Subject: Having my HWND and eating it, too...
Posted by tvanriper on Wed, 10 Oct 2007 19:29:57 GMT
View Forum Message <> Reply to Message

I have a problem.

I'm not using an OCX control. I'm not using OLE or COM in any form for this. I have a function to a third-party library that takes a HWND as its argument.

The HWND needs to represent an area within a TopWindow to which the third-party library may render whatever it wants.

I created a custom class derived from Ctrl, but quickly discovered that Ctrl objects do not have their own HWNDs.

So, I tried creating deriving the class from TopWindow instead, after noticing all the happy fun function for creating HWNDs there. However, for some reason, when I call GetHWND() against this control, I still get a null.

So I'm stumped. At the moment, I am not sure how I should create this window. I've tried some other things, but they lead to assertions.

Does anyone have anything to suggest?

Subject: Re: Having my HWND and eating it, too...
Posted by Oblivion on Wed, 10 Oct 2007 19:56:26 GMT
View Forum Message <> Reply to Message

From where do you get a handle to your window?
If it is your applications main window, (if you are using e.g. mainwindow.run()) you shouldn't try to get a handle to your window in class constructor (unless you explicitly call Open() or OpenMain() -- which, in this case (if you already call run() in your apps main()) you shouldn't use).

You could use instead something safer:

```
HWND MainWindow::GetHandle()
{
return GetHWND(); // use IsOpen() to check before calling;
}
```

or

```
HWND MainWindow::GetHandle()
{
return IsOpen() ? GetHWND() : null; // handle null as an error;
}
```

and call whenever it's needed.

Subject: Re: Having my HWND and eating it, too...
Posted by tvanriper on Wed, 10 Oct 2007 20:09:46 GMT
View Forum Message <> Reply to Message

I'm not attempting to do this from within the constructor.

Let me see if I can describe the sequence of events...

1. Application starts.

2. Main window comes up. This one holds all the other windows for the entire application. By 'comes up', I mean that the user is able to see it, play with the knobs, fiddle with the switches, and perhaps give us some money in appreciation.

3. User logs in.

4. Some information flows from the other side of the network to this application, bearing some initialization information for my third-party control.

5. At this point, I want to get an HWND to a window within the main window (a control, if you will) to feed to my third-party functions.

Problem is, when I call GetHWND() on my control, I get '0', because I don't seem to actually have an HWND in my control. I can't give it the HWND for the main application window, because then my third-party function will write over the entire application, rather than the small area I want it to have.

Subject: Re: Having my HWND and eating it, too...
Posted by Oblivion on Wed, 10 Oct 2007 20:31:55 GMT
View Forum Message <> Reply to Message

Ah, I see. I'm converting my MFC-based multiprotocol (so a client-server system is used) messenger to UPP and encountered similar problems.

Well, AFAIK it is not possible that way. If I understand it right, you derive your child windows from TopWindow, but you do not Open() them. Unless they are opened, they cannot be TopWindows; they stay as ctrl (which -- conceptually -- has no HWND). On the other hand, if they are Opened() they cannot be child ctrls anymore, at most they can be child top or popup windows. If your client programs are in U++ too, you could pass ctrl* instead. But if they are not, and you are trying to "draw" something, you could pass a "Rect" and an ID to the ctrl (this means that you have to modify your apps IPC message framework.

Namely, as far as I know there is no HWND of ctrls "in the main window." (AFAIK, not explicitly).

Subject: Re: Having my HWND and eating it, too...
Posted by tvanriper on Wed, 10 Oct 2007 20:45:42 GMT
View Forum Message <> Reply to Message

I see a hint towards a solution, but I'm not there yet.

If you create a simple application with a WithDropChoice-type control, and set a breakpoint at Ctrl::Create (within Win32Wnd.cpp, which is in CtrlCore), you'll find that a window is created when you expand the drop choice.

Looking further in the code, it seems someone created a Ctrl class object and called its 'PopUp' function, which eventually calls a 'PopUpHWND' function, which itself executes (successfully) a Create().

I want to understand how this window is removed before I go too much further down this line.

I totally understand why U++ doesn't create a lot of windows (and, actually, I really applaud that), but I wish it were easier to create a sub-window. Or, perhaps it is easy, but I just don't see it yet.

Subject: Re: Having my HWND and eating it, too...
Posted by Oblivion on Wed, 10 Oct 2007 20:50:34 GMT
View Forum Message <> Reply to Message

Be careful...

That is a "Popupwindow" of dropchoice -- which exacly works in native U++ way. It is NOT the dropchoice ctrl (as a whole) and not all ctrl's have or are popups (as menus, dropchoices are).

In fact, you can popup any Ctrl by calling Ctrl::Popup().

But, if I understand you right, that will give you a "floating" popup window (such as popup menus); nothing more.

Subject: Re: Having my HWND and eating it, too...
Posted by tvanriper on Wed, 10 Oct 2007 21:06:44 GMT
View Forum Message <> Reply to Message

I understand that... but it was exactly the clue I needed to fix my problem.

It's a little gnarly, but my control now works mostly the way I want it to work (I need to fix a small problem related to positioning, but it's close).

I needed to create a new control derived from Upp::Ctrl. This control has a constructor (for no real reason, I could probably have ommitted this), and an 'Initialize' function that takes an HWND (the 'owner' HWND) and a Rect (the size of the window to create).

Initialize then calls Upp::Ctrl::Create(HWND owner, WS_CHILD | WS_VISIBLE, 0, false, SW_SHOW, true) to create the HWND.

I then added a member variable of this class in my control.

When I need to initialize the HWND, I call my new class's Initialize() function, which creates the necessary HWND of exactly the type I needed.

I'm able to feed this HWND into the functions I'm using, and it's painting to the area (except for being slightly above it... something I'm trying to fix now).

All-in-all, it works, but not quite as elegantly as I had hoped. It's still sufficiently U++-like (I think) to be acceptable.

Subject: Re: Having my HWND and eating it, too...
Posted by tvanriper on Wed, 10 Oct 2007 21:07:53 GMT
View Forum Message <> Reply to Message

(maybe, if I can think to do this, I'll create some example code to illustrate this solution a little better).

Subject: Re: Having my HWND and eating it, too...
Posted by Oblivion on Wed, 10 Oct 2007 21:11:19 GMT
View Forum Message <> Reply to Message

Sounds like a nice trick, indeed!

Subject: Re: Having my HWND and eating it, too...
Posted by tvanriper on Thu, 11 Oct 2007 17:07:41 GMT
View Forum Message <> Reply to Message

Well, some example code for the trick I used...

Here's the child that creates the HWND...

```
class ChildCtrl : public Upp::Ctrl
{
```

```
public:
 ChildCtrl() : Upp::Ctrl() {};
 void Initialize( HWND owner, const Upp::Ctrl& ctrl )
 {
  SetRect( ctrl.GetRect() );
  Create( owner, WS_CHILD | WS_VISIBLE, 0, false, SW_SHOW, true );
 };
};
```

Of course, you might need to change the styles to suite your own needs. This is, obviously, Windows-centric stuff, but if you had a need to do something like this for X11, I'm pretty sure you could use a similar technique.

Later, I make use of this within my other control as such:

```
class TestCtrl : public Upp::Ctrl
{
protected:

 ChildCtrl m_wnd;
public:
 TestCtrl()
 {
 };

 ~TestCtrl()
 {
 };

 void IncomingMessage( const SomeCommand& command )
 {
  if ( m_wnd.GetHWND() == 0 )
  {
   m_wnd.Initialize( ::GetActiveWindow(), *this );
  }
  ::DoSomeCommand( m_wnd.GetHWND(), command );
 };

 void EnableCtrl( bool enable = true )
 {
  if ( enable )
  {
   m_wnd.Show();
  }
  else
  {
```

```
    m_wnd.Hide();
  }
 };
};
```

The only thing not quite working the way I want is the size of window... the HWND seems to think it should appear above where you actually place TestCtrl.

I'm currently working around this by moving the control down well below where I really want it... but I want to fix this when I have some time. It's puzzling the SetRect() seems to be a little inaccurate for HWNDs. I'm guessing that U++ is compensating for something (I will have to dig in the code sometime to see).

Of course, it's also possible that ::DoSomeCommand() is actually putting the video off somewhere, but I kind of doubt it; we use ::DoSomeCommand() elsewhere, and it works just fine.

Using this technique, the control should get deallocated on destruction (our favorite U++ and C++ idiom, from what I've seen). The only awkward thing is that EnableCtrl() bit... I'm having to use this to hide and show the window when necessary. I could probably handle this more elegantly by overriding the Hide/Show commands in TestCtrl() to also hide and show the child control.

Subject: Re: Having my HWND and eating it, too...
Posted by mirek on Thu, 11 Oct 2007 22:09:25 GMT
View Forum Message <> Reply to Message

Please look at DHCtrl (and perhaps GlCtrl that uses it).

DHCtrl is a special "hack" kind of child widget with HWND (

Mirek
