Modality, part 5: Setting the correct owner for modal UI

devblogs.microsoft.com/oldnewthing/20050224-00

February 24, 2005

Raymond Chen

Here is the very simple fix for the buggy program we presented last time.

```
void OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
  switch (ch) {
  case ' ':
    MessageBox(hwnd, TEXT("Message"), TEXT("Title"), MB_OK);
    if (!IsWindow(hwnd)) MessageBeep(-1);
    break;
  }
}
```

We have fixed the problem by passing the correct owner window for the modal UI. Since `MessageBox` is modal, it disables the owner while the modal UI is being displayed, thereby preventing the user from destroying or changing the owner window's state when it is not expecting it.

This is why all the shell functions that can potentially display UI accept a window handle as one of its parameters. They need to know which window to use as the owner for any necessary UI dialogs. If you call such functions from a thread that is hosting UI, you must pass the handle to the window you want the shell to use as the UI owner. If you pass NULL (or worse, `GetDesktopWindow`), you may find yourself in the same bad state that our buggy sample program demonstrated.

If you are displaying a modal dialog from another modal dialog, it is important to pass the correct window as the owner for the second dialog. Specifically, you need to pass the modal dialog initiating the sub-dialog and not the original frame window. Here's a stack diagram illustrating:

```
MainWindow
 DialogBox(hwndOwner = main window) [dialog 1]
  ... dialog manager ...
   DlgProc
    DialogBox(hwndOwner = dialog 1) [dialog 2]
```

If you mess up and pass the main window handle when creating the second modal dialog, you will find yourself back in a situation analogous to what we had last time: The user can dismiss the first dialog while the second dialog is up, leaving its stack frames orphaned.

Raymond Chen

Follow