Huiswerkopgave: Rectangles

Leerdoelen:

- Gebruik maken van bestaande Abstracte Data Types op grond van hun contract (zie het bestand rectangles.pas voor de ADTs TRectangle en TRectangleList);
- Meer ervaring opdoen met ontwikkeling van een eenvoudige GUI applicaties.

De te maken GUI applicatie moet een scherm produceren dat er ongeveer als in Figuur 1 uitziet. Er worden een aantal gele genummerde assen-parallelle rechthoeken getoond, samen met hun zogenaamde *bounding box* in het rood. Met de muis kan een rechthoek geselecteerd worden, die dan lichtblauw getekend wordt en waarvan de index en coördinaten in het paneel rechts te zien zijn. Met de knoppen kan de lijst leeggemaakt worden (*Clear*), uitgebreid worden met een willekeurige rechthoek (*New Rectangle*), of geheel vervangen door een lijst willekeurige rechthoeken (*New List*). Bij *Count* is het aantal rechthoeken te zien.

Illustration 1: Screen shot

De applicatie kan in de volgende stappen opgebouwd worden (zie ook de slides van week 12):

1. Creëer een nieuw project (in een eigen map) voor een GUI applicatie.
2. Stel de compiler opties in.
3. Bouw het form op uit componenten:
   - Rechts ligt een TPanel (property Align = alRight) met daarop 3 keer een TButton, 6 keer een TLabel en 6 keer een TEdit (property ReadOnly = True, Text = '').
   - Links ligt een TPaintBox (property Align = alClient)
   - Geef deze componenten betere namen (property Name: ClearButton, CountEdit)
   - Je kan het programma nu een keer compileren en uitvoeren. Het doet nog niets, maar je kan wel de componenten zien en het venster van grootte veranderen.
4. Voeg de unit Rectangles in het bestand `rectangles.pas` toe aan het project.
5. (**Nieuw**) Voeg aan het formulier een privé veld FList toe met type TRectangleList, om de lijst met rechthoeken "vast te houden". Dit doe je in het interface deel met de definitie van het form:

```
type
  ...
  TForm1 = class(TForm)
    ...
  private
    { private declarations }
    Flist: Trectangles; { de lijst met rechthoeken }
  public
    ...
```

6. Voeg een lokale procedure UpdateViews aan het form toe (zie ook slide 6) die de edit boxes invult op grond van de FList. Let er op dat SelectedEdit = '' betekent dat er geen rechthoek is geselecteerd, en anders vind je met FList.GetItem(StrToInt(SelectedEdit)) de geselecteerde rechthoek en kun je de coordinater ervan invullen. SelectedEdit blijft zoals die is en wordt elders ingevuld door event handlers.
7. Maak ook lokale functies RandomRectangle en RandomRectangleList:

```
function RandomRectangle: TRectangle;
function RandomRectangleList: TRectangleList;
```

   RandomRectangleList maakt een lijst met een willekeurig aantal willekeurige rechthoeken. Het aantal kan je laten variëren tussen 0 en 19 via Random(20).
8. Voeg een OnCreate event handler toe aan het form (die heet FormCreate als je dit met de Object Inspector via dubbelklikken in het OnCreate event van het form doet). Deze methode dient Randomize aan te roepen en FList in te stellen op een willekeurige lijst rechthoeken (via RandomRectangleList). Bovendien roept deze handler de procedure UpdateViews aan. Je kan het programma nu proberen, al tekent het nog niets. Wel moet de Count te zien zijn.
9. Verder hebben we lokale procedures nodig om (lijsten van) rechthoeken te tekenen:

```
procedure DrawRectangle(ARectangle: Trectangle;
```

```
      APenColor: Tcolor; ABrushColor: TColor; AText: String);
  procedure DrawRectangleList;
  procedure DrawSelectedRectangle;
```

Om je wat op weg te helpen volgt hier de code voor het tekenen van een rechthoek, waarvan de rand APenColor heeft, het binnenste ABrushColor, en waar linksboven AText staat:

```
procedure TForm1.DrawRectangle(ARectangle: TRectangle;
    APenColor: TColor; ABrushColor: TColor; AText: String);
var
  VRect: Trect; { om tekst te tekenen }
begin
  with PaintBox1.Canvas do begin
    // draw rectangle
    Pen.Color := APenColor;
    Brush.Color := ABrushColor;
    with ARectangle do begin
      Rectangle(XL, YL, XH+1, YH+1);
      // draw text in rectangle that fits within ARectangle
      VRect := Rect(XL+1, YL+1, XH, YH);
      TextRect(VRect, XL, YL, AText);
    end;
  end;
end;
```

10. Lokale procedures DrawRectangleList en DrawSelectedRectangle zijn hiermee eenvoudig te maken: de lijst in met ABrushColor = clYellow, en de geselecteerde met clAqua). N.B. Als SelectedEdit.Text = '', dan is er geen geselecteerde rechthoek. Let er bij het maken DrawRectangleList op dat je dit **van achter naar voren** doet, opdat de laagst genummerde rechthoek bovenop ligt. (Figuur 1 doet die verkeerd!)
11. Maak nu de OnPaint event handler van de PaintBox en laat deze de lijst FList tekenen, alsmede daarna de geselecteerde rechthoek (in deze volgorde, dan komt de geselecteerde rechthoek bovenop te liggen).
12. Compileer het programma en probeer het eens uit. Er wordt geen bounding box getekend. Je kan ook nog geen rechthoek met de muis selecteren. Bovendien doen de knoppen niets.
13. Maak OnClick event handlers voor de drie knoppen. Na het creëren van een losse rechthoek, kun je deze geselecteerd maken: SelectedEdit.Text := IntToStr(FList.Count – 1). Bij de twee andere doe je SelectedEdit.Text := ''. Roep telkens als laatste UpdateViews aan. Probeer het programma opnieuw. Nu moet je na New Rectangle deze rechthoek in het lichtblauw zien.
14. Om de bounding box te tekenen introduceren we de lokale procedure

    ```
    procedure DrawBoundingBox;
    ```

    die gebruik kan maken van de procedure Hull van TRectangle, om daarmee een omhullende rechthoek te bepalen en die via DrawRectangle te tekenen met APenColor = clRed , ABrushColor = clWhite, AText = ''. Roep DrawBoudingBox als eerste aan in de OnPaint event handler van de PaintBox. Probeer het programma.
15. Tenslotte kun je via een OnMouseDown event handler van de PaintBox een rechthoek laten

selecteren. Daartoe is het handig om even een lokale functie FindFirstWindow te maken om in een ARectangleList van type TRectangleList een een punt met coördinaten AX, AY op te zoeken. De functie retourneert de index van de eerste rechthoek met het punt, of -1 als die niet bestaat.

De OnMouseDown event handler heeft de volgende heading:

```
procedure TForm1.PaintBox1MouseDown(Sender: Tobject;
  Button: TmouseButton; Shift: TShiftState; X, Y: Integer);
```

Hierin zijn X, Y de coördinaten van het punt waar met de muis werd geklikt. Via FindFirstWindow kun je dit punt opzoeken in FList en daarmee SelectedEdit aanpassen.

16. Controleer je programma. Als je tevreden bent stuur je het in bij peach en laat je het aftekenen bij een assistent (als je niet helemaal zeker bent laat je het eerst zien en stuur je het daarna pas in).