Thread messages are eaten by modal loops

devblogs.microsoft.com/oldnewthing/20050426-18

April 26, 2005

Raymond Chen

Thread messages (as generated by the `PostThreadMessage` function) do not go anywhere when passed to the `DispatchMessage` function. This is obvious if you think about it, because there is no window handle associated with a thread message. `DispatchMessage` has no idea what to do with a message with no associated window. It has no choice but to throw the message away.

This has dire consequences for threads which enter modal loops, which any thread with a window almost certainly will do at one time or another. Recall that the traditional modal loop looks like this:

```
while (GetMessage(&msg, NULL, 0, 0)) {
 TranslateMessage(&msg);
 DispatchMessage(&msg);
}
```

If a thread message is returned by the `GetMessage` function, it will just fall through the `TranslateMessage` and `DispatchMessage` without any action being taken. Lost forever.

Thread messages are generally to be avoided on threads that create windows, for this very reason. Of course, if you're going to create a window, why not use `PostMessage` instead, passing that window as the target of the posted message? Since there is now a window handle, the `DispatchMessage` function knows to give the message to your window procedure. Result: Message not lost.

Raymond Chen

Follow