

26. Die REPEAT ... UNTIL ... - Schleife

Bei vielen Programmen ist es praktisch, wenn Anweisungen mehrfach ausgeführt werden, bis irgendwann eine Bedingung erfüllt ist, die zum Abbruch dieser „Schleife“ führt. Bei dieser hier erklärten Wiederholungsanweisung werden die Befehle, die zwischen **REPEAT** und **UNTIL** („Wiederhole ... bis ...“) stehen, solange wiederholt, bis die Bedingung hinter **UNTIL** wahr ist.

- Das Befehlspaar **REPEAT ... UNTIL** fasst die zwischen ihnen stehenden Anweisungen zu einem *Block* zusammen, auch ohne dass sie noch extra von **BEGIN ... END** eingeklammert sind.
- Die Anweisungen zwischen **REPEAT** und **UNTIL** werden mindestens einmal ausgeführt, da die Abbruchbedingung ja erst am Ende der Schleife überprüft wird.
- Ein Ausstieg mittendrin in (je)der Schleife ist möglich mit dem Befehl **BREAK**.

Beispiel: **procedure TForm1.Button1Click(Sender: TObject);**

```
begin
  REPEAT
    Edit1.Text := IntToStr(Random(54321));
    Beep;
    // Application.ProcessMessages;
  UNTIL StrToInt(Edit1.Text) = 333;
end;
```

Etwas störend ist, dass die Bildschirmanzeige nicht permanent aktualisiert wird: In obigem Beispiel tut sich längere Zeit nichts Erkennbares bis zum Ende der Schleife, dann erscheint schlagartig die 333 im Edit-Feld und es piepst einmalig. Wenn es nicht auf Geschwindigkeit ankommt, sollte deshalb in die Schleife besser die Anweisung **Application.ProcessMessages** eingebaut werden. Damit wird die „Botschaftswarteschlange“ von Windows abgebaut, also insbesondere der Bildschirm aktualisiert, bevor die Ablaufsteuerung wieder auf die Anwendung übertragen.

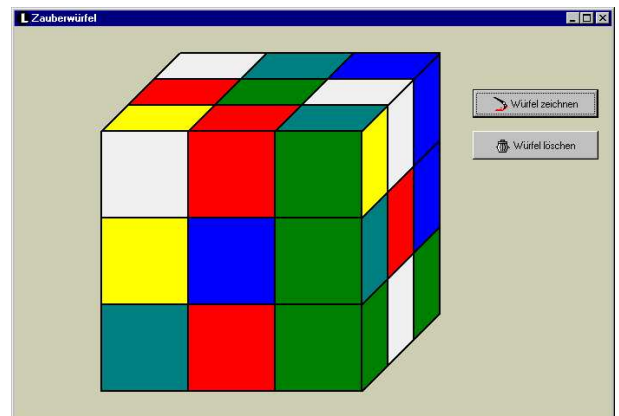
Aufgaben

1. Zeichne mehrere ineinandergeschachtelte Rechtecke mit ausgelosten Farben (selbstredend soll das in einer REPEAT-UNTIL-Schleife passieren, die anhand der aktuellen Formulargröße merkt, wie viele Rechtecke zu zeichnen sind).

Auch für die folgenden Aufgabe ist es sinnvoll, alle Zeichenaktionen sofort beim Programmstart ausführen zu lassen. Dafür ist allerdings **OnCreate** nicht geeignet, benutze **OnPaint**.

2. Zeichne das Schrägbild eines beliebigen Quaders. Die sichtbaren Kanten sollen schwarz, die unsichtbaren Kanten dünner und grau (**Pen.Color := clGray**) dargestellt sein.

3. Kennst du den Zauberwürfel „Rubics-Cube“ ? Jede Seite besteht aus 9 bunten Quadraten, die sich gegenseitig verdrehen lassen. Es ist Aufgabe des Spielers, die Quadrate so zu verdrehen, dass alle Seiten aus farbgleichen Quadraten bestehen. Auch wenn du die Lösung für den Zauberwürfel nie hinbekommst, gelingt es dir sicher, einen solchen Würfel auf den Bildschirm zu zeichnen ☺. Benutze für die vielen parallelen Linien **FOR-TO-DO-Schleifen**. (*Zugegeben, das mit den Koordinaten ist nicht ganz leicht. Vielleicht hilft es dir, wenn du den Würfel samt Koordinaten der Linienendpunkte auf Papier vorzeichnest.*)



Ach ja: Wenn die Linien des Würfels fertig sind, soll für jedes der 27 sichtbaren Würfelquadrate die Flächenfarbe ausgelost werden, 6 Farben stehen zur Auswahl. Für die Farb-Auslosungen solltest du eine „eigene“ Prozedur benutzen, zum Färben der Flächen dann die Methode **FloodFill**.

Der untere Button soll für einen sauberen Bildschirm sorgen - benutze auch hierfür **FloodFill**.