Subject: Best way to implement a two-way LineEdit
Posted by eeyore on Wed, 05 Mar 2008 21:47:02 GMT
View Forum Message <> Reply to Message

My apologies gentlemen, but I just started using this program today, so bare with me.

I'm writing a program that had a network interface and I want to use a LineEdit (?) to display command received, as well as a way for the user to enter commands to be sent. Basically, I'm trying to put a console in a GUI, for lack of a better of way of explaining it. I've got everything done except figuring out this LineEdit and how to do two things: 1) How do I detect when the user hits the enter key, and 2) is there a way to insert lines and have them be different colors?

In regards to the second item, I tried bypassing the call to Refresh in SetColor() and only refreshing the Rect that contains the line I just entered, but that didnt' work. Apparently, Refresh is being called somewhere else and it's changing the color of the entire LineEdit instead of just the Rect containing my newly inserted line.

So, any ideas would be greatly appreciated. Thanks in advance.

Subject: Re: Best way to implement a two-way LineEdit
Posted by mrjt on Thu, 06 Mar 2008 09:53:39 GMT
View Forum Message <> Reply to Message

Setting the line color is easy enough, you just need a sub-class that overloads LineEdit::HighlightLine. I think you could even set the color (or change font!) for every character if necessary.

I'm not sure about a console-like LineEdit though. Personally I'd use a separate EditString for the text entry and keep the LineEdit as read-only.

This example colours alternate lines red/blue:

```
#include <CtrlLib/CtrlLib.h>
using namespace Upp;

class Console : public LineEdit {
   virtual void  HighlightLine(int line, Vector<Highlight>& h, int pos) {
      for (int i = 0; i < h.GetCount(); i++)
         h[i].ink = (line % 2) ? SBlue : SRed;
   }
};

class CtrlLibTest : public TopWindow {
   Console line;
   EditString cmd;

public:
   typedef CtrlLibTest CLASSNAME;
```

```
    CtrlLibTest()
    {
        SetRect(0, 0, 308, 344);
        CenterScreen();

        Add(line.NoCutLine().WantFocus(false).HSizePos().VSizePosZ(0, 23));
        Add(cmd.HSizePos().BottomPosZ(0, 24));
    }

    virtual bool Key(dword key, int count)
    {
        if (key == K_RETURN && cmd.HasFocus()) {
            if (line.GetLineCount() >= 199)
                line.Remove(0, line.GetLineLength(0)+1);
            line.Insert(line.GetLength(), (WString)~cmd + '\n');
            line.SetCursor(line.GetLength());
            cmd.Clear();
            return true;
        }
        return false;
    }
};

GUI_APP_MAIN
{
    CtrlLibTest().Run();
}
```

but if you really want to intercept the return key in the LineEdit you just add a Key overload to Console and remember to call LineEdit::Key if key != K_RETURN.

Hope that helps.
James

Subject: Re: Best way to implement a two-way LineEdit
Posted by eeyore on Thu, 06 Mar 2008 14:30:35 GMT
View Forum Message <> Reply to Message

Thanks for the tips. I had thought about using a separate EditString, and it looks like that might be the best way to go. I'll give the LineHighlight a try and see what I can do. Thanks for the code!
