Subject: need some expert advice to extend the LineEdit
Posted by navi on Sun, 02 Dec 2012 13:54:17 GMT
View Forum Message <> Reply to Message

I need some expert advice/pointer. I am trying to extend the LineEdit Class to add an option for displaying line number on the left of the each line possible with PAPER_READONLY color background.

So far I have I have extended my class as follows:

```
class LineEditExtended : public LineEdit{
 public:
  virtual void   Paint(Draw& w);

  void   Paint0(Draw& w);

  Rect DropCaret();
...}
```

As I do not wish the break the original LineEdit Ctrl, I am overriding the above function then copy/pasting the content from LineEdit.cpp and modifying it. I have modified the above function

regular backgrounds. My plan is to create some empty space on the left of the LineEdit ctrl vie making it start drawing from x-offset. Once that is achieved then I plan to add my own codes to draw the line number on that empty space as read-only non-selectable text.

So far I have manage to create some empty space on the left but have broken the 'Caret' placement on the text. I am still trying to fix the issue but finding quite hard to understand the mechanism of the TextCtrl/LineEdit Ctrl.

I have thought of doing it in other ways like making a compound ctrl with either two LineEdit or a "Label" ctrl and a LineEdit ctrl. Where one of the LineEdit or Label ctrl will be used to print the line numbers. In this way, I am not quite sure how I might be able to deal with scrolling and new lines.

I have tried investigating the CodeEditor Ctrl to see how it implemented line number. I concluded it does it through the friend class EditorBar. I am still trying to understand how.

Attached with this post is a test project using the LineEditExtended Ctrl.

File Attachments
1) testGUItmp1 03-12-2012 #0037.rar, downloaded 344 times

Subject: Re: need some expert advice to extend the LineEdit
Posted by navi on Sun, 02 Dec 2012 14:00:41 GMT
View Forum Message <> Reply to Message

File Attachments

1) screen-caps.png, downloaded 660 times

Subject: Re: need some expert advice to extend the LineEdit
Posted by dolik.rce on Sun, 02 Dec 2012 14:35:06 GMT
View Forum Message <> Reply to Message

Hi navi,

Do you have any reason to not use CodeEditor (other than curiosity, perhaps)? It can be set up to behave the same way as LineEdit, just with linenumbers... Also you mentioned some other features that might be useful for you.

Honza

Subject: Re: need some expert advice to extend the LineEdit
Posted by navi on Sun, 02 Dec 2012 15:04:56 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 02 December 2012 15:35Hi navi,

Do you have any reason to not use CodeEditor (other than curiosity, perhaps)? It can be set up to behave the same way as LineEdit, just with linenumbers... Also you mentioned some other features that might be useful for you.

Honza
Hello Honza,
I am not using CodeEditor Ctrl, because I believe it is much more complex then LineEdit Ctrl. And quite a bit harder to understand. The only example of CodeEditor Ctrl I could find is theIDE. No examples or documentation or sample program to see how it might be used or customized. Another point is, since CodeEditor is derived from LineEdit. it is essential that I understand LineEdit and TextCtrl inner workings properly. Since I just started to learn how line LineEdit is doing what it is doing, trying to extend CodeEditor will just add another layer of codes and mechanism that I do not yet understand.

For instance I notice that CodeEditor does not include a Print function. Obviously it is using the Print of LineEdit to get the job done. so no point working with CodeEditor yet as I have to understand the LineEdit first.

regards
Navi

Subject: Re: need some expert advice to extend the LineEdit
Posted by dolik.rce on Sun, 02 Dec 2012 16:38:20 GMT
View Forum Message <> Reply to Message

OK, so the reason is curiosity/self-education I am OK with that...

The EditorBar as you correctly found is what makes the line numbers work. In theory, it is nothing more than a FrameCtrl in the CodeEditors frame that paints the line numbers and other info. The "friend class" declaration is there only to allow direct access to the private members and thus make the implementation a bit simpler.

The EditorBar in CodeEditor does actually much more then line numbering. It takes alo care of annotations, breakpoints, ifdef tracking etc... To show it to you in simpler form, here is a commented example of it's very stupid cousin

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

using namespace Upp;

class MyEditorBar : public FrameLeft<Ctrl> {
public:
 virtual void Paint(Draw& w);
 LineEdit* editor; // we need to know who we work for, so we can
                   // read some necessary info (font, line positions etc.)
public:
 void SetEditor(LineEdit *e)          { editor = e; }

 MyEditorBar() {
  Width(27); // set some default width
             // note that 27 will not work with more than 99 lines ;)
 };
 virtual ~MyEditorBar() {};
};

void MyEditorBar::Paint(Draw& w)
{
 Size sz = GetSize();
 w.DrawRect(0, 0, sz.cx, sz.cy, SColorLtFace); // paint background

 if(!editor)
  return; // can't work without editor

 // read some info from the associated editor
 int fy = editor->GetFontSize().cy;
 Font f = editor->GetFont();
 int line = editor->GetScrollPos().y;
 int linecount = editor->GetLineCount();

 // iterate over all the visible lines
 int y = 0;
 while(y < sz.cy) {
  if(line == linecount)
   break; // we reached last line

  if(editor->GetCaret().top == y) {
   // current line should be highlighted
   w.DrawRect(0, y, sz.cx, fy, Blend(SColorHighlight(), SColorLtFace(), 200));
   w.DrawText(2, y + 2, IntStr(line+1), f, SColorHighlightText());
  } else {
   //other lines should be rendered normally
   w.DrawRect(0, y, sz.cx, fy, SColorLtFace());
   w.DrawText(2, y + 2, IntStr(line+1), f);
  }

  y += fy;
  line++;
 }
}

class LineEditExtended : public LineEdit {
 typedef LineEditExtended CLASSNAME;
 MyEditorBar bar;
public:
 LineEditExtended(){
  AddFrame(bar); // add the frame
  bar.SetEditor(this);
 }
 virtual bool Key(dword key, int count){
  // refresh every time user presses something (he might add/remove lines or scroll)
  bar.Refresh();
  // and forward the event to parent class
  return LineEdit::Key(key, count);
 }
 virtual Image MouseEvent(int event, Point p, int zdelta, dword keyflags){
  // refresh every time user uses mouse (he might cut/paste lines or scroll)
  bar.Refresh();
  // and forward the event to parent class
  return LineEdit::MouseEvent(event, p, zdelta, keyflags);
 }
};

class MainWindowDlg : public TopWindow {
 typedef MainWindowDlg CLASSNAME;
public:
 MainWindowDlg(){
  Add(edit.SizePos());
  edit <<= "Some\nsample\ntext...\n";
 }
private:
 LineEditExtended edit;
};

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

It is not complex at all, if you remove all the clutter around Basically just a Paint function and some simple overrides in the LineEditExtended class to keep things synced.

Also, just for your reference, I attach a very simple tool I wrote for myself some time ago. It uses CodeEditor to issue commands to SqlLite database. It is rather stupid, but you can see that using CodeEditor can be very simple To get idea of other capabilities of any class, it is always good to read the public part of its interface, the methods in U++ are usually quite self-explaining.

Honza

File Attachments

1) main.cpp, downloaded 307 times

Subject: Re: need some expert advice to extend the LineEdit
Posted by navi on Sun, 02 Dec 2012 18:16:36 GMT
View Forum Message <> Reply to Message

Quote:OK, so the reason is curiosity/self-education I am OK with that...
Thanks for accepting the reason to help me. big thanks for spending your time helping me.

Quote:The EditorBar as you correctly found is what makes the line numbers work. In theory, it is nothing more than a FrameCtrl in the CodeEditors frame that paints the line numbers and other info. The "friend class" declaration is there only to allow direct access to the private members and thus make the implementation a bit simpler.

The EditorBar in CodeEditor does actually much more then line numbering. It takes alo care of annotations, breakpoints, ifdef tracking etc... To show it to you in simpler form, here is a commented example of it's very stupid cousin

It is not complex at all, if you remove all the clutter around Wink Basically just a Paint function and some simple overrides in the LineEditExtended class to keep things synced.

again, big thanks and I really appreciate the you dumb down the EditorBar for me. Amazing! It would have taken me hours and hours or perhaps days to go through EditorBar's code to understand what you manage to explain in few lines of comment!

Quote:Also, just for your reference, I attach a very simple tool I wrote for myself some time ago. It uses CodeEditor to issue commands to SqlLite database. It is rather stupid, but you can see that using CodeEditor can be very simple

looking at your example program, indeed it does look codeeditor is quite easy to use. I will read the public interface thoroughly to learn how to customize it to my needs.

Quote:To get idea of other capabilities of any class, it is always good to read the public part of its interface, the methods in U++ are usually quite self-explaining.
I agree. Most of the times they are. public interfaces of U++ class are almost hundreds of times easier then trying to make seance of the inner working of the classes themselves. though rarely but unfortunately at times the public interface aren't enough to extend the classes if need be. since codes and more impotently variables are un-commented make some times quite hard to understand them.
but thankfully for those times, great members of U++ community like yourself are around and very very helpful.

regards
navi

p.s. A question lingering in my mind, in your dumb down version of Editorbar, it is not declared as friend in LineEditExtended class unlike the CodeEditor. was that because in this version there was no need to access any private members?

Subject: Re: need some expert advice to extend the LineEdit
Posted by dolik.rce on Sun, 02 Dec 2012 18:56:53 GMT
View Forum Message <> Reply to Message

navi wrote on Sun, 02 December 2012 19:16Quote:OK, so the reason is curiosity/self-education I am OK with that...
Thanks for accepting the reason to help me. big thanks for spending your time helping me.I'm helping because I believe that curious users like you might one day become a developers of U++

navi wrote on Sun, 02 December 2012 19:16Quote:To get idea of other capabilities of any class, it is always good to read the public part of its interface, the methods in U++ are usually quite self-explaining.
I agree. Most of the times they are. public interfaces of U++ class are almost hundreds of times easier then trying to make seance of the inner working of the classes themselves. though rarely but unfortunately at times the public interface aren't enough to extend the classes if need be. since codes and more impotently variables are un-commented make some times quite hard to

understand them.Yes, the internals are sometimes quite hard to read. I personally find it easiest to pick one distinct feature and search through all the related code to figure out where it comes to play and what code is necessary to make it work, ussualy starting at the method that turns the feature on/off, working my way inside the class using search to see where the variables are used and what function calls are made. This might not of course fit everyone, but it usually works for me.

navi wrote on Sun, 02 December 2012 19:16p.s. A question lingering in my mind, in your dumb down version of Editorbar, it is not declared as friend in LineEditExtended class unlike the CodeEditor. was that because in this version there was no need to access any private members?There was no need. The only place where it accesses the LineEditor is the few calls in Paint, and they are all public functions, so there was no reason to make it a friend class.

Honza