How to expand a TMemo while keying in

Question

I need a memo component or text component (that can handle multiple lines), that should stay at its design time height and get enlarged when it receives focus. In other words, it should automatically grow or shrink while the user types in words.

The following control will do that. You can set a maximum height for the control, if the text needs more space the control will sprout a scrollbar. *WordWrap* should be true: the control as is does not deal well with a horizontal scrollbar, if present.

```
unit PopupMemo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TPopupMemo = class(TMemo)
  private
    FDesigntimeHeight: Integer;
    FFocusedHeight: Integer;
    FMaximumHeight: Integer;
    FCanvas: TControlCanvas;
    procedure CMTextChanged(var msg: TMessage); message CM_TEXTCHANGED;
    procedure SetFocusedHeight(const Value: Integer);
    procedure SetMaximumHeight(const Value: Integer);
    procedure UpdateHeight;
    procedure ChangeScrollbar( value: TScrollStyle );
  protected
    procedure DoEnter; override;
    procedure DoExit; override;
    procedure Change; override;
    procedure AdjustHeight;
    property Canvas: TControlCanvas read FCanvas;
  public
    Constructor Create( aOwner: TComponent ); override;
    Destructor Destroy; override;
    property FocusedHeight: Integer
      read FFocusedHeight write SetFocusedHeight;
  published
    property MaximumHeight: Integer
      read FMaximumHeight write SetMaximumHeight;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('PBGoodies', [TPopupMemo]);
end;

{ TPopupMemo }

procedure TPopupMemo.AdjustHeight;
const
  alignflags: Array [TAlignment] of DWORD =
    (DT_LEFT, DT_CENTER, DT_RIGHT);
var
  oldrect, newrect: TRect;
  newheight: Integer;
  S: String;
begin
  if not HandleAllocated then
```

```
    Exit;
  Perform( EM_GETRECT, 0, lparam(@oldrect));
  S:= Text;
  {DrawText discards a trailing linebreak for measurement, so if the
  user hits return in the control and the new line would require a
  larger memo we do not get the correct value back. To fix that we add
  a blank just for the measurement if the last character is a linefeed.}
  if (Length(S) > 0) and (S[Length(S)] = #10) then
    S := S + ' ';
  Canvas.Font := Font;
  newrect := oldrect;
  DrawText(
    Canvas.Handle,
    PChar(S),
    Length(S),
    newrect,
    DT_CALCRECT or DT_EDITCONTROL or DT_WORDBREAK or DT_NOPREFIX
      or DT_EXPANDTABS or alignflags[ Alignment ]
  );
  if oldrect.bottom <> newrect.bottom then
  begin
    newHeight := Height - (oldrect.bottom-oldrect.top)
      + (newrect.bottom - newrect.top );
    if newHeight > MaximumHeight then
      ChangeScrollbar( ssVertical )
    else
      ChangeScrollbar( ssNone );
    FocusedHeight := newHeight;
  end;
end;

procedure TPopupMemo.Change;
begin
  AdjustHeight;
  inherited;
end;

procedure TPopupMemo.ChangeScrollbar(value: TScrollStyle);
var
  oldpos: Integer;
begin
  if Scrollbars <> value then
  begin
    {Changing the scrollbar recreates the window and looses the
    caret position!}
    oldpos := SelStart;
    Scrollbars := value;
    SelStart := oldpos;
    Perform( EM_SCROLLCARET, 0, 0 );
  end;
end;

procedure TPopupMemo.CMTextChanged(var msg: TMessage);
begin
  AdjustHeight;
  inherited;
end;

constructor TPopupMemo.Create(aOwner: TComponent);
begin
  inherited;
  FFocusedHeight := Height;
  FMaximumHeight := 5 * Height;
  FCanvas:= TControlCanvas.Create;
  FCanvas.Control := Self;
end;

destructor TPopupMemo.Destroy;
begin
  FCanvas.Free;
  inherited;
end;

procedure TPopupMemo.DoEnter;
```

```
begin
  inherited;
  FDesigntimeHeight := Height;
  UpdateHeight;
  {Height := FFocusedHeight;}
end;

procedure TPopupMemo.DoExit;
begin
  inherited;
  Height := FDesigntimeHeight;
end;

procedure TPopupMemo.SetFocusedHeight(const Value: Integer);
begin
  if FFocusedHeight <> Value then
  begin
    if Value > MaximumHeight then
      FFocusedHeight := MaximumHeight
    else
      FFocusedHeight := value;
    if Focused then
      UpdateHeight;
  end;
end;

procedure TPopupMemo.SetMaximumHeight(const Value: Integer);
begin
  if FMaximumHeight <> Value then
  begin
    FMaximumHeight := Value;
    if Value < FocusedHeight then
      FocusedHeight := Value;
  end;
end;

procedure TPopupMemo.UpdateHeight;
var
  line: Integer;
begin
  if HandleAllocated and Focused then
  begin
    Height := FocusedHeight;
    if Scrollbars = ssNone then
    begin
      line := Perform( EM_GETFIRSTVISIBLELINE, 0, 0 );
      if line > 0 then
        Perform( EM_LINESCROLL, 0, - line );
    end;
  end;
end;

end.
```

Original resource:	The Delphi Pool
Author:	Peter Below
Added:	2009-10-26
Last updated:	2009-10-26