Eventually, nothing is special any more

devblogs.microsoft.com/oldnewthing/20081006-00

October 6, 2008

Raymond Chen

Commenter ulric suggested that two functions for obtaining the "current" window should exist, one for normal everyday use and one for "special use" when you want to interact with windows outside your process.

> I'd be more at ease however if the default behaviour of the API was to return HWND for the current process only, and the apps that really need HWND from other potentially other processes would have to be forced to use another API that is specifically just for that.

This is an excellent example of suggesting something that Windows already does. The special function has become so non-special, you don't even realize any more that it's special.

Originally, in 16-bit Windows, the function for getting the "current" window was `GetActiveWindow`. This obtained the active window across the entire system. One of the major changes in Win32 is the asynchronous input model, wherein windows from different input queues receive separate input. That way, one program that has stopped responding to input doesn't clog up input for other unrelated windows. Win32 changed the meaning of `GetActiveWindow` to mean *the active window from the current input queue*.

In 16-bit Windows, there was only one input queue, the global one. In 32-bit Windows, each thread (or group of input-attached threads) gets its own input queue.

As a result of this finer granularity, when a program was ported from 16-bit Windows to 32-bit Windows, it didn't "see" windows from other programs when it called functions like `GetFocus` or `GetActiveWindow`. As every Win32 programmer should know, these states are local to your input queue.

Okay, let's look at what we've got now. `GetFocus` and `GetActiveWindow` give you the status of your input queue. In other words, in a single-threaded program (which, if you're coming from 16-bit Windows, is the only type of program there is), calling `GetActiveWindow` gives you the active window from your program. It doesn't return the active window from another program.[1] Things are exactly as ulric suggested!

Now let's look at the second half of the suggestion. If a program really needs to get a window from potentially other processes, it would have to use some other function that is specifically just for that. And indeed, that's why the `GetForegroundWindow` function was added. The `GetForegroundWindow` function is *the special function specifically designed for obtaining windows from other processes*.

Therefore, we did exactly what ulric recommended, and it still turned into a mess. Why?

Because once you create something special, it doesn't remain special for long.

It may take a while, but eventually people find that the regular function "doesn't work" (for various definitions of "work"), and they ask around for help. "When I call `GetActiveWindow`, I'm not getting the global active window; I'm just getting the local one. How do I get the global one?" Actually, they probably don't even formulate the question that clearly. It's probably more like "I want to get the active window, but `GetActiveWindow` doesn't work."

And then somebody responds with "Yeah, `GetActiveWindow` doesn't work. I've found that `GetForegroundWindow` works a lot better."

The response is then "Wow, that works great! Thanks!"

Eventually, the word on the street is "`GetActiveWindow` doesn't work. Use `GetForegroundWindow` instead." Soon, people are using it for everything, waxing their car, calming a colicky baby, or improving their sexual attractiveness.

What used to be a function to be used "only in those rare occasions when you really need it" has become "the go-to function that gets the job done."

In fact, the unfashionableness of the active window has reached the point that people have given up on calling it the active window at all! Instead, they call it the foreground window from the current process. It's like calling a land line a "wired cell phone".

Requiring a new flag to get the special behavior doesn't change things at all. It's the same story, just with different names for the characters. "`GetFocalWindow`[2] doesn't work unless you pass the `GFW_CROSSPROCESS` flag." Soon, everybody will be passing the `GFW_CROSSPROCESS` not because they understand what it does but just because "That's what I was told to do" and "It doesn't work if I don't pass it."

Footnotes

[1]Assuming you haven't run around attaching your thread to some other program's input queue. This is a pretty safe assumption since the `AttachThreadInput` function didn't exist in 16-bit Windows either.

[2] `GetFocalWindow` is an imaginary function created for the purpose of the example.

[Raymond is currently away; this message was pre-recorded.]

Raymond Chen

Follow