Subject: Re: Problem with ColumnList (with example)
Posted by James Thomas on Fri, 13 Oct 2006 11:00:28 GMT
View Forum Message <> Reply to Message

I've looked at the source more closely and I believe I've fixed the selection and highlighting problems.

The first change is to LeftDown. There was no way of selecting an item without using the Shift or Ctrl keys and some of the multi-selection behaviour was being incorrectly applied when in single selection mode. My version is below.

```
void ColumnList::LeftDown(Point p, dword flags) {
 int i = GetDragColumn(p.x);
 if(i >= 0) {
  ci = i;
  dx = p.x - GetColumnCx(0) * (i + 1);
  mpos = p.x;
  SetCapture();
  Refresh(mpos - dx, 0, 1, GetSize().cy);
 }
 else {
  int anchor = cursor;
  SetWantFocus();
  PointDown(p);
  p.y %= cy;
  p.x %= GetColumnCx(0);
  if(cursor >= 0) {
   // JT 13/10/06 Added multi check to if statements
   if(multi && flags & K_SHIFT && anchor >= 0) {
    ShiftSelect(anchor, cursor);
    WhenLeftClickPos(p);
    return;
   }
   else
   if(multi && flags & K_CTRL) {
    if(anchor >= 0 && !IsSelection())
     SelectOne(anchor, true);
    SelectOne(cursor, !IsSelected(cursor));
    WhenLeftClickPos(p);
    return;
   }
   else if (multi) ClearSelection(); // JT 13/10/06
   SelectOne(cursor, true); // JT 13/10/06 Added to fix selection
  }
  else
   ClearSelection();
  WhenLeftClickPos(p);
 }
}
```

A change was also required to the GetItemStyle function that decides what colour an item is drawn in. The original code was:

```
 if(m.sel) {
  style |= Display::SELECT;
  paper = SColorShadow;
  if(HasFocus()) {
   paper = SColorPaper;
   ink = SColorText;
  }
 }
```

Which makes no sense. Why would the highlighting behaviour be different depending on focus? And you certainly don't want to not highlight items when the control has focus. My version:

```
 if(m.sel) {
  style |= Display::SELECT;
  paper = SColorShadow;
 }
```

I would personally prefer to use SColorHighlight, but SColorShadow seems consistent with TheIDE. Perhaps the intention of the original code was to use shadow if the control doesn't have focus and SColorHighlight if it does?

In the same function there is also the code:

```
 if(i == cursor) {
  style = isselection ? Display::CURSOR : Display::CURSOR|Display::SELECT;
  paper = isselection ? Blend(SColorHighlight, SColorFace) : SColorFace;
  if(HasFocus()) {
   style |= Display::FOCUS;
   paper = isselection ? Blend(SColorHighlight, SColorPaper) : SColorHighlight;
   ink = SColorPaper;
  }
```

Which I cannot understand the purpose of. If you would like to explain I would be interested in the reason for this.

Overall this control seems very strangely implemented and works very differently from the ColumnList controls in TheIDE (I haven't checked the IDE source to see how they are implemented, but they definitely aren't using this code!).

To make the control consistent with TheIDE I have made some other changes (in the attached files), and it now works in a much more sensible way. I have also added a GetSelectedItem member to reduce the code required to access the first selected item when using it (most useful in single selection mode). I am also unhappy with the amount of list scanning/iteration required by this control (happens whenever an item is selected or the selected item is retrieved) as it would be horribly inefficient with very large lists, but fixing this is difficult so I haven't done it yet.

All of my changes have been made to the latest dev source (Dev-1) and are maked by // JT 13/10/06.

I hope this is useful

In addition, here is a modified version of the test code I gave in my first post that also illustrates the us of the new GetSelectedItem function:

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

class AWindow : public TopWindow {
public:
 typedef AWindow CLASSNAME;

 Option _optMulti;
 ColumnList _List;
 Label _Label1, _Label2;
 EditInt _intSelCount;
 EditString _txtItem;

 AWindow()
 {
  Ctrl::LogPos p = GetPos();
  p.x.SetB(228);
  p.y.SetB(380);
  SetPos(p);

  _List.LeftPosZ(4, 220).TopPosZ(32, 292);
  _intSelCount.LeftPosZ(176, 48).TopPosZ(328, 19);
  _Label1.SetLabel("Number of items selected:").LeftPosZ(48, 124).TopPosZ(328, 19);
  _Label2.SetLabel("Selected item:").LeftPosZ(100, 72).TopPosZ(352, 19);
  _optMulti.SetLabel("Multi-Select").LeftPosZ(4, 76).TopPosZ(8, 15);
  _txtItem.LeftPosZ(176, 48).TopPosZ(352, 19);
  Add(_List);
  Add(_intSelCount);
  Add(_Label1);
  Add(_Label2);
  Add(_optMulti);
  Add(_txtItem);

  _List.Columns(1);
  _List.Multi(false);
  _List.WhenSelection = THISBACK(Selection);

  String s = "Spam ";
  for (int i = 0; i < 20; i++) {
   _List.Add(s + AsString(i));
  }

  _intSelCount.SetData(0);
  _intSelCount.SetEditable(false);
```

```
  _txtItem.SetText("None");
  _txtItem.SetEditable(false);

  _optMulti <<= THISBACK(MultiChange);
 }

 void Selection()
 {
  int cnt = 0, i;

  for (i = 0; i < _List.GetCount(); i++) {
   if (_List.IsSelected(i))
    cnt++;
  }
  _intSelCount.SetData(cnt);


                // Test the new member function
  i = _List.GetSelectedItem();
  if (i < 0)
   _txtItem.SetText("None");
  else
   _txtItem.SetText((String)_List.Get(i));
 }

 void MultiChange()
 {
  _List.Multi(_optMulti.Get());
 }
};

GUI_APP_MAIN
{
 AWindow w;

 w.Run();
}
```

File Attachments

1) ColumnList.cpp, downloaded 3211 times
2) ColumnList.h, downloaded 3146 times