Make sure you disable the correct window for modal UI

devblogs.microsoft.com/oldnewthing/20061102-02

November 2, 2006

Raymond Chen

Some time ago, I was asked to look at two independent problems with people trying to do modal UI manually. Well, actually, when the issues were presented to me, they weren't described in quite that way. They were more along the lines of, "Something strange is happening in our UI. Can you help?" Only in the discussion of the scenarios did it become apparent that it was improper management of modal UI that was the cause.

We already saw one subtlety of managing modal UI manually, namely that you have to enable and disable the windows in the correct order. That wasn't the root of the problems I was looking at, but enabling and disabling windows did play a major role.

When we took a look at the dialog loop, the first steps involved manipulating the `hwndParent` parameter to ensure that we enable and disable the correct window at the correct time.

```
if (hwndParent == GetDesktopWindow())
 hwndParent = NULL;
if (hwndParent)
 hwndParent = GetAncestor(hwndParent, GA_ROOT);
HWND hdlg = CreateDialogIndirectParam(hinst,
              lpTemplate, hwndParent, lpDlgProc,
              lParam);
BOOL fWasEnabled = EnableWindow(hwndParent, FALSE);
```

In both cases, the first two "if" statements were missing. We already saw the danger of disabling the desktop window, which is what the first "if" statement protects against. But the specific problem with modal UI was being caused by the missing second "if" statement.

Both of the problems boiled down to somebody passing a child window as the `hwndParent` and the code doing manual modal UI failing to convert this window to a top-level window. As a result, when they did the `EnableWindow(hwndParent, FALSE)`, they disabled a child window, leaving the top-level window enabled.

The two problems had the same root cause but manifested themselves differently. The first problem led to strange behavior because the user could still interact with the top-level window since it was still enabled. Sure, a portion of the window was disabled (the portion controlled by the child window passed as `hwndParent`), but the caption buttons still worked, as did many of the other controls on the window.

In the second case, disabling the wrong window created a different problem: When the modal UI was complete, the window manager activated the top-level window that was the owner of the modal window since that window was never disabled. This caused the top-level window to receive a `WM_ACTIVATE` message, which it handled by putting focus on the control that had focus when the top-level window was deactivated. Unfortunately, that window was the window that was passed as the `hwndParent`, which was *disabled by mistake*. The attempt to restore focus failed, and when the manual modal UI finally finished up and enabled the child window, it was too late. You wound up with focus nowhere and a dead keyboard. This second problem was reported as simply "`SetFocus` is not working." Only after peeling back a few layers (and application of some psychic powers) did the root cause emerge.

Now, even though this was a subtle problem, you already knew all the pieces that went into it since I had covered them earlier. And as for those psychic powers that I used? It's really not that magic. In this case of psychic debugging, I worked backwards. In response to the report that `SetFocus` was not working, the next set of questions was to determine why. Is it a valid window handle? Does the window belong to your thread? Is it enabled?

Aha, the window isn't enabled. That's when the customer also mentioned that they were doing this inside a `WM_ACTIVATE` handler. If you're gaining activation, who were you gaining it from? Oh, a modal dialog, you say? One that you're managing manually? Once I discovered that they were trying to manage modal UI manually, I suspected that they were disabling the wrong window, since that fit all the symptoms and it's something that people tend to get wrong.

Most of what looks like psychic debugging is really just knowing what people tend to get wrong.

Raymond Chen

Follow