How can I have a window that rejects activation but still receives pointer input?

devblogs.microsoft.com/oldnewthing/20160912-00

September 12, 2016

Raymond Chen

A customer had a dedicated system with two touch screens. One screen was covered by the main app window, and the other was covered by a secondary window. They needed focus to remain on the main app window because reasons.[1]

One way of preventing the secondary window from getting focus is to use the `WS_EX_NOACTIVATE` extended style. Another way is to disable it. However, these cause the secondary window to ignore input, but the customer also wanted the user to be able to interact with the secondary window. Can they have their cake and eat it too?

Let's start with the new scratch program and make these changes. The first set of changes is basically the stuff we did in an earlier article to turn the main window into a logging window.

```
#include <strsafe.h>

class RootWindow : public Window
{
public:
 ...
 void AppendText(LPCTSTR psz)
 {
  ListBox_SetCurSel(m_hwndChild,
      ListBox_AddString(m_hwndChild, psz));
 }

 void LogMessage(UINT uMsg, WPARAM wParam, LPARAM lParam)
 {
  TCHAR szMsg[80];
  StringCchPrintf(szMsg, 80, TEXT("%d\t0x%04x\t%p\t%p"),
      GetTickCount(),
      uMsg,
      wParam,
      lParam);
  AppendText(szMsg);
 }
 ...
};
```

The logging comes from the side window:

```
class SideWindow : public Window
{
public:
 SideWindow(RootWindow* prw) : m_prw(prw) {}
 virtual LPCTSTR ClassName() { return TEXT("SideWindow"); }
 static SideWindow *Create(RootWindow* prw);
protected:
 LRESULT HandleMessage(UINT uMsg, WPARAM wParam, LPARAM lParam);
private:
 RootWindow* m_prw;
};

LRESULT SideWindow::HandleMessage(
 UINT uMsg, WPARAM wParam, LPARAM lParam)
{
 switch (uMsg) {
 case WM_MOUSEACTIVATE:
  m_prw->LogMessage(uMsg, wParam, lParam);
  return MA_NOACTIVATE;
 case WM_MOUSEMOVE:
 case WM_LBUTTONDOWN:
 case WM_LBUTTONUP:
  m_prw->LogMessage(uMsg, wParam, lParam);
  break;
 }

 return __super::HandleMessage(uMsg, wParam, lParam);
}

SideWindow *SideWindow::Create(RootWindow* prw)
{
 SideWindow *self = new SideWindow(prw);
 if (self && self->WinCreateWindow(0,
     TEXT("SideWindow"), WS_OVERLAPPEDWINDOW,
     CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
     NULL, NULL)) {
     return self;
 }
 delete self;
 return NULL;
}
```

The side window logs selected mouse messages so we can see what's going on. The interesting thing is that it responds to the `WM_MOUSEACTIVATE` with `MA_NOACTIVATE`, which means, "Thanks for your interest in my window, but I decline your offer to activate me." Another way to decline activation is to return `MA_NOACTIVATEANDEAT`, which goes a step further and says, "Throw away the input that caused you to want to activate this window." That's not what we want today, because we want to keep the input; we simply don't want activation.

Let's finish up the program before discussing further.

```
int PASCAL
WinMain(HINSTANCE hinst, HINSTANCE, LPSTR, int nShowCmd)
{
 ...
  RootWindow *prw = RootWindow::Create();
  if (prw) {
      ShowWindow(prw->GetHWND(), nShowCmd);
      SideWindow *sw = SideWindow::Create(prw);
      ShowWindow(sw->GetHWND(), SW_SHOWNA);
      MSG msg;
      while (GetMessage(&msg, NULL, 0, 0)) {
          TranslateMessage(&msg);
          DispatchMessage(&msg);
      }
  }
  ...
}
```

Okay, run this program, and it will open two windows. (I didn't bother putting each one on a separate monitor. You can use your imagination.) While focus is on the main window, use your finger or moues to click on the second window. Observe that the second window does not activate, but the logging window shows that it did receive the `WM_LBUTTONDOWN = 0x0201` message. Drag your finger over the window, or drag the mouse, and you'll see the `WM_MOUSEMOVE = 0x0200` messages, and you'll get a `WM_LBUTTONUP = 0x0202` message when the pointer goes up.

So there you have it: A window that rejects activation but stil receives touch and mouse input.

Raymond Chen

Follow