The window manager needs a message pump in order to call you back unexpectedly

devblogs.microsoft.com/oldnewthing/20110304-00

March 4, 2011

Raymond Chen

There are a bunch of different ways of asking the window manager to call you when something interesting happens. Some of them are are in response to things that you explicitly asked for right now. The enumeration functions are classic examples of this. If you call `EnumWindows` and pass a callback function, then that callback is called directly from the enumerator. On the other hand, there is a much larger class of things that are in response either to things that happen on another thread, or in response to things that happen on your thread, but not as a direct result of an immediate request. For example, if you use the `SendMessageCallback` function, and the window manager needs to trigger your callback, the window manager needs a foot in the door of your thread in order to get control. It can't just interrupt code arbitrarily; that way lies madness. So we're looking for some way the window manager can regain control of the CPU at a time when the program is in a stable, re-entrant state. That foot in the door for the window manager is the message pump. That's the one component that the window manager can have some confidence that the program is going to call into periodically. This solves the first problem: How do I get control of the CPU. Furthermore, it's a known quantity for programs that when you call `GetMessage` or `PeekMessage`, incoming sent messages are dispatched, so your program had better be in a stable, re-entrant state when you call those functions. That solves the second problem: How do I get control of the CPU when the program is in a stable state? Take-away: When you register a callback with the window manager, you need to pump messages. Otherwise, the window manager has no way of calling you back.

Related: The alertable wait is the non-GUI analog to pumping messages.

Raymond Chen

Follow