Why do I have to add 1 to the color index when I set it as the hbrBackground of a window class?

devblogs.microsoft.com/oldnewthing/20140305-00

March 5, 2014

Raymond Chen

Our scratch program sets the background color to `COLOR_WINDOW` by setting the class background brush as follows:

```
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
```

What's with the `+1`?

Okay, first of all, let's backtrack a bit.

The real first question is, "What's the deal with taking an integer (`COLOR_WINDOW`) and casting it to a `HBRUSH` and expecting anything sane to happen?"

The window manager wants to provide multiple ways of setting the class background brush.

1. The application can request that no automatic background drawing should occur at all.
2. The application can request custom background drawing and provide that custom drawing by handling the `WM_ERASEBKGND` message.
3. The application can request that the background be a specific brush provided by the application.
4. The application can request that the background be a specific system color.

The first three cases are easy: If you don't want automatic background drawing, then pass the hollow brush. If you want custom background drawing, then pass `NULL` as the brush. And if you want background drawing with a specific brush, then pass that brush. It's the last case that is weird.

Now, if `RegisterClass` were being invented today, we would satisfy the last requirement by saying, "If you want the background to be a system color, then use a system color brush like this:

```
wc.hbrBackground = GetSysColorBrush(COLOR_WINDOW);
```

System color brushes match the corresponding system color, so this sets your background to whatever the current system window color is."

But just as NASA couldn't use the Space Shuttle to rescue the Apollo 13 astronauts, the `RegisterClass` function couldn't use `GetSysColorBrush` for class brushes: At the time `RegisterClass` was designed, system color brushes had not yet been invented yet. In fact, they won't have been invented for over a decade.

Therefore, `RegisterClass` had to find some way of smuggling an integer inside a pointer, and the traditional way of doing this is to say that certain numerically-small pointer values are actually integers in disguise. We've seen this with the `HINSTANCE` returned by `ShellExecute`, with the `MAKEINTATOM` macro, with the `MAKEINTRESOURCE` / `IS_INTRESOURCE` macro pair, and with the second parameter to the `GetProcAddress` function. (There are plenty of other examples.)

The naïve solution would therefore be to say, "Well, if you want a system color to be used as the brush color, then just cast the `COLOR_XXX` value to an `HBRUSH`, and the `RegisterClass` function will recognize it as a smuggled integer and treat it as a color code rather than an actual brush."

And then you run into a problem: The numeric value of `COLOR_SCROLLBAR` is zero. Casting this to a `HBRUSH` would result in a `NULL` pointer, but a `NULL` brush already means something else: Don't draw any background at all.

To avoid this conflict, the `RegisterClass` function artificially adds 1 to the system color number so that none of its smuggled integers will be mistaken for `NULL`.

Raymond Chen

Follow