Zahlen, Zeichen, Farben

Delphi – Kurs, 2.Teil

Robert P. Michelic
ftp://pcnews.at/pcn/65/michelic/delphi/prg/

Als Fortsetzung des im ersten Teil dieses Beitrags gestarteten Projektes geht es in diesem Teil um die Interpretation eines oder mehrerer Bytes als Zahl, als Zeichen oder als Farbe bzw. Farbanteil. Wieder möchte ich zwei Aspekte beachten: einerseits die Programmiertechnik, andrerseits den didaktischen Aspekt.

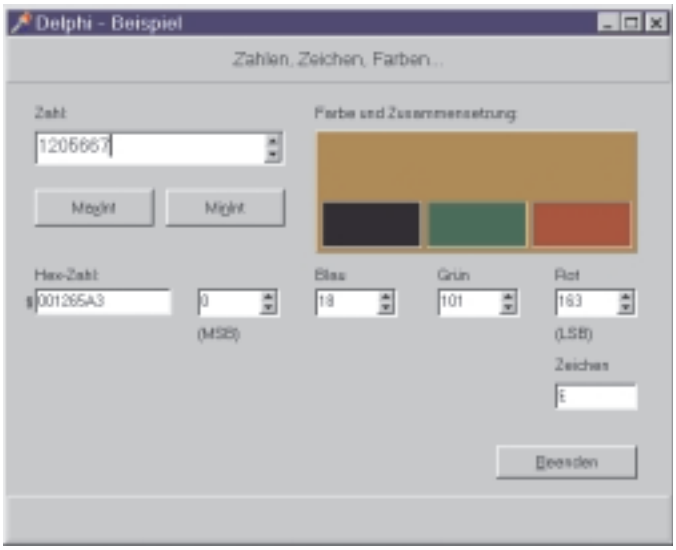
Unsere Aufgabenstellung: Wir wollen sehen, wie ein Byte je nach Interpretation für eine Zahl, ein Zeichen oder auch einen Farbanteil steht. Dazu nehmen wir als zentralen Datentyp eine Integer-Zahl (4 Byte) und zeigen diese als Hexzahl (weil man da die Zerlegung in die einzelnen Bytes besser sieht), als Farbe mit ihren Farbanteilen (3 der 4 Bytes, jeweils einzeln und als Farbmischung) sowie als Zeichen (nur das letzte Byte).

Wie im ersten Projekt verwenden wir eine Eigenschaft „Zahl" als zentrale Schnittstelle im Programm – jede Änderung des Wertes der „Zahl" widerspiegelt sich in allen Komponenten, die von diesem Wert abhängen. Vorsichtig müssen wir sein, damit wir keine endlose Schleife erzeugen.

Komponenten

Wir verwenden folgende Komponenten: Ein `SpinEdit`, um die Zahl dezimal darzustellen. Mit zwei Buttons `MaxInt` und `MinInt` können wir die größte und kleinste Integer-Zahl direkt erzeugen lassen. Ein Panel mit drei kleineren, eingebetteten Panels eignet sich gut, um die Farbe und ihre drei Farbanteile (rot, grün, blau) darzustellen. Ein Edit-Feld zeigt die hexadezimale Darstellung der Zahl an, vier weitere `SpinEdits` jeweils eines der vier Bytes dezimal. Ein letztes Edit-Feld zeigt uns das Zeichen an, das dem letzten der vier Bytes der Zahl entspricht.

Um die Farbe im Panel zu ändern, verwenden wir eine Komponente `ColorDialog`, die den entsprechenden Windows-Dialog



kapselt.

Programm

Hier sind nur die wichtigsten Teile des Programms erläutert, die Details sind im Listing zu finden.

Wie im ersten Projekt ist die `procedure SetZahl` die zentrale Schaltstelle:

```
procedure TfrmSeminar.SetZahl(const Value: integer);
var Zahl3:integer;
begin
  if not Internal then begin
    Internal:=true;
    FZahl:=Value;
    try
      seZahl.Value:=FZahl;
      edChar.Text:=Char(FZahl);
      if not edHex.Focused then edHex.Text:=IntToHex(FZahl,8);
      seMSB.Value:=(FZahl and $FF000000) shr 24;
      Zahl3:=Zahl and $FFFFFF; {die unteren 3 Bytes}
      panColor.Color:=Zahl3;
      panRed.Color:=Zahl3 and $FF;
      panGreen.Color:=Zahl3 and $FF00;
      panBlue.Color:=Zahl3 and $FF0000;
      if not seRed.Focused then seRed.Value:=PanRed.Color;
      if not seGreen.Focused then seGreen.Value:=panGreen.Color shr 8;
      if not seBlue.Focused then seBlue.Value:=panBlue.Color shr 16;
    finally
      Internal:=false
    end
  end
end;
```

Sobald sich der Wert von `Zahl` ändert, werden alle Komponenten aktualisiert, nur jene nicht, die gerade den Focus hat (das wäre sehr lästig beim Editieren). Die Boole'sche Variable `Internal` wacht darüber, dass wir uns nicht selber aufrufen.

Die drei Panels für die Farbanteile rot, grün und blau erhalten das jeweilige Byte zugewiesen, wir verwenden entsprechende Bitoperationen (siehe oben, z.B.: Farbe3 and $FF00), um das richtige Byte herauszufiltern.

In diesem Projekt möchten wir die `Zahl` auch über die Farbe des Panels ändern können, das geht ganz einfach, wenn wir einen vorgefertigten `ColorDialog` (den wir auf unserem Formular platziert haben) dafür verwenden:

```
procedure TfrmSeminar.panColorDblClick(Sender: TObject);
begin
  ColorDlg.Color:=Zahl and $FFFFFF;
  if ColorDlg.Execute then Zahl:=ColorDlg.Color;
```

end;

Sobald auf das Panel doppelgeklickt wird, weisen wir dem `Colordialog` die aktuelle `Zahl` als „Farbe" zu und umgekehrt wird bei der Rückkehr der „Zahl" der Farbwert aus dem Dialog zugewiesen. Für die erste Zuweisung verwenden wir nur die drei hinteren Bytes, da nur diese drei Farbinformation enthalten.

Mit den SpinEdits bei den einzelnen Bytes können wir gezielt einzelne Farbkomponenten beeinflussen, das geht ganz einfach: Alle Änderungen laufen in einer Ereignisbehandlung zusammen:

```
procedure TfrmSeminar.seColorChange(Sender: TObject);
begin
  Zahl:=seMSB.Value shl 24 + seBlue.value shl 16
        + seGreen.Value shl 8 + seRed.Value;
end;
```

Wie schon zum ersten Projekt angemerkt, ist auch diese Projekt etwas vollgestopft für den Unterricht und soll nur dazu anregen, die eine oder andere Idee daraus aufzugreifen.

Der nächste Teil dieses Beitrags geht noch mehr ins Detail, es soll die Zerlegung eines Bytes in seine Bit untersucht werden, dazu entwickeln wir eine eigene Komponente, welche die bitweise Darstellung und Eingabe einer Zahl ermöglicht.

Noch das Programmlisting, das jeder auch gerne per Email haben kann, um es nicht abtippen zu müssen (rpmsoft@via.at):

```
unit USem;

// Musterbeispiel
// Robert P. Michelic 1999

interface

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

type
  TfrmSeminar = class(TForm)
    panTitle: TPanel;
    panStatus: TPanel;
    btnClose: TButton;
    seZahl: TSpinEdit;
    Label1: TLabel;
    edChar: TEdit;
    edHex: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    panColor: TPanel;
    Label4: TLabel;
    seBlue: TSpinEdit;
    seGreen: TSpinEdit;
    seRed: TSpinEdit;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    panBlue: TPanel;
    panGreen: TPanel;
    panRed: TPanel;
    Label8: TLabel;
    seMSB: TSpinEdit;
    Label9: TLabel;
    Label10: TLabel;
    btnMaxInt: TButton;
    btnMinInt: TButton;
    ColorDlg: TColorDialog;
    procedure btnCloseClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure seZahlChange(Sender: TObject);
    procedure seColorChange(Sender: TObject);
    procedure edCharChange(Sender: TObject);
    procedure edHexChange(Sender: TObject);
    procedure btnMaxIntClick(Sender: TObject);
    procedure btnMinIntClick(Sender: TObject);
    procedure panColorDblClick(Sender: TObject);
  private
    function GetZahl: integer;
    procedure SetZahl(const Value: integer);
  private
    { Private-Deklarationen }
    FZahl:integer;
    Internal:Boolean;
    procedure SetStatus(const Value: String);
    property Status:String write SetStatus;
    property Zahl:integer read GetZahl write SetZahl;
  public
    { Public-Deklarationen }
  end;

var
  frmSeminar: TfrmSeminar;

implementation

{$R *.}

{ TForm3 }

procedure TfrmSeminar.SetStatus(const Value: String);
begin
  panStatus.Caption:=Value;
end;

procedure TfrmSeminar.btnCloseClick(Sender: TObject);
begin
  Close
end;

procedure TfrmSeminar.FormCreate(Sender: TObject);
begin
  Status:='';
  Zahl:=0;
end;

procedure TfrmSeminar.seZahlChange(Sender: TObject);
begin
  Zahl:=seZahl.Value;
end;

function TfrmSeminar.GetZahl: integer;
begin
  Result:=FZahl;
end;

procedure TfrmSeminar.SetZahl(const Value: integer);
var Zahl3:integer;
begin
  if not Internal then begin
    Internal:=true;
    FZahl:=Value;
    try
      seZahl.Value:=FZahl;
      edChar.Text:=Char(FZahl);
      if not edHex.Focused then edHex.Text:=IntToHex(FZahl,8);
      seMSB.Value:=(FZahl and $FF000000) shr 24;
      Zahl3:=Zahl and $FFFFFF; {die unteren 3 Byte}
      panColor.Color:=Zahl3;
      panRed.Color:=Zahl3 and $FF;
      panGreen.Color:=Zahl3 and $FF00;
      panBlue.Color:=Zahl3 and $FF0000;
      if not seRed.Focused then seRed.Value:=PanRed.Color;
      if not seGreen.Focused then seGreen.Value:=panGreen.Color shr 8;
      if not seBlue.Focused then seBlue.Value:=panBlue.Color shr 16;
    finally
      Internal:=false
    end
  end
end;

procedure TfrmSeminar.seColorChange(Sender: TObject);
begin
  Zahl:=seMSB.Value shl 24 + seBlue.value shl 16 + seGreen.Value shl 8
+ seRed.Value;
end;

procedure TfrmSeminar.edCharChange(Sender: TObject);
begin
  if length(edChar.Text)>0 then Zahl:=ord(edChar.Text[1]);
end;

procedure TfrmSeminar.edHexChange(Sender: TObject);
var Value,i,l:integer;
    ch:Char;
begin
  l:=length(edHex.Text);
  if l>0 then begin
    Value:=0;
    for i:=1 to l do begin
      ch:=Upcase(edHex.Text[i]);
       ch of
        '0'..'9': begin
          Value:=(Value shl 4) + ord(ch)-ord('0');
        end;
        'A'..'F': Begin
          Value:=(Value shl 4) + ord(ch)-ord('A')+10;
        end
        else raise(EConvertError.Create('Keine gültige Hex-Zahl.'));
      end
    end;
    Zahl:=Value;
  end;
end;

procedure TfrmSeminar.btnMaxIntClick(Sender: TObject);
begin
  Zahl:=MaxInt;
end;

procedure TfrmSeminar.btnMinIntClick(Sender: TObject);
begin
  Zahl:=Low(integer);
end;

procedure TfrmSeminar.panColorDblClick(Sender: TObject);
begin
  ColorDlg.Color:=Zahl and $FFFFFF;
  if ColorDlg.Execute then Zahl:=ColorDlg.Color;
end;

end.
```