Why does my radio button group selection get reset each time my window regains activation?

devblogs.microsoft.com/oldnewthing/20140522-00

May 22, 2014

Raymond Chen

A customer reported (all incomplete information and red herrings preserved):

> We have an issue related to two radio buttons in a window. The code programmatically checks the second button by sending the `BM_SETCHECK` message. We observe that if the user clicks somewhere else on the screen (so that our application loses focus), and then clicks on the taskbar icon to return to our application, the first radio button spontaneously gets selected.
>
> We watched all the messages in Spy++, and it appears that the radio button is receiving a `WM_SETFOCUS` followed by a `WM_SETCHECK`.
>
> Is this by design? If not, what should I be looking for in my code that is causing this erroneous selection change to occur?

The incomplete information is that the customer didn't say how they created those radio buttons.

The red herring is that the customer said that they had a problem with their window. This suggested that they were doing a custom window implementation (because if they were using the standard dialog implementation, they would have said *dialog*).

But from the symptoms, it's clear that what's most likely happening is that the radio button is created as a `BS_AUTORADIOBUTTON`. And automatic radio buttons select themselves automatically (hence the name) when they receive focus.

That explains the message sequence of `WM_SETFOCUS` followed by a `WM_SETCHECK`: The automatic radio button receives focus, and in response it checks itself.

Therefore, the next level of investigation is why the first radio button is getting focus when the window is activated.

If the application window is a custom window, then the place to look is their window's activation and focus code, to see why focus is going to the first radio button instead of the second one. Perhaps it is putting focus on the first radio button temporarily, and then later realizes, "Oh wait, I really meant to put it on the second radio button." The fix would be to get rid of the temporary focus change and go straight to the second radio button.

If the application window is a standard dialog, then we saw last time that the dialog manager restores focus to the window that had focus last, and that you could mimic the same behavior in your own code.

It turns out that the customer was indeed using a standard dialog, in which case the problem is that they put the dialog into an inconsistent state: They checked the second radio button but left focus on the first radio button. This is a configuration that exists nowhere in nature, and therefore when the dialog manager tries to recreate it (given its lack of specialized knowledge about specific controls), it can't.

The fix is to put focus on the second radio button as well as setting the check box. In fact, you can accomplish both by setting the focus to the second radio button (noting that there is a special process for setting focus in a dialog box) since you already are using automatic radio buttons.

Here's a program that demonstrates the problem:

```
// scratch.rc
1 DIALOGEX 32, 32, 160, 38
STYLE DS_MODALFRAME | DS_SHELLFONT | WS_POPUP | WS_VISIBLE |
      WS_CAPTION | WS_SYSMENU
CAPTION "Test"
FONT 9, "MS Shell Dlg"
BEGIN
CONTROL "First", 100, "Button",
        WS_GROUP | WS_TABSTOP | BS_AUTORADIOBUTTON, 4,  4, 152, 13
CONTROL "Second", 101, "Button",BS_AUTORADIOBUTTON, 4, 20, 152, 13
END
// scratch.cpp
#include <windows.h>
#include <windowsx.h>
INT_PTR CALLBACK DlgProc(
    HWND hdlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
 switch (uMsg) {
 case WM_INITDIALOG:
  SetFocus(GetDlgItem(hdlg, 100));
  CheckRadioButton(hdlg, 100, 101, 101);
  return FALSE;
 case WM_COMMAND:
  switch (GET_WM_COMMAND_ID(wParam, lParam)) {
  case 100:
  case 101:
    CheckRadioButton(hdlg, 100, 101,
                     GET_WM_COMMAND_ID(wParam, lParam));
    break;
  case IDCANCEL: EndDialog(hdlg, 0); break;
  }
 }
 return FALSE;
}
int WINAPI WinMain(HINSTANCE hinst, HINSTANCE hinstPrev,
                   LPSTR lpCmdLine, int nShowCmd)
{
 DialogBox(hinst, MAKEINTRESOURCE(1), nullptr, DlgProc);
 return 0;
}
```

Observe that we set focus to the first button but check the second button. When the dialog regains focus, the second button will fire a `WM_COMMAND` because it thinks it was clicked on, and in response the dialog box moves the selection to the second button.

The fix here is actually pretty simple: Let the dialog manager handle the initial focus. Just delete the `SetFocus` call and return `TRUE`, which means, "Hey, dialog manager, you do the focus thing, don't worry about me."

Another fix is to remove the code that updates the radio buttons in response to the `WM_COMMAND` message. (I.e., get rid of the entire `case 100` and `case 101` handlers.) Again, just let the dialog manager do the usual thing, and everything will work out just fine.

It's great when you can fix a bug by deleting code.

Raymond Chen

Follow