When you synthesize input with SendInput, you are also synthesizing the timestamp

devblogs.microsoft.com/oldnewthing/20121101-00

November 1, 2012

Raymond Chen

A customer was reporting a problem when they used the `SendInput` function to simulate a drag/drop operation for automated testing purposes.

> I see the mouse move from one location to another, and the starting and stopping locations are correct on the screen, but the mouse moves instantaneously rather than waiting 500ms between operations. Here's how I'm sending the input.

```
INPUT input[3] = { 0 };
// Click
input[0].type = INPUT_MOUSE;
input[0].mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
input[0].mi.time = 500;
// Drag
input[1].type = INPUT_MOUSE;
input[1].mi.dwFlags = MOUSEEVENTF_MOVE;
input[1].mi.dx = 100;
input[1].mi.dy = 100;
input[1].mi.time = 1000;
// Release
input[2].type = INPUT_MOUSE;
input[2].mi.dwFlags = MOUSEEVENTF_LEFTUP;
input[2].mi.time = 500;
SendInput(3, input, sizeof(INPUT));
```

Well, yeah, all the events occur immediately because you submitted them all at once.

The `time` field in the `MOUSEINPUT` structure is not for introducing delays in playback. Though I'm not sure what the customer thought the `time` field was. They say that they want a 500ms delay between operations. At first, I thought that they may have misinterpreted it as a delay relative to the time the `SendInput` call is made, since they set `input[0].mi.time` to 500 and `input[1].mi.time` to 1000. But if thay were the case, then setting `input[2].mi.time` to 500 would end up going backward in time. But looking at the big picture, it's probably not worth trying to figure out what they were thinking, since that code will have to be scrapped anyway.

The `time` field is for letting an input source (typically a hardware device) say, "Hi, um, the mouse left button went down at 9:30 this morning. Yes, I know it's already 10am. The PCI bus got a flat tire, and then the spare was also flat, and really there's no point going into the details. Sorry this message arrived late." The window manager (and anybody else who bothers to check the `time` member of the `MSG` structure) uses this information to do things like detect double-clicks. If the input source later reports, "Hi, um, the mouse left button went up at 9:30:00.100 this morning, sorry for the late report," the window manager says, "Well, that was only 100 milliseconds after the button went down *thirty minutes ago*, so I guess that's a double-click after all. *Could you try to be a bit more prompt with this information in the future?*" (Sarcasm added.)

In other words, the `time` member of the `MOUSEINPUT` structure is for backdating input events. They still get delivered immediately, but the timestamp allows the window manager (and other code which looks at the timestamp) to make decisions about how they should respond.

Note that post-dating the timestamp does not cause the input delivery to be delayed, and back-dating the timestamp does not cause the input to be inserted into the input stream ahead of other input. The input is merely delivered with a timestamp in the future or in the past. (And who knows what sort of havoc that will create if a program checks the timestamps and notices that they are either from the future or have traveled back in time. Maybe you'll get a call from Microsoft Research asking for more information about your time machine.)

If you want three input events to take place with a 500ms delay between them, then you need to call `SendInput` three times, with a 500ms delay between the calls.

Raymond Chen

Follow