Subject: more containers of widgets
Posted by exolon on Tue, 13 Feb 2007 01:48:01 GMT
View Forum Message <> Reply to Message

Hey guys,

I've been wondering if there are nicer ways of adding widgets to [Parent]Ctrls than specifying an exact position or using H/VSizePos and H/VCenterPos or whatever, or using the layout designer's springs.

The Java Swing (I think) API uses layout managers to determine where a new widget will be added in a container Panel, basically following the useful Strategy design pattern.

So you might have a Panel and decide that it uses the BorderLayout class, and then add items to it specifying north, south, northeast etc. Or you could set it to use some other (default) layout class which just spaces the widgets equally horizontally as you add them.
It's very convenient this way to have panels with subpanels, just like how you'd arrange a webpage with nested tables, each with its own alignment and configuration.

I can't seem to find any way to just Add() a widget to a ParentCtrl and have it 'figure out' a simple heuristic positioning for it, perhaps like this default layout manager in Swing, by spacing widgets horizontally if nothing is specified.
Has this kind of work been done already, or is it purposely avoided in UPP, or is it just too much work at the moment?

It would be so handy to just go...

```
ParentCtrl pane;
pane.SetLayout(Ctrl.LAYOUT_TOPDOWN);

ParentCtrl topSubPane;
ParentCtrl bottomSubPane;

topSubPane.Add(mickeyMouseDrawing);
bottomSubPane.Add(button1).Add(button2).Add(button3);
```

It would mean creating and reorganising forms/etc would be much faster and potentially more flexible.

The reason I've discounted the layout manager immediately is that it just isn't cut out for dealing with ParentCtrls in ParentCtrls - even if I designed the subpanes separately, widgets which were conceptually related would end up in different classes and they're not visible in the one place.

What do you think?

Subject: Re: more containers of widgets
Posted by darrs on Tue, 13 Feb 2007 19:17:32 GMT
View Forum Message <> Reply to Message

Hi there.

I was just thinking about the same thing last night. I'm currently making an application with the controls of one window spread over three ParentCtrl (panes).

Previously I've used the Fox GUI toolkit. One thing it was good at was automatic layout. It has many layout containers (e.g. pack vertically/horizontally, add to borders, center, distribute uniformly, line up in matrix). I've never used coordinates to place a control while using Fox. (However it has other problems, hence why I'm trying UPP now).

Anyway, I've been thinking about whether one would need to change UPP Ctrl to support different layout engines (possibly by using different selectable "layout" objects), or whether it is sufficient just having a temporary object to help during initial placement. For example you could use MyLayout.AddLeft(MyButton) and it would compute the position so that it is on the left side after all the previously added left controls.

The latter approach has the advance on not requiring changes to UPP itself, but is probably more limited in what it can achieve.   I thought there was going to be a problem with the TAB key focus ordering but while writing this message I just thought of a way around that.

I might give the latter approach a try and see how far I get. Unfortunately I'm not getting much time to spend on UPP at the moment so it could be a while.

I'd be interested to see what other people thing about this.

Cheers,
Darrin.

PS. I'm fairly new to UPP so everything I've just said could be complete rubbish.

Subject: Re: more containers of widgets
Posted by mirek on Wed, 14 Feb 2007 08:55:11 GMT
View Forum Message <> Reply to Message

Well, I think in theory, there is almost everything needed to get this work - because what you really need is in fact "GetStdSize".

In practice the only trouble is that GetStdSize is not always implemented to return value that is "correct enough"...

Anyway, I will add this to the list of things to think about.

Mirek

Subject: Re: more containers of widgets
Posted by exolon on Wed, 14 Feb 2007 21:05:20 GMT
View Forum Message <> Reply to Message

Darrs: What were the other problems with Fox? I considered this before I stumbled upon Ultimate++, but felt the look was too Windows-98ish, and I liked the more modern design of UPP (especially the "everything is owned by something" style).

Subject: Re: more containers of widgets
Posted by darrs on Thu, 15 Feb 2007 03:21:07 GMT
View Forum Message <> Reply to Message

Quote:What were the other problems with Fox?

It's been a while, but some of the things I still remember are:

a) Probably the biggest problem was the way the cut/paste (clipboard) was implemented. Often it would "lose" the text in the clipboard when cutting from one part of a Fox application and pasting into another part of the same Fox application. For example cutting text from one cell in a table and trying to paste into another cell. This was very annoying as it was very noticable by the customers.

[This wasn't a bug but a consequence of the way things were done. i.e. there was probably no way to fix apart from rewriting the clipboard sub-system].

b) Some problems with keyboard focus and using applications using the keyboard. For example in one dialog no matter what I did the focus would always start on one of the spin buttons inside a "EditIntSpin" widget rather than on the actual text part. Again very noticable to the customers.

c) No help system for the applications.

d) Inconsistency between widgets. For example many styling options (e.g. insert frame) only worked on some widgets and not others, some layout options would act differently for different types of widgets.

e) Some problems with keyboard accelerators. For example in a Wizard type dialog, the accelerators for the other "hidden" panes would collect the keystrokes. [Note: I think I noticed something similar in the tab dialog example in UPP. Will look into that when I get a chance.]

f) Lack of flexibility / control in some of the bigger widgets. For example in one application I had to copy the "table" and "file-open-dialog" implementations and do a lot of copy changes to get the necessary behavior. (Of course in some cases this may have been my lack of knowledge of how things were meant to be done.)

g) I kept running into things that I couldn't easily do. I have a number of little projects that are half done and then got stuck somewhere. (Once I have more experience with UPP I'll try these in UPP and see how I go).

h) The looks are definitely Win-98 based. Until recently that didn't bother me too much. But now with Win-XP so common it is something I care more about.

i) Missing various things I wanted.

Fox also has good points. When you get the layout options right (sometimes by long trial and error) the results are very good. Users could adjust fonts, window sizes, frames, etc. and things would re-adjust themselves properly. For example buttons would know how much space they really need (based on font and actual text) and would try to get that if possible.

Fox also has a mechanism for disabling "commands" automatically. You can provide functions to decide if a particular command ID should be enabled or not. These functions get called automatically by the GUI framework when becoming idle. If for example you indicate that this command is now disabled the GUI would automatically gray out all the associated buttons, menus, and widgets.

Fox has some things that UPP is currently missing (e.g. Drag and Drop) and visa-versa. On the whole I think UPP provides more things than Fox (although it is sometimes hard to find them in UPP). Note however that I haven't looked at Fox for a while (maybe a year).

Cheers,
Darrin.

PS. I should probably mention that my job is an embedded systems programmer. I only do small GUI stuff for work (e.g. configuration tools), and some small applications in my spare time. I don't write big PC applications.

Subject: Re: more containers of widgets
Posted by mrjt on Tue, 27 Mar 2007 10:29:03 GMT
View Forum Message <> Reply to Message

After reading this topic I realised that something along the lines of a simple pane layout manager would be useful in one of my projects, so I've knocked up a quick-and-dirty implementation and thought I'd share the code in case it's useful.

I've tried to get the implementation Upp-like rather than Java-like, but you might have to change the style for a more complicated manager like BorderLayout, especially since you can't overload Ctrl::Add(Ctrl &).

This is the class that does the actual layout:
```
template<class T>
class BoxLayout : public T
{
```

```
public:
   BoxLayout() {ishorz = false; border = 4; spacing = 4;}

   virtual void Layout();

   BoxLayout<T> &SetHorz(bool _ishorz = true)   {ishorz = _ishorz; return *this;}
   bool GetHorz() const                         {return ishorz;}

   BoxLayout<T> &SetBorder(int _border = 4)     {ASSERT(_border>=0); border = _border; return
*this;}
   int  GetBorder() const                       {return border;}

   BoxLayout<T> &SetSpacing(int _spacing = 4)   {ASSERT(_spacing>=0); spacing = _spacing;
return *this;}
   int  GetSpacing() const                      {return border;}
protected:
   int spacing;
   int border;
   bool ishorz;
};

template<class T>
void BoxLayout<T>::Layout()
{
   Size sz = T::GetSize();
   Ctrl::LogPos cp;
   int children = 0, total_space;
   Ctrl *c = T::GetFirstChild();

   while (c) {
      children++;
      c = c->GetNext();
   }
   if (!children) return;

   total_space = border*2+(children-1)*spacing;
   if (ishorz) {
      if (total_space > sz.cx) return;
      cp.x = Ctrl::Logc(LEFT, border, (sz.cx-total_space)/children);
      cp.y = Ctrl::Logc(SIZE, border, border);
   }
   else {
      if (total_space > sz.cy) return;
      cp.x = Ctrl::Logc(SIZE, border, border);
      cp.y = Ctrl::Logc(TOP, border, (sz.cy-total_space)/children);
   }

   c = T::GetFirstChild();
```

```
    while (c) {
        c->SetPos(cp);
        if (ishorz)
            cp.x.SetA(cp.x.GetA()+cp.x.GetB()+spacing);
        else
            cp.y.SetA(cp.y.GetA()+cp.y.GetB()+spacing);
        c = c->GetNext();
    }
}

typedef BoxLayout<LabelBox> LabelBoxPane;
typedef BoxLayout<StaticRect> StaticBoxPane;
```

And this is a test window:

```
class AWindow : public TopWindow {
public:

 typedef AWindow CLASSNAME;
 LabelBoxPane top_pane;
 StaticBoxPane sub_pane;
 Button btn[5];

 AWindow()
 {
  SetRect(Size(400, 400));
  CenterScreen();
  Title("Layout Test");
  Sizeable();

  // Config buttons and add to top pane
  for (int i = 0; i < 3; i++) {
   btn[i].SetLabel("Button " + AsString(i));
   btn[i] <<= THISBACK(OnButton1);
   top_pane.Add(btn[i]);
  }
  // Config other buttons and add to sub pane
  for (int i = 3; i < 5; i++) {
   btn[i].SetLabel("Button " + AsString(i));
   btn[i] <<= THISBACK(OnButton2);
   sub_pane.Add(btn[i]);
  }
  // Config sub pane
  sub_pane.SetHorz(true).SetBorder(0);

  // Config top pane and add sub pane (LabelBox ignores mouse by default)
  top_pane.SetLabel("LabelBox with BoxLayout").IgnoreMouse(false);
  top_pane.SetBorder(12).Add(sub_pane);
  // Add top pane
  Add(top_pane.VSizePosZ(10, 10).HSizePosZ(10, 10));
 }

 void OnButton1() {
  // Flips orientation of top pane
  top_pane.SetHorz(!top_pane.GetHorz());
  top_pane.Layout();
 }

 void OnButton2() {
  // Flips orientation of sub pane
  sub_pane.SetHorz(!sub_pane.GetHorz());
  sub_pane.Layout();
 }
};
```

And for completeness some ESC code to expose two possible instanciations in the Layout Manager:

```
ctrl LabelBoxPane {
 group "Layout";

 >LabelBox;
 bool SetHorz = true;
 int SetBorder = 4;
 int SetSpacing = 4;
};

ctrl StaticBoxPane {
 group "Layout";

 >Base;
 bool SetHorz = true;
 int SetBorder = 4;
 int SetSpacing = 4;
};
```

It seems to work quite well, and although you wouldn't want to use containers like this all the time it's always nice to have more tools

There are some problems though:
1) The border and spacing around/between controls ignores ZoomRatio, which I thought made sense. However, when used with a LabelBox or similar changing the StdFont will make hard-coded border values wrong (I think?).
2) This style of layout will not work well with the layout manager. I can see no way of adding controls to the panes via the Layout Manager, so this will always have to be done from code.
3) I'm not sure how to make this work with frames, since using GetView inside Ctrl::Layout causes

a failed assertion. Is this possible?

Subject: Re: more containers of widgets
Posted by exolon on Wed, 28 Mar 2007 19:28:02 GMT
View Forum Message <> Reply to Message

Wow! That looks impressive so far, Mrjt! Although I can't test it (or breathe, hardly...) before getting some work done on my project at college, I'll try to give it a go at the weekend.

Nice to see some progress being made on this front, for sure.