How do I enable and disable the minimize, maximize, and close buttons in my caption bar?

devblogs.microsoft.com/oldnewthing/20100604-00

June 4, 2010

Raymond Chen

A customer was having problems with the small icon that appears in the upper left corner of the caption:

> In my program, I need to enable and disable the Close button programmatically, since the program sometimes goes into a state where I don't want the user to close it. I do this by removing the `WS_SYSMENU` style when I want to disable the Close button, and adding it back when I want to re-enable it. However, doing this has as a side effect that the icon for my program doesn't appear in the title bar any more. If I never touch the `WS_SYSMENU` style, then it works fine (but then I don't get the enable/disable behavior that I want).

Okay, the first problem is that you want to disable the Close button. Usability research indicates that users really don't like it when you disable the Close button. It makes them feel trapped and uncomfortable. This is the reason why standard Windows wizards don't provide a way for you to disable the Close button. The user should always have a way out.

Imagine showing a dialog box saying "By clicking OK, you agree to the following terms," and not having any way for the user to say, "No, thanks." You should leave the Close button enabled, and if the user clicks it at a bad time, you can say, "Okay, I heard you, but I'm right in the middle of something, but once that's done, I'll close."

Okay, but suppose you're in one of those cases where, really, you need to disable the Close button. You've read the guidelines, you understand why they're there, but you believe that you are an exceptional case.

If you never want the Close button enabled, then you can just specify the `CS_NOCLOSE` style when you register the class.

To disable the Close button in the caption dynamically, you enable and disable the `SC_CLOSE` menu item in your system menu.

```
void DisableCloseButton(HWND hwnd)
{
 EnableMenuItem(GetSystemMenu(hwnd, FALSE), SC_CLOSE,
                MF_BYCOMMAND | MF_DISABLED | MF_GRAYED);
}
void EnableCloseButton(HWND hwnd)
{
 EnableMenuItem(GetSystemMenu(hwnd, FALSE), SC_CLOSE,
                MF_BYCOMMAND | MF_ENABLED);
}
```

The other two caption buttons are controlled by window styles:

```
void DisableMinimizeButton(HWND hwnd)
{
 SetWindowLong(hwnd, GWL_STYLE,
               GetWindowLong(hwnd, GWL_STYLE) & ~WS_MINIMIZEBOX);
}
void EnableMinimizeButton(HWND hwnd)
{
 SetWindowLong(hwnd, GWL_STYLE,
               GetWindowLong(hwnd, GWL_STYLE) | WS_MINIMIZEBOX);
}
void DisableMaximizeButton(HWND hwnd)
{
 SetWindowLong(hwnd, GWL_STYLE,
               GetWindowLong(hwnd, GWL_STYLE) & ~WS_MAXIMIZEBOX);
}
void EnableMaximizeButton(HWND hwnd)
{
 SetWindowLong(hwnd, GWL_STYLE,
               GetWindowLong(hwnd, GWL_STYLE) | WS_MAXIMIZEBOX);
}
```

Why is the close button managed differently from the minimize and maximize buttons?

History.

Originally, the window caption had only two buttons in the upper right corner, the minimize and maximize buttons, and they were controlled with a window style. Windows 95 added the Close button, but then there was the question of knowing when to enable and disable it. But wait, we already know when to enable and disable it: The application told us when it enabled and disabled the `SC_CLOSE` menu item. Bingo, just hook up the Close button to the existing menu item (which applications were already in the habit of maintaining), and magic, it just works. No need for applications to write special code to support the Close button. They already wrote the code; they just didn't realize it!

Exercise: What's wrong with these alternative functions for enabling and disabling the Close button:

```
// code in italics is wrong
void DisableCloseButton(HWND hwnd)
{
 SetClassLong(hwnd, GCL_STYLE,
              GetClassLong(hwnd, GCL_STYLE) | CS_NOCLOSE);
}
void EnableCloseButton(HWND hwnd)
{
 SetClassLong(hwnd, GCL_STYLE,
              GetClassLong(hwnd, GCL_STYLE) & ~CS_NOCLOSE);
}
```