A common mistake when you try to create a C++ class that wraps a window procedure: Saving the window handle too late

devblogs.microsoft.com/oldnewthing/20191014-00

October 14, 2019

Raymond Chen

A common mistake when you try to create a C++ class that wraps a window procedure is saving the window handle too late.

```
// Code in italics is wrong.
class MyWindowClass
{
private:
 HWND m_hwnd = nullptr;

 static LRESULT CALLBACK StaticWndProc(
    HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
 {
  MyWindowClass *self;
  if (uMsg == WM_NCCREATE) {
   LPCREATESTRUCT lpcs = reinterpret_cast<LPCREATESTRUCT>(lParam);
   self = static_cast<MyWindowClass*>(lpcs->lpCreateParams);
   SetWindowLongPtr(hwnd, GWLP_USERDATA,
                    reinterpret_cast<LONG_PTR>(self));
  } else {
   self = reinterpret_cast<MyWindowClass*>(
               GetWindowLongPtr(hwnd, GWLP_USERDATA));
  }
  if (self) {
   return self->WndProc(uMsg, wParam, lParam);
  }
  return DefWindowProc(hwnd, uMsg, wParam, lParam);
 }

 LRESULT WndProc(UINT uMsg, WPARAM wParam, LPARAM lParam)
 {
   switch (uMsg) {
   ...
   default:
    return DefWindowProc(m_hwnd, uMsg, wParam, lParam);
   }
 }

public:
 void CreateTheWindow()
 {
   ... RegisterClass() ...
   m_hwnd = CreateWindowEx(..., this);
 }
};
```

This code follows the usual pattern for a window procedure wrapper: The `this` pointer is passed as the creation parameter, and the `WM_ NCCREATE` message handler stashes the creation parameter in the window extra bytes, thereby allowing the `this` pointer to be recovered from the window handle when handling future messages.

However, there's a problem with the above code. Can you spot it?

The problem is that it sets the `m_hwnd` member variable too late.

As written, the code doesn't set the `m_hwnd` member variable until the `CreateWindowEx` function returns. But creating a window involves sending many messages.

For every message received during window creation, The `WndProc` member function runs with a null `m_hwnd`. This means that when it calls `DefWindowProc(m_hwnd, ...)`, it's passing an invalid parameter.

Many of the messages sent during window creation are kind of important to pass through to `DefWindowProc`. For example, if you neglect to pass `WM_ NCCREATE` to `DefWindowProc`, your window will not be properly initialized.

The solution is to set `m_hwnd` as soon as you learn what the window handle is.

```
if (uMsg == WM_NCCREATE) {
 LPCREATESTRUCT lpcs = reinterpret_cast<LPCREATESTRUCT>(lParam);
 self = static_cast<MyWindowClass*>(lpcs->lpCreateParams);
 self->m_hwnd = hwnd; // save the window handle too!
 SetWindowLongPtr(hwnd, GWLP_USERDATA,
                  reinterpret_cast<LONG_PTR>(self));
}
```

Don't wait for `CreateWindowEx` to return. By then, it's too late.

Raymond Chen

Follow