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
