Subject: Problem with ColumnList (with example) [BUG/FIXED]
Posted by James Thomas on Thu, 12 Oct 2006 17:20:20 GMT
View Forum Message <> Reply to Message

I hope I'm not asking an obvious question (again ), but I'm having serious problems with ColumnList. I can't get it to work properly at all.

In the example below I fill a list box with entries and after every selection I count the number of selected entries and display it in an EditInt ctrl. However, the behaviour of the list is very strange.

When you click on an item the selection callback is triggered and the item is highlighted, but when I check the items it appears that none has actually been selected. When you hold down the Ctrl or Shift key it correctly registers the selection but the row is not highlighted (unless you use Shift in which case it only highlight the anchor item). This is obviously completely wrong and happens both in single and multi-select modes.

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

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

 Option _optMulti;
 ColumnList _List;
 Label _Label;
 EditInt _intSelCount;

 AWindow()
 {
  Ctrl::LogPos p = GetPos();
  p.x.SetB(228);
  p.y.SetB(356);
  SetPos(p);

  _List.LeftPosZ(4, 220).TopPosZ(32, 292);
  _intSelCount.LeftPosZ(176, 48).TopPosZ(328, 19);
  _Label.SetLabel("Number of items selected:").LeftPosZ(48, 124).TopPosZ(328, 19);
  _optMulti.SetLabel("Multi-Select").LeftPosZ(4, 76).TopPosZ(8, 15);
  Add(_List);
  Add(_intSelCount);
  Add(_Label);
  Add(_optMulti);


  _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);

  _optMulti <<= THISBACK(MultiChange);
 }

 void Selection()
 {
  int cnt = 0;

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

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

GUI_APP_MAIN
{
 AWindow w;

 w.Run();
}
```

I suspect I've done something wrong, since this seems like a rather obvious problem, but then I looked at the code in ColumnList.cpp:

```
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) {
   if(flags & K_SHIFT && anchor >= 0) {
    ShiftSelect(anchor, cursor);
    WhenLeftClickPos(p);
    return;
   }
   else
   if(flags & K_CTRL) {
    if(anchor >= 0 && !IsSelection())
     SelectOne(anchor, true);
    SelectOne(cursor, !IsSelected(cursor));
    WhenLeftClickPos(p);
    return;
   }
  }
  ClearSelection();
  WhenLeftClickPos(p);
 }
}
```
And it looks like SelectOne is never called unless Ctrl or Shift are held down. It is still a mystery to me how the row can be highlighted though.

What's going on?

(tested on Dev-2, but the ColumnList source looks the same in 605)

---

## Subject: Re: Problem with ColumnList (with example)
Posted by James Thomas on Thu, 12 Oct 2006 17:24:31 GMT
View Forum Message <> Reply to Message

Redundant. See below.

---

## 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 3213 times
2) ColumnList.h, downloaded 3151 times

---

Subject: Re: Problem with ColumnList (with example)
Posted by mirek on Fri, 13 Oct 2006 14:31:06 GMT
View Forum Message <> Reply to Message

Thanks. Actually, the real problem was wrong colors for selected item in GetItemStyle. This bug escaped our attention because ColumnList so far is used just for filelists, where it has different Display.

As for not selecting the item without Shift or Ctrl, that is correct - cursor is not considered to be the selection.

ColumnList returns IsSelection when there are selected items. When there are not, IsCursor still can be true - in that case you have to use GetCursor to find which item has cursor.

My fixed GetItemStyle version:

```
void ColumnList::GetItemStyle(int i, Color& ink, Color& paper, dword& style)
{
 ink = SColorText;
 paper = SColorPaper;
 const Item& m = item[i];
 style = 0;
 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;
  }
 }
 if(m.sel) {
  style |= Display::SELECT;
  paper = SColorShadow;
  if(HasFocus())
   style |= Display::FOCUS;
 }
}
```

Note: Instead that complicated game with LogPos at the benning of AWindow, SetRect(0, 0, 228, 356); would be a simpler variant.

---

## Subject: Re: Problem with ColumnList (with example)
Posted by James Thomas on Mon, 16 Oct 2006 11:32:22 GMT
View Forum Message <> Reply to Message

Thanks for the tip. And I had a feeling you would have an easier way of fixing the bug.

My main problem was obviously not understanding having to call IsCursor and GetCursor as well as IsSelected. In addition you have to use the WhenEnterItem callback instead of WhenSelection to pick up on the change. I do not like this interface, IMO it is not obvious to new users (ie. me ) and needs extra code for functionality that could be a single line (like retrieving the first selected item). I appreciate this gives more flexibility but some additional member functions with obvious names would make this control easier to use.

I also still think that there needs to be an 'if (multi)' check in the LeftDown handling. When in single selection mode using the Ctrl key causes the item to be highlighted in grey, and Shift also causes a second item to be highlighted. As this behaviour makes no sense when you can't select more than one item it should be removed. Simply changing:

```
  if (cursor >= 0) {
```
to
```
  if (multi && cursor >= 0) {
```
Fixes this.

Just my 2c really, since I can make it work however I like for my own projects. Thanks again.

---

Subject: Re: Problem with ColumnList (with example)
Posted by mirek on Mon, 16 Oct 2006 11:49:34 GMT
View Forum Message <> Reply to Message

James Thomas wrote on Mon, 16 October 2006 07:32
I do not like this interface, IMO it is not obvious to new users (ie. me ) and needs extra code for functionality that could be a single line (like retrieving the first selected item). I appreciate this gives more flexibility but some additional member functions with obvious names would make this control easier to use.

Actually, I am thinking about improving since my reply in this thread....

Maybe it could be as simple as returning true for IsSelected when items has cursor (return item.sel || iteminde == cursor).

```
  if (multi && cursor >= 0) {
```

You are right. (Fixed).

Mirek

---