How do I know whether a window message was processed?

devblogs.microsoft.com/oldnewthing/20090518-00

May 18, 2009

Raymond Chen

Commenter Skeets Norquist asks how to tell whether the original window procedure processed a message when you use `CallWindowProc` to call the original window procedure. "CallWindowProc() always seems to return an LRESULT of 0."

No, actually, `CallWindowProc` returns whatever the window procedure did. If the window procedure returned zero, then `CallWindowProc` returns zero. If the window procedure returned 5, then `CallWindowProc` returns 5.

Anyway, back to the original question. You actually know the answer, if you think about it the right way, and there are many right ways of thinking about it.

Technique number 1: Reading the contract from the other side.

How do you know whether the original window procedure handled the message? Well, how does the window manager know that *you* handled the message? When you subclass a window, you are simultaneously implementing both sides of the message-handling contract. The window manager calls your window procedure, which is the recipient side of the contract. You, in turn, call the original window procedure, acting as the sending side of the contract. Therefore, the way you know whether the message was handled by the original window procedure is the same way the window manager knows that you handled the message yourself. For example, if the message is `WM_SETCURSOR`, then the window procedure returns `TRUE` to halt further processing or `FALSE` to continue. This statement applies both to your window procedure as well as the original window procedure, since they're both window procedures!

Technique number 2: Use the golden rule.

This is a very common technique for answering questions like "How do I make «something somebody else has written» «behave in some way»?" If you want to look for a way to somebody else behave in some way, you mentally turn the tables: How would somebody else make me behave in that way?

Here's a specific example: "What message do I send a window to ask if it contains an unsaved document?" Well, let's turn it around: What message would somebody send to you to ask if you contained an unsaved document?

"I don't have a message for that. It's just a boolean flag in my `CDocument` class. There's no message for retrieving it."

Well, if you don't have a message for retrieving the "dirty" flag from your own document, then clearly there is no generic message for it. Because if there were, you'd have implemented it!

Now let's transfer this to the message handling case. You write a window procedure. How do you indicate that you processed the message? Whatever method you use as a window procedure to indicate that you processed the message is the same method you use as the caller of `CallWindowProc` to tell whether the message was handled.

Technique number 3: How do you know whether the base class performed the operation when you override a method in C++? (Or C# or Java or whatever object-oriented language you prefer.)

Subclassing a window is like subclassing a C++ class: Method calls are given to your class first, and you can decide whether to handle it entirely yourself, whether to pass it through to the base class unchanged, or to combine the two (calling the base class and also performing some operation on your own). So let's take the message handling question and turn it into a method override question: If you override a method and then call the base class, how do you know whether the base class implemented the method?

That's right, and that's how you tell whether the original window procedure handled a message.

(The misssing paragraph is, "Well, it depends on the method and what the specification is for how classes should respond to the method.")

Look at the problem three different ways, but it's all the same answer: You tell whether a window procedure processed a message by comparing the actual behavior against the specification.

Now, in practice, window messages are almost all of the form "You must process the message. If you don't want to do anything special, then pass it to `DefWindowProc` and let it do the default processing and return an appropriate value." In those cases, the answer to the question of how to tell whether the message was handled is much easier: The fact that it returned means that the message was handled.

One might even say that the messages like `WM_SETCURSOR` fall into the same category of "The fact that it returned means that the message was handled." Because even if the original window procedure returned `FALSE` to indicate that it wants processing to continue, that is in a sense how it handled the message. It handled the message by saying, "I am handling this message by telling you that I want you to continue processing."

Raymond Chen

Follow