

控件开发:讲述如何开发一个控件,很有价值

(六)

疯狂代码 <http://CrazyCoder.cn/> <http://CrazyCoder.cn/Delphi/Article11916.html>

ASH - Automatic Syntax highlight (Attempt 2)

[Please note: I have my Delphi Editor colors -to the [Ocean] colour speed tings for testing purposes. This ting works well _disibledevent=> Now put the following base code o the [OnChange] event of the RichEdit1 in Unit1.pas of Project1. Most of this code is just based _disibledevent=> procedure TForm1.RichEdit1Change(Sender: TObject); var

```
WasSelStart,WasRow,Row,BeginSelStart,EndSelStart: Integer;  
MyRe: TRichEdit;  
MyPBuff: .gif' />[0..255] of char;
```

```
begin
```

```
MyRe := TRichEdit(Sender);  
WasSelStart := MyRE.SelStart;  
WasRow := MyRE.Perform(EM_LINEFROMCHAR, MyRE.SelStart, 0);  
BeginSelStart := MyRe.Perform(EM_LINEINDEX, Row, 0);  
EndSelStart := BeginSelStart + Length(MyRE.Lines.Strings[Row]);  
Row := WasRow;
```

```
end;
```

Were going to use the GetToken function to do all the hard work. We'll need some extra variables to pass to the GetToken function, so add to the var section:

```
MyTokenStr;;  
MyTokenState:TTokenState;
```

```
MyRun:PChar;  
MySelStart: Integer;
```

These are similar to the variables we used in the ConvertReadStream - in fact we want to do
\"exactly\" the same thing, just _disibledevent=>MySelStart := BeginSelStart;
MyRun := MyPBuff;

```
while(MyRun^ <> #0) do  
begin
```

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

```
//  
// ScanForRtf;  
// SetRtf;  
// WriteBuffer(Prefix + TokenStr + Postfix);  
//
```

```
end;
```

```
end;
```

NB: As we will be using PasCon you'll have to move it from being a local variable of
TForm1.Button1Click to be a global variable. This will mean you'll have to move all the initialising:

```
PasCon:=TPasConversion.Create;  
PasCon.UseDelphiHighlighting(3);
```

to a TForm1.Show, and the PasCon.Free to TForm1.Close procedure. It will still work you
_disibledevent=>

We did this back in the beginning remeber? When we the >10 character lines to the color red. But
how do we do this now? Lets look at what we have in the variables at hand when we hit \"// SetRtf\"
the first time:

(these example uses Uni1.pas as the input file as its more eresting)

VARIABLES

```
01234567901234567890
Lines.Strings[R0] unit Unit1;
MyPBuff unit Unit1;
MyTokenState tsIdentier
MyTokenStr unit
MyRun Unit1;
```

So what we need to do is select the word `\unit\` in the RichEdit control, and its attributes. We do this by ting `SelStart` to the position of `\unit\` in the RichEdit control, and `SelLength` to the length of the word `\unit\`. And since `\unit\` is at the beginning of the current line - thats position is `BeginSelStart` (which I conveniently have stored in `MySelStart` - you'll see why). Lets replace the `\pseudo\` comment code with the following:

```
MyRe.SelStart := MySelStart;
MyRe.SelLength := Length(MyTokenStr);
MyRe.SelAttributes.Assign(PasCon.FParseFont[MyTokenState]);

end;
```

But remember we are in a loop - when we go around again we'll have the next token in the line, and the variables will look like this:

VARIABLES

```
01234567901234567890
Lines.Strings[R0] unit Unit1;
MyPBuff unit Unit1;
MyTokenState tsSpace
MyTokenStr (space character)
MyRun Unit1;
```

But (space character) isn't at BeginSelStart (#0) in the RichEdit control. Its further along (at position #4). Which just happens to be BeginSelStart + Length('\unit\'). We need to update MySelStart after we process the preceding token, but before we go around the loop again:

```
MySelStart := MySelStart + Length(MyTokenStr);
```

```
end;
```

Okay - this is where we are standing at the moment:

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

```
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);

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);

MyRe.SelAttributes.Assign(PasCon.FParseFont[MyTokenState]);

MySelStart := MySelStart + Length(MyTokenStr);

end;

MyRE.SelStart := WasSelStart;
MyRE.SelLength := 0;

end;

```

Now: put the Debugging code _disibledevent=>

Somehow in our event we are triggering off another [OnChange] event. This call to the [OnChange] event code is stored in the message queue. When the event were currently in is finished, a
_disibledevent=>SaveOnChangeIn := MyRe.OnChange;
MyRe.OnChange := nil;

~~~~~rest of code

MyRe.OnChange := SaveOnChangeIn;

end;

Try it out!!! Compile and Run

Open up Unit1.pas in the \"editor\" we have written

Click in the RichEdit in the center of the first lines, in the middle of \"unit\".

Press the [space bar]

Press the [backspace key]

Arrow to the end of the line

Press [Enter]

Press [BackSpace]

[BackSpace] away the entire line

Re-type the entire line

Result: \"functionally\" the Control should look that same as it did before we clicked in it. The line  
\"unit Unit1;\" should highlighted properly as per your Delphi 3.0 Editor (save the background  
colour). However its slow and flickers a great deal. Try opening up a line and just type a long phrase  
- e.g \" (RichEdit = Santa) then GetPresents(\"box of choclote\"); \" and you'll agree with me that:  
GOOD - It is highlighting properly

BAD - There is flickering

BAD - The longer the line gets, the longer it takes to do the re-highlighting

BAD - you get the \"someone is chasing me effect\"

The flickering is due to a number of components. We'll have to deal with each seperately.

The most obvious is the \"selecting\" of each Token. Visually the control is just repeating what we  
were able to do manually - when a piece of text is selected it becomes highlighted by the black

stripe. We need to stop this from happening. Back to the helpfile(s) again. Have a search around, and come back after a snack with some ideas... I'm hungry :-)

Death to the black stripe

Marks: 5/10

Most of you would have found the HideSelection property of the RichEdit control. When it is to TRUE and the RichEdit loses the focus (the user clicks \_disibledevent=>MyRe.Enabled := False;

~~~~~

```
MyRe.Enabled := True;
```

```
end;
```

Oops. I should have known. After all I said it: a disabled Control is not \"Focused\" - barely ten lines ago! When the RichEdit is enabled again, we also have to SetFocus back to it. Shees.. :-)

```
begin
```

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

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

~~~~~

```
MyRe.Enabled := True;
```

```
MyRe.SetFocus;
```

```
end;
```

Try it again. This time things are working better, and we're leaving poor old Edit1 Control alone. That's good practice, as it may have had an [OnFocus] event that does wierder things than what we're trying to do. Maybe not now, but it could in the future!

Marks: 10/10

On the other kind, some of you may have found instead the EM\_HIDESELECTION message in the Win32.HLP. If you had delved in, you would have found something very eresting. The Delphi HideSelection property \_disibledevent=>MyRe.Perform(EM\_HIDESELECTION,1,0);

~~~~~

```
MyRe.Perform(EM_HIDESELECTION,0,0);
```

```
end
```

Yummy. Nice clean coding:

Death to the FLICKER

The next major problem is this bloody flicker. You should pop back o Delphi for a second, and types some lines in its editor, to see it flickers at all. It does. But _disibledevent=>In fact we don't need to do even that - we can just check whether the SelAttributes (which represents the current selection's attributes) is any dferent from what we want to change it to i.e. FParseFont[MyTokenType]. This way even the TokenType had changed, but the and old TokenType shared the same display attributes, we would still conserve our drawing.

Actually the problems is that the RichEdit isn't doing the conserving. In the old text based system I used to use, you pred something to the screen, and it was the same as something already
_disiblevent=>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;

And off you go and try it out... (PS. Yes the last bit of code is bad programming...)

2009-2-12 3:33:51

疯狂代码 <http://CrazyCoder.cn/>