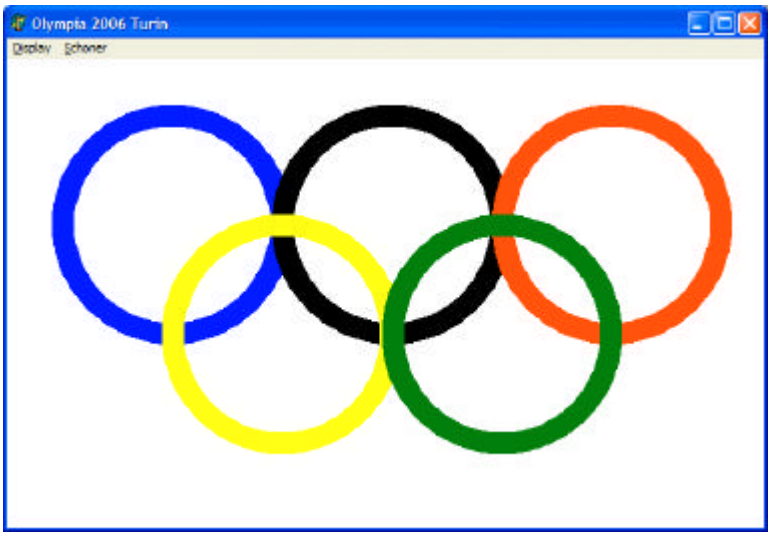
Projekt Olympische Ringe, Teil 3

Schreiben Sie ein Projekt, welches die Olympischen Ringe zeichnet.

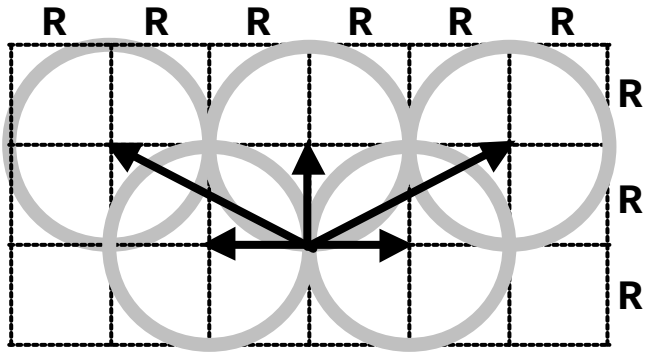


- Die Menüauswahl **Ringe** soll die Olympischen Ringe zentriert auf das Fenster zeichnen (s.rechts):
- Die Menüauswahl **Vollbild** soll das Fenster voller Olympischer Ringe (ca. 100) zeichnen.
- Beim Menüpunkt **Bewegung** sollen die Ringe zufällig auf dem Bildschirm erscheinen und sich dann – immer größer werdend – über den Bildschirm bewegen.
- Beim Menüpunkt **Rotation** kommt zur Bewegung noch eine Rotation der Ringe hinzu.

Arbeitsauftrag / Rotation

1. Um die Ringe rotieren zu lassen, ist es vorteilhaft, die Positionen der einzelnen Ringe **innerhalb der Prozedur Ringe** in Polarkoordinaten (Radius und Winkel) anzugeben.

Wählt man den unteren Schnittpunkt dreier Kreisringe als Drehpunkt (X|Y) (s.Abb.), so ergeben sich folgende Abstände zu den fünf Ringmittelpunkten (Pythagoras):



$$r_1 = R$$

$$r_2 = \sqrt{(2 \cdot R)^2 + R^2} = \sqrt{5} \cdot R$$

Für die Winkelkoordinaten φ ergibt sich:

grün	R	φ = 0	(0°)
schwarz	R	φ = π / 2	(90°)
gelb	R	φ = π	(180°)
rot	R ∗ sqrt(5)	φ = 0 + 0.4636476	
blau	R ∗ sqrt(5)	φ = π – 0.4636476	

aus $\tan(\varphi) = \frac{\sin(\varphi)}{\cos(\varphi)} = \frac{\Delta y}{\Delta x} = \frac{R}{2R} = 0.5$

Alle fünf Winkelkoordinaten, welche die relative Lage der Ringe beschreiben, werden bei einer Drehung um den variablen Drehwinkel α erhöht. Die x- und y-Koordinaten ergeben sich wiederum als Projektion auf die Achsen, d.h. als Cosinus- und Sinuswerte, so z.B. für

den blauen Ring:

```
Canvas.Pen.Color := clBlue;
Xm := X + round(sqrt(5)*R*cos(pi-0.4636476+alpha));
Ym := Y - round(sqrt(5)*R*sin(pi-0.4636476+alpha));
Kreis(Xm, Ym, R);
```

und den schwarzen Ring

```
Canvas.Pen.Color := clBlack;
Xm := X + round(R*cos(pi/2+alpha));
Ym := Y - round(R*sin(pi/2+alpha));
Kreis(Xm, Ym, R);
```

- Die Funktion **`round(real)`** rundet einen Wert des Typ Real auf einen Integer-Wert.
- Ändern Sie Ihren Quelltext entsprechend für die restlichen Ringe ab.

2. Da der **Prozedur Ringe** nunmehr auch der Drehwinkel übergeben werden muss, muss auch der Prozedurkopf und die Deklaration entsprechend abgeändert werden:

```
private
    { Private declarations }
    procedure Kreis(Xm, Ym, R: integer);
    procedure Ringe(X,  Y,  R: Integer; alpha:real);
```

```
procedure TFoOlympia.Ringe(X, Y, R: Integer; alpha:real);
var Xm, Ym: Integer;
```

Der Drehwinkel **`alpha`** selbst muss wiederum als globale Variable deklariert werden.

Wenn Sie nun die Compilierung starten, wird Delphi automatisch bei jedem Aufruf der Prozedur **`Ringe(x,y,r)`** mit der Fehlermeldung „Nicht genügend wirkliche Parameter“ stoppen und Sie zur Ergänzung eines weiteren Übergabewertes (für **`alpha`**) auffordern. Erweitern Sie vorerst nur mit einem festen Wert z.B. **`Ringe(x,y,r,pi/2)`**.

3. Um nun die Ringe rotieren zu lassen, muss der Wert des Drehwinkels bei jedem Schleifendurchlauf entsprechend erhöht werden: **`a := a + alpha;`**

- Für **`alpha <> 0`** erfolgt bei jedem Schleifendurchlauf eine Drehung um den Übergabewert **`alpha`**.
- Für **`alpha = 0`** bleibt a konstant und es erfolgt keine Rotation.

```
while <Bedingung> do
  begin
    Ringe(x, y, r, a);
    sleep(20);
    Ringe(x, y, r, a);
    x := x + xdir;
    y := y + ydir;
    a := a + alpha;
    r := r + 1;
  end;
```

4. Um beide Möglichkeiten

- Bewegung ohne Rotation für **`alpha = 0;`**
- Bewegung mit Rotation für **`alpha <> 0;`**

aufrufen zu können, muss

- die (bisher als Ereignisbehandlungsroutine) deklarierte Prozedur **`TFoOlympia.Bewegung1Click(Sender: TObject);`**
- als private Prozedur **`TFoOlympia.Bewegung(alpha: real);`**

deklariert werden, welche dann von den neu zu deklarierenden Ereignisbehandlungsroutinen (siehe unten) mit verschiedenen Übergabewerten aufgerufen werden kann:

```
procedure TFoOlympia.Rotation1Click(Sender: TObject);
begin
  Bewegung(pi/20);
end;
```

```
procedure TFoOlympia.Bewegung1Click(Sender: TObject);
begin
  Bewegung(0);
end;
```