Why is my message queue full of WM_TIMER messages?

devblogs.microsoft.com/oldnewthing/20160624-00

June 24, 2016

Raymond Chen

Dmitry wondered how it's possible for a lot of auto-generated messages to pile up in the message queue. I remarked, "That's a good question, and I didn't provide all the information necessary to answer it. Answering it will take more than two sentences, so I will toss it onto the topic queue."

One of my colleagues wrote to me and said, "Hey, could you bump up the priority of that topic? I happen to have a bug where COM calls are failing because the message queue is full. I wrote some diagnostic code to drain the message queue to see what was in it, and it was full of unprocessed `WM_TIMER` messages. There were 53 timers running at 16ms each, and the UI thread stopped processing messages for 9 seconds."

Back when I explained how asynchronous input worked, I didn't talk about where auto-generated messages came from.

If a request for a message is about to say, "Nope, no matching messages," the window manager makes one last check: "Is there an auto-generated message that could satisfy this request?" If so, then it generates the message, and hey look, there's a message!

The catch is that auto-generated messages are grouped together. For example, if you ask for any kind of mouse message, and there is an auto-generated `WM_MOUSEMOVE` available, then the window manager will generate a `WM_MOUSEMOVE` and then check if that matches the filter you provided. The auto-generation is done this way so that message ordering is preserved within a group. You wouldn't want a mouse-up to be generated before the corresponding mouse-down.

The message groups can be see in functions like `GetQueueStatus` and the `PM_QS_*` flags to `PeekMessage`.

Okay, now we're getting closer to seeing how auto-generated messages can pile up: If you are filtering for a message, and there is an auto-generated message from the same group, but which doesn't match your filter, then the window manager will auto-generate the message, and then go back and re-run the "Find a message" code, which sees the auto-generated message but says, "Nope, I'm not interested in that message."

Another piece of the puzzle is understanding the timer group. There are two messages in the timer group. One is your friend and mine, `WM_TIMER`. The other is an undocumented internal message known as `WM_SYSTIMER`. This is an alternate universe of timers used by the system to manage system things, like the animated concentric circles in the Mouse Sonar feature, deciding when to time out the system tooltips (like the one that appears when you hover over the × button), driving autorepeat when you click on the scroll bar, and blinking the caret in an edit control.

The final piece of the puzzle is the COM modal message loop. This is the message loop used by COM when you call a method on an STA that needs to be marshaled. COM notifies the destination thread that it needs to run some code, and then it enters a modal message loop waiting for the destination thread to reply, “Okay, I’m done. Here’s the answer.”

The COM message loop is a complicated beast, most likely the result of over twenty years of evolution rather than having been designed that way from the beginning. One of the things that it does is peek `WM_SYSTIMER` messages. Another thing that it does is dispatch timer messages, provided you passed the `COWAIT_DISPATCH_WINDOW_MESSAGES` flag.

Okay, here comes the wild ride.

COM wants to process `WM_SYSTIMER` messages, but not `WM_TIMER` messages. It therefore does a `PeekMessage(&msg, nullptr, WM_SYSTIMER, WM_SYSTIMER, PM_REMOVE)`. If there is a `WM_SYSTIMER` message due, then the window manager generates the `WM_SYSTIMER` message on the fly, puts it in the queue, and the `PeekMessage` function returns it. That’s the good case.

Another good case is that there is neither a `WM_SYSTIMER` message nor a `WM_TIMER` message due. In that case, the window manager generates nothing, and the `PeekMessage` function returns “Sorry, I didn’t find anything.”

The bad case is where there is no `WM_SYSTIMER` message due, *but there is a `WM_TIMER` message due*. In that case, the window manager generates the `WM_TIMER` message on the fly and puts it in the queue. But the `PeekMessage` function ignores that message because it’s interested only in `WM_SYSTIMER` messages.

Result: A `WM_TIMER` message got generated and dumped into the queue.

Every time a `WM_TIMER` comes due, another `WM_TIMER` message gets generated and added to the queue. Eventually, your queue fills up with `WM_TIMER` messages.

My colleague replied, “Thanks for the explanation. Of course, it’s COM, the Bermuda triangle of Win32!”

Raymond Chen

Follow