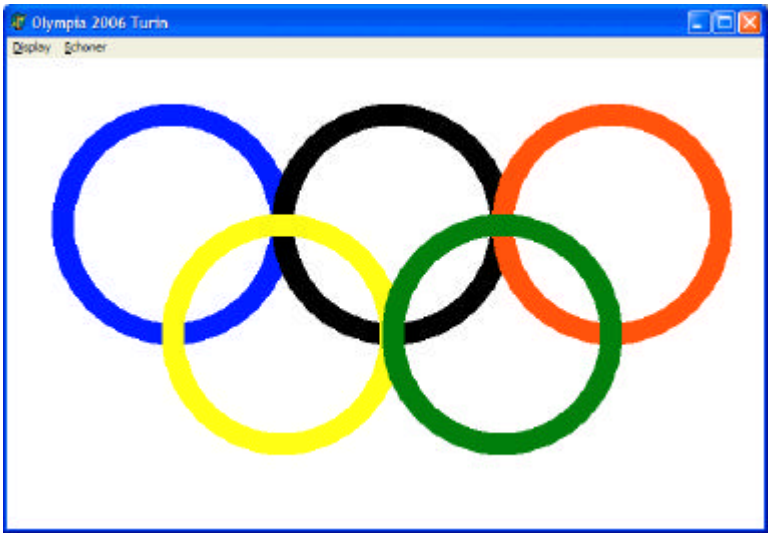
Projekt Olympische Ringe, Teil 2

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 / Bewegung

1. Implementieren und deklarieren (Private declarations) Sie nun das Unterprogramm:

```
procedure TFoOlympia.Bewegung1Click(Sender: TObject);
```

An einer zufälligen Stelle **(x,y)** sollen die Olympischen Ringe auftauchen (Der Befehl **randomize** startet den Zufallsgenerator),

```
randomize;
x := random(ClientWidth);
y := random(ClientHeight);
```

sich in einer zufälligen Richtung **(xdir,ydir)**

```
xdir := random(10)-5;
ydir := random(10)-5;
```

über den Bildschirm hinweg bewegen und dabei immer größer werden **(r:=r+2),** **solange** sie innerhalb des Fensters liegen.

```
x := x + xdir;
y := y + ydir;
r := r + 2;
```

Sie benötigen also eine äußere Schleife, die angibt, wie viele Male die Ringe erscheinen sollen und eine innere Schleife, die regelt, wie lange sich die Ringe bewegen sollen.

Nutzen Sie hierfür eine Schleife mit Anfangsbedingung, die **WHILE-DO-Schleife** (Syntax siehe rechts), wobei die **<Bedingung>** alle Kontakte der olympischen Ringe mit den Bildschirmbegrenzungen (links, rechts, oben, unten) beinhalten soll. Mehrere Bedingungen werden dabei mit dem logischen **AND** verknüpft:

```
While <Bedingung> do
begin
  <Anweisung>
end;
```

```
while (x > 3*r) and (x < ClientWidth  - 3*r)
  and (y > 2*r) and (y < ClientHeight - r) do
begin
.....
end;
```

2. Damit die Olympischen Ringe bei ihrer Bewegung keinen „Schweif“ hinterlassen, müssen sie an der alten Position gelöscht werden, bevor sie an ihrer neuen Position neu gezeichnet werden.

- Der Modus **`Canvas.Pen.Mode := pmNotXor;`** löscht durch erneutes Zeichnen.
- Der Modus **`Canvas.Pen.Mode := pmCopy`** schaltet wieder in den Zeichenmodus.

Beachten Sie, dass Sie für die Prozedur **Bewegung** nur den Modus **`pmNotXor`** benötigen, die Ringe nun jedoch zweimal innerhalb des Schleifenkörpers zeichnen müssen.
In der Prozedur **Vollbild** müssen Sie wieder in den Modus **`pmCopy`** zurückschalten.

3. Zwei Eigenschaften widersprechen noch denen eines Bildschirmschoners:

- Das Programm lässt sich nicht beenden, solange es sich innerhalb einer Schleife befinden.
- Durch die äußere Zählschleife wird eine feste Anzahl von Schleifendurchläufen festgelegt, d.h. es läuft eben nicht **solange bis** eine Tastatureingabe erfolgt.

Die Bedingung „solange bis“ entspricht einer **Abbruchbedingung**, d.h. wir ersetzen die

Schleife mit Zähler

```
for i := 1 to 20 do
begin
......
end;
```

durch eine

Schleife mit Abbruchbedingung

```
repeat
......
  application.ProcessMessages;
until abbruch;
```

Durch den Aufruf der Standardfunktion **`application.ProcessMessages;`** innerhalb des Schleifenkörpers erreicht man, dass Tastatureingaben berücksichtigt werden.

Die Abbruchbedingung wird hier nicht durch eine Vergleichsoperation bestimmt, sondern besteht lediglich in der Abfrage des Wahrheitswertes der Variablen **`abbruch`**.

Diese **Variable vom Typ `boolean`** kann nur zwei Werte annehmen: true bzw. false.

- Solange Sie den Wert **`false`** besitzt, wird die Schleife erneut durchlaufen,
- besitzt sie den Wert **`true`**, erfolgt der Schleifenabbruch.

Überlegen Sie sich, durch welche Tastatur- bzw. Mauseingabe (Ereignisse der Form) die Variable auf true gesetzt werden soll, z.B. durch einen Mausklick auf das Fenster:

```
procedure TFoOlympia.FormClick(Sender: TObject);
begin
 abbruch := true;
end;
```

und beachten Sie, dass es sich um eine **globale** Variable handeln muss, da sie innerhalb einer Prozedur verändert und innerhalb einer anderen abgefragt wird.
Sie muss also noch vor dem implementation Abschnitt deklariert werden !

Schleife mit Zähler	Schleife mit	
	Durchlaufbedingung	Abbruchbedindung

`for i := 1 to 20 do`
`begin`
`  <Anweisung>`
`end;`
	`While <Bedingung> do`
	`begin`
	`  <Anweisung>`
	`end;`
		`repeat`
		`  <Anweisung>`
		`until <Bedingung>;`