Subject: Re: ArrayCtrl with labels / layouts inside ?
Posted by mrjt on Thu, 29 Nov 2007 12:57:35 GMT
View Forum Message <> Reply to Message

Quote:Well, that pointer version SetCtrl should not really be public. Do not use itAre you sure a about this? The documentation says:
Sets an external control to use as the editor for a single array cell. This function transfers the ownership to the control (from now on, it is stored within the array and destroyed as necessary - upon destruction of the array, deletion of the relevant row or another call to SetCtrl).And this makes sense because it is very difficult to use SetCtrl and have to manage ctrls externally.

Also, since I have some time on my hands I'm going to wade in with my opinion whether you like it or not . I'm not sure if SetCtrl is the correct approach to use in this case. The code above should work with the addition of:
CtrlLayout(*test); array_waiters.SetLineCy(0, w.GetMinSize().cy);but there are two (and probably more) ways of doing this that may be more suitable:

1- Avoids SetCtrl completely

```
struct CallerInfo {
 String callername;
 String country;
 String telephone;
};

class MainWindow : public TopWindow
{
public:
 typedef MainWindow CLASSNAME;
 ArrayCtrl array;

 struct Waiter : public WithWaiterLayout<ParentCtrl> {
  Value data;

  Waiter() { CtrlLayout(*this); }
  virtual void SetData(const Value &v) {
   if (IsTypeRaw<CallerInfo>(v)) {
    const CallerInfo &c = ValueTo<CallerInfo>(v);
    callername.SetLabel(c.callername);
    country.SetLabel(c.country);
    telephone.SetLabel(c.telephone);
    data = v;
   }
  }
  virtual Value GetData() const { return data; }
 };

 MainWindow() {
  Sizeable();
```

```
  array.AddColumn("Data 1");
  array.AddColumn("Data 2");
  array.AddColumn("Waiter").Ctrls<Waiter>();
  Add(array.SizePos());

  CallerInfo c;
  c.callername = "Joe Sixpack";
  c.country = "USA";
  c.telephone = "666-666666";

  array.Add("Cell (0,0)", "Cell (1,0)", RawToValue<CallerInfo>(c));
  array.Add("Cell (0,1)", "Cell (1,1)", RawToValue<CallerInfo>(c));
  array.Add("Cell (0,2)", "Cell (1,2)", RawToValue<CallerInfo>(c));
 }
};
```

2- Uses Display. Unless you have a really good reason for using controls (ie. some sort of user interaction required) this should be the default method:

```
struct CallerInfo {
 String callername;
 String country;
 String telephone;
};

class MainWindow : public TopWindow
{
public:
 typedef MainWindow CLASSNAME;
 ArrayCtrl array;

 struct CallerDisplay : public Display
 {
  void Paint(Draw& w, const Rect& r, const Value& q, Color ink, Color paper, dword style) const {
   PaintBackground(w, r, q, ink, paper, style);
   if (IsTypeRaw<CallerInfo>(q)) {
    const CallerInfo &c = ValueTo<CallerInfo>(q);
    int ccy = r.GetWidth() / 3;
    int h = r.Height();
    int x = r.right - ccy;

    w.Begin();
    w.DrawText(x, r.top+1, c.telephone, StdFont(), ink);
    w.ExcludeClip(x, r.top, ccy, h);
    x -= ccy;
    w.DrawText(x, r.top+1, c.country, StdFont(), ink);
    w.ExcludeClip(x, r.top, ccy, h);
    x -= ccy;
    w.DrawText(x, r.top+1, c.callername, StdFont(), ink);
```

```
    w.End();
   }
  }
 };

 MainWindow() {
  Sizeable();

  array.AddColumn("Data 1");
  array.AddColumn("Data 2");
  array.AddColumn("Waiter").SetDisplay(Single<CallerDisplay>());
  Add(array.SizePos());

  CallerInfo c;
  c.callername = "Joe Sixpack";
  c.country = "USA";
  c.telephone = "666-666666";

  array.Add("Cell (0,0)", "Cell (1,0)", RawToValue<CallerInfo>(c));
  array.Add("Cell (0,1)", "Cell (1,1)", RawToValue<CallerInfo>(c));
  array.Add("Cell (0,2)", "Cell (1,2)", RawToValue<CallerInfo>(c));
 }
```
}; you could further improve the Paint function with the '...' when string are cut short and give a much better appearance than using Labels. See StdDisplayClass::Paint0 for how to do this.

Hope that helps.
James
