of detecting whether mouse buttons are swapped is just to ask non-intrusively.

```
BOOL AreMouseButtonsSwapped()
{
 return GetSystemMetrics(SM_SWAPBUTTON);
}
```

Raymond Chen

Follow