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?