Subject: Re: Form Designer
Posted by alendar on Fri, 05 Feb 2010 02:57:06 GMT
View Forum Message <> Reply to Message

Took me a while to figure it out, but its pretty cool once you do. Take a look at the examples and its straight forward.

1) insert a lay file into your project. You do that by right clicking in the bottom-left frame and selecting "Insert Package Directory File(s)". That was not intuitive for a while. I tried U++ a few months ago and that pissed me off so much I dumped it and went back to struggling with VC. Now I've been more tolerant of the product and its just "Not Microsoft". Plus its free so I can't expect it to behave like a $3,000 product. In fact IMO its better because I've been able to create some fun code, once I got past the little things.

2) Select the file, which opens it. You may have to press Ctrl+T if you see text. Then right-click in the grid and select an object to add. Again, the right-click to add frustrated me for a few seconds, but I got over it. The rubber band lines annoyed me too, until I figured out they actually made life VERY easy! Simply click on/off to lock it to that side of the window. Very slick.

U++ focuses on the things developers want to generally do, instead of trying to force you to use some new hair-brained framework, like ATL or WTL or MFC or WinForms or Managed C++ and .NET VBX then OCX then COM and OLE2 then ActiveX then I don't remember what.

3) name each item. These automagically become members in the class you include the layout in.

4) Create a window class (use an example). All it is is class W : public WithXXXLayout<TopWindow>.

5) Drop this in your constructor:

CtrlLayout(*this, "Title");

or CtrlLayoutOk or OkCancel.

You do have to be careful about window sizing code being after or before the Layout macro.

6) Then you .Run() the window from your Main.

Total Cake.

Now in your constructor, you type one of your objects and poof - you get a context-sensitive popup of all the bases of your window.

To do a menu:

1) Something like this in your class def:
MenuBar menuFrame; // Horizontal stack

```
void constructMenuFrame() {
 AddFrame(menuFrame);
 menuFrame.Set(THISBACK(menuFrameCallback));
}
void menuFrameCallback(Bar& bar) {
bar.Add("File", THISBACK(FileMenu));
bar.Add("Edit", THISBACK(EditMenu));
bar.Add("View", THISBACK(ViewMenu));
bar.Add("Help", THISBACK(HelpMenu));
void HelpMenu(Bar& bar) {
 bar.Add("About Jammit", THISBACK(About));
}
void About() {
 aboutWindow.Open();     }
```

2) Generate it in your constructor.
constructMenuFrame();

Can't get any easier.
Oh, to get callbacks you'll need this in your window class:
typedef MainWin CLASSNAME;

This is a simple thing that allows TopWindow or some such to generate the proper function call.

Other things.
1) Drop a file with extension ".rc" in there and U++ picks it up.
2) Drop an IML file in there and all your images are available.

I put this in my project share header:

```
#define IMAGECLASS MyImages
#define IMAGEFILE "project.iml"
#include <Draw/iml.h>
```

Then to set the icon I just go "this->LargeIcon(MyImages::bigmainicon);" in my window class.

That sets the Taskbar icon. The regular app icon I set in the resource file.

I threw a little Icon on a push button with:

playOrPause = MyImages::play();

On clicking I do "playOrPause = MyImages::pause()"

One learning curve issue is that the writers overrode more operators than a new rooster in a henhouse. The "~" was a surprise to me, but its fine if you just Alt+O to it and view the implementation.

Callbacks are so easy I'm very impressed. Qt's method seems much more complex.

One thing obviously better than MSVC: You can easily extend anything. I extended TopWindow without blinking. A few fanagles with default constructors, the close method, and such.

Now I have my own little custom Window wrapper. Try doin that in VC. Subclassing is a major effort.