If it's possible to do something, then it's possible to do something WRONG

devblogs.microsoft.com/oldnewthing/20110523-00

May 23, 2011

Raymond Chen

Once you make it possible to do something, you have to accept that you also made it possible to do something *wrong*.

When the window manager was originally designed, it made it possible for programs to override many standard behaviors. They could handle the `WM_NCHITTEST` message so a window can be dragged by grabbing any part of the window, not just the caption bar. They could handle the `WM_NCPAINT` message to draw custom title bars. The theory was that making all of these things possible permitted smart people to do clever things.

The downside is that it also permits stupid people to do dumb things.

Changing the window procedure model from *call* `DefWindowProc` *to get default behavior* to *return whether you handled the message* wouldn't have helped. First of all, the handled/not-handled model is too restrictive: It requires you to do *everything* (handled) or *nothing* (not handled). There is no option to do *a little bit*. (Imagine if C++ didn't let you call the base class implementation of an overridden method.)

Doing *a little bit* is a very common pattern. The `WM_NCHITTEST` technique mentioned above, for example, uses the default hit-testing implementation, and then tweaks the result slightly:

```
case WM_NCHITTEST:
 // call base class first
 lres = DefWindowProc(hwnd, uMsg, wParam, lParam);
 // tweak the result
 if (lres == HTCLIENT) lres = HTCAPTION;
 return lres;
```

How would you do this with the handled/not-handled model?

```
case WM_NCHITTEST:
 if (not handling this message would have resulted in HTCLIENT) {
  lres = HTCAPTION;
  handled = TRUE;
 } else {
  handled = FALSE;
 }
 break;
```

The trick about that bit in parentheses is that it requires the research department to finish the final details on that time machine they've been working on. It's basically saying, "Return *not handled*, then follow the message until handling is complete and if the final result is `HTCLIENT`, then fire up the time machine and rewind to this point so I can change my mind and return *handled* instead."

And even if the research department comes through with that time machine, the handled/not-handled model doesn't even solve the original problem!

The original problem was people failing to call `DefWindowProc` when they decided that they didn't want to handle a message. In the handled/not-handled model, the equivalent problem would be people returning `handled = TRUE` unconditionally.

```
BOOL NewStyleWindowProc(HWND hwnd, UINT uMsg,
 WPARAM wParam, LPARAM lParam, LRESULT& lres)
{
 BOOL handled = TRUE;
 switch (uMsg) {
 case WM_THIS: ...; break;
 case WM_THAT: ...; break;
 // no "default: handled = FALSE; break;"
 }
 return handled;
}
```

(Side note: The dialog manager uses the handled/not-handled model, and some people would prefer that it use the `DefXxxProc` model, so you might say "We tried that, and some people didn't like it.")

This topic raises another one of those "No matter what you do, somebody will call you an idiot" dilemmas. On the one side, there's the *Windows should perform extra testing at runtime to detect bad applications* school, and on the other side, there's the *Windows should get rid of all the code whose sole purpose in life is to detect bad applications* school.

Raymond Chen

Follow