

控件开发:讲述如何开发一个控件,很有价值

(七)

疯狂代码 <http://CrazyCoder.cn/> <http://CrazyCoder.cn/Delphi/Article11915.html>

SUCCESS - (Nearly...)

I think you'll agree we are pretty close. There is just a little bit of flicker. This flicker is the SelStart jumping the Cursor position around the text. We need to hide this. This "\Cursor\" is also known as a Caret. Looking through Win32.Hlp again we find the lovely, and appropriately named, HideCaret function.

Lets try this then: everytime we change the value of MyRe.SelStart lets call HideCaret(MyRe.Handle) immediately before.

I'll be kind - that doesn't work. I tried 2 x HideCaret(MyRe.Handle), and it still didn't work. Neither did three, four or 25x. So close - but yet - so far. I think its time for another Delphi Rule:

DELPHI RULE #5 - If you bother to get your way through the atrocious index of the Win32.HLP file to find what you are looking for - make sure you really read what you found properly!

The key was the last paragraph in the description of not HideCaret but ShowCaret (which I had also read as I thought we were going to need it, especially to reverse my 25x HideCaret). You also need another Delphi Rule to understand it:

The caret is a shared resource; there is _disibledevent=>DELPHI RULE #6 - Everything (basically) is a Window

You see the RichEdit is a windows control and is also.. in a weird sense.. a window. It has a Handle, which is why HideCaret would accept it. So re-reading the last line again we get:

The caret is a shared resource; there is _disibledevent=>ASH Version 0.9b

```
procedure TForm1.RichEdit1Change(Sender: TObject); var
```

```
SaveOnChangeIn: TNotifyEvent;
```

```
WasSelStart,WasRow,Row,BeginSelStart,EndSelStart: Integer;
```

```
MyRe : TRichEdit;
```

```
MyPBuff: .gif' />[0..255] of char;
```

```
MyTokenStr;
```

```
MyTokenState:TTokenState;
```

```
MyRun:PChar;
```

```
MySelStart: Integer;
```

```
begin
```

```
MyRe := TRichEdit(Sender);
```

```
SaveOnChangeIn := MyRe.OnChange;
```

```
MyRe.OnChange := nil;
```

```
MyRe.Enabled := False;
```

```
WasSelStart := MyRE.SelStart;
```

```
WasRow := MyRE.Perform(EM_LINEFROMCHAR, MyRE.SelStart, 0);
```

```
Row := WasRow;
```

```
BeginSelStart := MyRe.Perform(EM_LINEINDEX, Row, 0);
```

```
EndSelStart := BeginSelStart + Length(MyRE.Lines.Strings[Row]);
```

```
StrPCopy(MyPBuff,MyRE.Lines.Strings[Row]);
```

```
MYPBuff[Length(MyRE.Lines.Strings[Row])] := #0;
```

```
MySelStart := BeginSelStart;
```

```
MyRun := MyPBuff;
```

```

while(MyRun^ <> #0) do
begin

MyRun := PasCon.GetToken(MyRun,MyTokenState,MyTokenStr);

MyRE.SelStart := MySelStart;
MyRE.SelLength := Length(MyTokenStr);

If MyRE.SelAttributes.Name <> PasCon.FParseFont[MyTokenState].Name then
MyRE.SelAttributes.Name := PasCon.FParseFont[MyTokenState].Name;

If MyRE.SelAttributes.Color <> PasCon.FParseFont[MyTokenState].Color then
MyRE.SelAttributes.Color := PasCon.FParseFont[MyTokenState].Color;

  MyRE.SelAttributes.Style <> PasCon.FParseFont[MyTokenState].Style then MyRE.SelAttributes.Style
:= PasCon.FParseFont[MyTokenState].Style;

MySelStart := MySelStart + Length(MyTokenStr);

end;

MyRE.SelStart := WasSelStart;
MyRE.SelLength := 0;
MyRe.OnChange := SaveOnChangeIn;
MyRe.Enabled := True;

MyRe.SetFocus;

end;

```

Towards - ASH Version 1.0b

Couple of problems with the last version you try it out for size: Its slightly inefficient in that everytime

SelAttributes is changed it forces a reparse of the same token in the control. We should instead use some variable (e.g. var DoFormat:Boolean) to decide we need to reformat, and then check the value of DoFormat at the end of this checking, and do it all then by a simple SelAttribute.Assign(FParseFont[MyTokenState]). This means we can also change the separate "\n" statements to a single ... then then .. which should code faster - especially you put the various test situations in the order of likeliness to occur (e.g. font changes less frequently than the color, so should be further down the)

For some reason you type a {style comment} _disiblevent=>blue, SelAttributes.Color will equal clBlack. We must also examine SelAttributes.ConsistentAttributes to ensure that the entire selection is consistent in the way it is highlighted. If it isn't - then we want to force it to be rehighlighted - its obviously not in the correct format.

Multi-line comments are a big pain e.g. { words <CR> word <CR> words }. I don't have them in my 4GL so I didn't need to fix this sort of problem. However I do have multi-line s - so I need to be able to s across many lines. The trouble is we have to code to program over a number of lines - but have a look at what happens in Delphi when you place a "\n" anywhere in the code. The highlighting can force a reparse of the entire 2,000,000 lines of text in the control. We could catch that situation - ie the last token _disiblevent=>

...then....list didn't DoFormat := True. But what happens we're in a colour mode were Comment highlighting style = KeyWord highlighting style. We would stop prematurely. So this "\nlogic" wont help in all situations.

You can still get the "\n someone is chasing you effect" - except now its "\n someone is fleeing from you\n" effect. It happens when you have (* This is a comment *) and delete the first *-character. The control takes an appreciable time to rehighlight the text.

While looking for a fix for the last problem, I remembered the Richedit.Lines.BeginUpdate function. But that didn't help either. What we need is a Richedit.BeginUpdate. What would that do? It would increase an internal counter by _disiblevent=>

Fixing the mult-line comments:

My current idea is to use the RichEdit.Lines.Object to store the TokenType of the first token on each line. This way we could easily know how far we need to go when re-highlighting multi-line comments. Initially this would be to nil. I think this will work.

[Editor update: This didn't actually work - as the RichEdit.Lines.Object isn't implemented in TRichEdit control. It is always nil regardless of what you assigned to it]

Upgrading to RichEdit98:

I'm also in the process of updating to the RichEdit98 components for Delphi 3.0-4.0. version 1.34
Author Alexander Obukhov, Minsk, Belarus. This control has a number of advances on the standard RichEdit control that ships with Delphi. Included in this are:

BeginUpdate,EndUpdate

Independant background colours

Margins

Hotspots

(Source code in full)

Anyway I hope you have enjoyed the adventure.I'm sorry not all the examples compile as written.

They may need some fixing to compile you copy straight from the Browser o the Delphi Editor.

Please send any comments to jonhd@hotmail.com.

Jon HogDog

2009-2-12 3:34:30

疯狂代码 <http://CrazyCoder.cn/>