Why are my posted messages getting lost when the user drags my window around?

devblogs.microsoft.com/oldnewthing/20130905-00

September 5, 2013

Raymond Chen

This question was inspired by an actual customer question, but I changed a lot of it around to make for a more interesting story. (Trust me, the original story was *even more boring*.)

A customer's background thread posted a message to the main UI thread to signal something (details not important). They found that the posted message was never received if the user was in the process of dragging the main window around at the time the message was posted. Why would dragging a window cause posted messages to be lost? "We used to post a thread message, but then we saw that *thread messages are eaten by modal loops*, so we switched to posting a message, as you recommended. But that didn't help."

Dragging a window doesn't cause messages to be lost. The modal loop created by the window dragging code calls `DispatchMessage` to deliver the posted message to its target window's window procedure.

"Oh, we don't handle the message in the window procedure. We process it here:

```
BOOL MyApp::PreTranslateMessage(MSG *pmsg)
{
  if (pmsg->message == OUR_SPECIAL_MESSAGE) {
    ... special code here ...
    return TRUE; // handled
  }
  return FALSE; // not handled
}
```

Could that be the problem?"

Yes, that's the problem.

The customer saw the recommendation to use `PostMessage` instead of `PostThreadMessage` but simply blindly followed the advice rather than understanding its rationale so they could apply the advice correctly.

If you read the original recommendation, you'll see that the problem is that when a modal loop runs, your message loop is no longer in control, and therefore any customizations you've made to your message loop will not be in effect. This is normally a good thing. For example, if a dialog box calls `MessageBox`, the dialog keyboard shortcuts shouldn't be active while the message box is displayed. It would be very strange if hitting `Enter` caused the dialog box to invoke its default button *while the modal message box is still on the screen*. The result would most likely be a dialog box without underlying support, which leads to unhappiness.

If there is some sort of message processing you want to happen regardless of which message loop is control, then you can't put it in your custom message loop because (tautologically) your custom message loop is not in control when it is not running. But message loops will call `DispatchMessage`, and that will deliver the message to your window procedure. (Of course, the converse also applies: If you *want* the behavior to be suspended when a modal operation is in progress, you can put it in your message loop.)

Raymond Chen

Follow