Dialog boxes return focus to the control that had focus when you last switched away; how do I get in on that action for my own windows?

devblogs.microsoft.com/oldnewthing/20140521-00

May 21, 2014

Raymond Chen

When you switch away from a dialog box, and then switch back to the dialog box, focus returns to the control that had focus when you last left the dialog box. But if you have a window that manually hosts two controls, then when the user switches away from your window, and then switches back, focus goes to, um...

Let's find out what happens! Take our scratch program and make these changes.

```
HWND g_hwndChild2;

void
OnSize(HWND hwnd, UINT state, int cx, int cy)
{
 // if (g_hwndChild) {
 //    MoveWindow(g_hwndChild, 0, 0, cx, cy, TRUE);
 // }
}

BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
 g_hwndChild = CreateWindow(TEXT("button"), TEXT("&1"),
                            WS_CHILD | WS_VISIBLE |
                            WS_TABSTOP | BS_PUSHBUTTON,
                            0, 0, 100, 100,
                            hwnd, nullptr, g_hinst, 0);
 g_hwndChild2 = CreateWindow(TEXT("button"), TEXT("&2"),
                            WS_CHILD | WS_VISIBLE |
                            WS_TABSTOP | BS_PUSHBUTTON,
                            100, 0, 100, 100,
                            hwnd, nullptr, g_hinst, 0);
 return TRUE;
}

// message loop

 while (GetMessage(&msg, NULL, 0, 0)) {
  if (IsDialogMessage(hwnd, &msg)) continue;
  TranslateMessage(&msg);
  DispatchMessage(&msg);
 }
```

Run the program, and the first thing you notice is that *neither button* has keyboard focus. Focus is on the frame window, not that that helps you much, since the frame window ignores keyboard input.

Well, anyway, hit the `Tab` key to put focus on one of the buttons, then switch away to some other application, then switch back via `Alt`+`Tab` or clicking on the taskbar button. (Just don't click on the window itself, because that would interfere with the experiment, since the click also sets focus to the window you clicked on.)

Again, focus is not on either button because it's on the frame window.

Focus is on the frame window because nobody bothered to put it anywhere else. Let's fix that. Let's say that focus goes to the first child button when the user activates the main frame window.

```
void OnActivate(HWND hwnd, UINT state,
                HWND hwndActDeact, BOOL fMinimized)
{
 if (state != WA_INACTIVE && !fMinimized) {
  SetFocus(g_hwndChild);
 }
}

 HANDLE_MSG(hwnd, WM_ACTIVATE, OnActivate);
```

There's a little gotcha here: We don't want to do this when minimized. When the window is activated while minimized, the user can't see any of the child windows, so putting focus on the child causes the user's keypresses to start doing invisible things. (If the user hits the space bar, it will push the invisible button!) Instead, focus should stay on the frame window. It is not well-known, but if you hit the space bar when focus is on a minimized window, it will open the system menu. Keeping focus on the frame when minimized preserves this behavior.

Okay, at least this time, focus goes somewhere when the user activates our window. Of course, it would be better if we restored focus to where it was when the user last used our window. (For one thing, it means that the default pushbutton effect is more likely to remain correct!) If we don't know where to restore the focus to, then we fall back to using the first child window.

```
HWND g_hwndLastFocus;
void OnActivate(HWND hwnd, UINT state,
                HWND hwndActDeact, BOOL fMinimized)
{
 if (!fMinimized) {
  if (state == WA_INACTIVE) {
   HWND hwndFocus = GetFocus();
   if (hwndFocus && IsChild(hwnd, hwndFocus)) {
       g_hwndLastFocus = hwndFocus;
   }
  } else {
   SetFocus(g_hwndLastFocus ? g_hwndLastFocus
                            : g_hwndChild);
  }
 }
}

 HANDLE_MSG(hwnd, WM_ACTIVATE, OnActivate);
```

The basic idea is that when the window is deactivated, we remember the window that had focus, and when the window is reactivated, we restore the focus to that same window (or to our first child if we don't know what the previous focus was).

Again, it's important to watch out for the minimized window case. If the window is minimized when the user deactivates it, we would end up saving the frame window for future restoration, when in fact we should just ignore the entire interaction while minimized.

Note that even when not minimized, we do not try to save focus if it belongs to something outside our frame window. For our simple program, this is more of a safety check than something we expect to happen.

Next time, we'll see an unexpected consequence of this auto-restore behavior.

Raymond Chen

Follow