Why doesn't RealGetWindowClass return the real window class for my superclass?

devblogs.microsoft.com/oldnewthing/20120713-00

July 13, 2012

Raymond Chen

A customer was reporting that the `RealGetWindowClass` function was not reporting the base window class of their superclass. (Error checking has been elided for expository purposes.)

```
// Get the static window class window procedure
WNDCLASS wc;
GetClassInfo(NULL, TEXT("static"), &wc);
WNDPROC StaticWndProc = wc.lpfnWndProc;
// Build our derived class
wc.lpfnWndProc = AwesomeWndProc;
wc.hInstance = g_hinst;
wc.lpszClassName = TEXT("AwesomeWindow");
RegisterClass(&wc);
LRESULT CALLBACK AwesomeWndProc(HWND hwnd, UINT uMsg,
                                WPARAM wParam, LPARAM lParam)
{
    TCHAR szClass[128];
    RealGetWindowClass(hwnd, szClass, 128);
    ASSERT(strcmp(szClass, TEXT("static")) == 0);
    switch (uMsg) { ... }
    return CallWindowProc(StaticWndProc, hwnd, uMsg, wParam, lParam);
}
```

The customer found that the assertion fails, returning a window class name of "AwesomeWindow" instead of "static". "I thought the point of RealGetWindowClass was to dig through the superclassing to find the base class. But it's not returning the base class."

That's right, because you haven't told it what the base class is yet!

"What do you mean I haven't told it? It's right there at the end of my function: `CallWindowProc(StaticWndProc)`."

Yeah, but that line of code hasn't executed yet. The external behavior of your program is like this:

```
WNDCLASS wc;
wc.style = (something);
wc.lpfnWndProc = AwesomeWndProc;
wc.cbClsExtra = (something);
wc.cbWndExtra = (something);
wc.hInstance = g_hinst;
wc.hIcon = (something);
wc.hCursor = (something);
wc.hbrBackground = (something);
wc.lpszMenuName = (something);
wc.lpszClassName = TEXT("AwesomeWindow");
RegisterClass(&wc);
LRESULT CALLBACK AwesomeWndProc(HWND hwnd, UINT uMsg,
                                WPARAM wParam, LPARAM lParam)
{
    TCHAR szClass[128];
    RealGetWindowClass(hwnd, szClass, 128);
    ASSERT(strcmp(szClass, TEXT("static")) == 0);
    // ... it doesn't matter what goes here
    // because it hasn't executed yet ...
```

The window manager isn't clairvoyant. It doesn't know that `AwesomeWndProc` is going to do a `CallWindowProc(StaticWndProc)` in the future. All it knows is that somebody registered a class, and then in response to its very first message, that class asked, "Hey, you're so smart, tell me what my base class is."

The window manager says, "Dude, you haven't shown me any base class yet. So I'm just going to say that you are your own base class."

Since anything can go into the "... it doesn't matter what goes here ...", we can demonstrate that the window manager cannot possibly know what you're going to pass to `CallWindowProc` by rewriting it like this:

```
// Get the static window class window procedure
WNDCLASS wc;
GetClassInfo(NULL, TEXT("static"), &wc);
WNDPROC StaticWndProc = wc.lpfnWndProc;
// Build our class
wc.lpfnWndProc = AwesomeWndProc;
wc.hInstance = g_hinst;
wc.lpszClassName = TEXT("AwesomeWindow");
RegisterClass(&wc);
LRESULT CALLBACK AwesomeWndProc(HWND hwnd, UINT uMsg,
                                WPARAM wParam, LPARAM lParam)
{
    TCHAR szClass[128];
    RealGetWindowClass(hwnd, szClass, 128);
    ASSERT(strcmp(szClass, TEXT("static")) == 0);
    switch (uMsg) { ... }
    // Psych! You thought that when I asked for StaticWndProc
    // I was going to be a superclass of "static", but in fact
    // I'm just a regular boring window class.
    return DefWindowProc(hwnd, uMsg, wParam, lParam);
}
```

If you felt really crazy, you could do this:

```
// Get the button window class procedure
WNDCLASS wcButton;
GetClassInfo(NULL, TEXT("button"), &wcButton);
WNDPROC ButtonWndProc = wcButton.lpfnWndProc;
// Get the static window class window procedure
WNDCLASS wc;
GetClassInfo(NULL, TEXT("static"), &wc);
WNDPROC StaticWndProc = wc.lpfnWndProc;
// Build our class
wc.lpfnWndProc = AwesomeWndProc;
wc.hInstance = g_hinst;
wc.lpszClassName = TEXT("AwesomeWindow");
wc.cbWndExtra = max(wc.cbWndExtra, wcButton.cbWndExtra);
wc.cbClsExtra = max(wc.cbClsExtra, wcButton.cbClsExtra);
RegisterClass(&wc);
LRESULT CALLBACK AwesomeWndProc(HWND hwnd, UINT uMsg,
                                WPARAM wParam, LPARAM lParam)
{
    TCHAR szClass[128];
    RealGetWindowClass(hwnd, szClass, 128);
    ASSERT(strcmp(szClass, TEXT("static")) == 0);
    switch (uMsg) { ... }
    // Decide at the last moment what we are.
    static WNDPROC BaseClass = nullptr;
    if (BaseClass == nullptr)
    {
        BaseClass = rand() % 2 ? StaticWndProc : ButtonWndProc;
        // Or if you are particularly perverse,
        // BaseClass = radioactive_decay_has_occurred() ?
        //                StaticWndProc : ButtonWndProc;
    }
    return CallWindowProc(BaseClass, hwnd, uMsg, wParam, lParam);
}
```

Since the code to decide the base class hasn't run yet, the window manager will have to use that time machine that the research division has been working on.

Raymond Chen

Follow