Subject: "Visual Inheritance" possible?
Posted by aschoem on Fri, 08 Jun 2007 07:35:58 GMT
View Forum Message <> Reply to Message

I want to do the following:

1) I want to create a custom composite control, i.e. a "dialog" with a few elements, call it "DialogBase".
2) I then want to derive custom dialogs from DialogBase, adding some other controls to the dialog, call it "Derived1", "Derived2" etc.
3) When starting my work on Derived1 etc. I want to use DialogBase as base class. Layout Designer should show the layout of DialogBase (with all the controls) and let me position the controls of DialogBase, i.e. initialize the base class members. I then want to add my new controls and save that as Dialog1 layout.

That is what I mean by "visual inheritance". Is this possible? If it is not possible then it would be a great thing to have, imho.

Subject: Re: "Visual Inheritance" possible?
Posted by mrjt on Fri, 08 Jun 2007 10:32:53 GMT
View Forum Message <> Reply to Message

In sense you mean it this it currently impossible as a layout class (one that inherits from WithxxxLayout) cannot inherit from another layout class without causing ambiguous function calls. To be honest, the way you suggest doesn't make sonse in some cases. For instance, if you move a control on the base layout what would happen to the sub-layouts?

However, I think the effect can be achieved quite well.
1) Copy/paste the controls from the base to the child, either in the Designer or in text mode. This gives a form of visual inheritace.
2) I assume you really want common control behaviour from the base class the be automatic on the sub-class windows? This can be done in one of two ways:
2a) Do not have the base class inherit from WithxxxLayout, but instead declare the controls explicitly as members of the class. You can then code the common bahaviour and use the base instead of TopWindow as the template parameter to WithxxxLayout when inheriting.
2b) Use a construct like this:

(op1/2 and result are all EditInt ctrls, calc andclose are buttons)

```
template <class P>
class AWindow : public P {
public:
 typedef AWindow CLASSNAME;

 void Calc()
 {
```

```
  if (!Accept()) return;
  result <<= ((int)~op1 + (int)~op2);
 }
protected:
 AWindow()
 {
  CtrlLayout(*this, "Base class");

  op1.NotNull() <<= 1;
  op2.NotNull() <<= 2;
 }
};

class BWindow : public AWindow< WithBWindowLayout<TopWindow> > {
public:
 typedef BWindow CLASSNAME;

 BWindow()
 {
  Title("Adder");
  calc <<= THISBACK(Calc);
  close <<= THISBACK(Close);
 }
};
```

This is neater in that you don't have to declare the controls explicitly, but does mean that you can't have multiple layers of inheritance. I've attached the package I tested the above code in.

Hope that helps,
James

File Attachments

1) LayoutTest.zip, downloaded 583 times

Subject: Re: "Visual Inheritance" possible?
Posted by aschoem on Fri, 08 Jun 2007 15:27:32 GMT
View Forum Message <> Reply to Message

I'm a beginner with UPP so am not familiar with the template layout mechanism. Maybe you can explain the big advantage of this approach to me?

Maybe my description was a bit abstract so let me give an example. Again, I'm not familiar with UPP so have just taken some snippets from other examples to get something working, see below.

DialogBase shall be a reusable dialog component. Imagine it could have several controls and lots of methods. It could be an abstract base class as well.

The derived class Dialog_1 may have several additional controls and methods.

The points here are:
1) when designing the layout of derived dialogs I want to avoid code replication.
2) I might want to use polymorphism in derived dialogs as well.
3) Of course I can simply type this example with a text editor. However, the layout of the controls and their attributes, especially position and size, which may change for each derived dialog, can't be done easily with a text editor. That is where I need the layout designer.

So the question is: can the layout designer produce code that lets me specify the graphical layout in a simple way without compromising the code structure I want?

```
#include <CtrlLib/CtrlLib.h>

using namespace UPP;

class DialogBase : public TopWindow
{
 typedef DialogBase CLASSNAME;

protected:
 Button b_;
 void b_cb() { PromptOK("Button clicked."); }

public:
 DialogBase()
 {
  Title("DialogBase");
  SetRect(0, 0, 100, 100);
  b_.LeftPos(10, 30).TopPos(10, 30);
  b_.SetLabel("?");
  b_ <<= THISBACK(b_cb);
  Add(b_);
 }
};

class Dialog_1 : public DialogBase
{
protected:
 DocEdit e_;

public:
 Dialog_1()
 {
  Title("Dialog_1").Sizeable();
  SetRect(0, 0, 200, 200);
  b_.LeftPos(50, 100).TopPos(160, 30);
  b_.SetLabel("Dialog_1");
```

```
  e_.LeftPos(10, 180).TopPos(10, 140);
  Add(e_);
 }
};

GUI_APP_MAIN
{
 Dialog_1().Run();
}
```