

Formen animieren

Table of contents

1 Ein einfach animiertes Dreieck.....	2
1.1 Die Angabe der Koordinaten.....	3
1.2 Die Do-While-Loop-Schleife.....	5

1. Ein einfach animiertes Dreieck

Im Prinzip können Sie per Makro jede beliebige Form erzeugen, d.h. Polygone oder sogar Kurven. Die Schwierigkeit liegt hier weniger in der Programmierung als in den mathematischen Kenntnissen, da für die präzise Berechnung komplizierter Formeln erforderlich sind. Umgekehrt kann es aber manchmal einfacher sein, eine exakte Form per Makro erzeugen, wie im [Beispiel-Makro](#).

Im dieser kleinen Animation wird zunächst ein gleichschenkliges Dreieck erzeugt.

Das vom Makro erzeugte Dreieck

Dieses wird anschließend vertikal gekippt, wodurch - wie bei Animationen üblich - der Eindruck einer Bewegung entsteht. Im Beispiel wird die kleine Animation zehnmal durchlaufen, entsprechend oft „wackelt“ das Dreieck.

Das umgeklappte Dreieck

Löschen Sie die Form, bevor Sie das Makro eventuell ein weiteres Mal ausführen.

Den gesamten Code des Makros zeigt die anschließende Übersicht. Die hier auskommentierten Code-Zeilen ganz unten beinhalten die Do-While-Loop-Schleife als elegantere Alternative zur Do-While-Schleife davor.

```

    REM ***** BASIC *****
    Option Explicit

    Dim oDokument As Object
    Dim oSeite As Object
    Dim oPolygonForm As Object
    Dim aKoordinaten(2) As New com.sun.star.awt.Point
    Dim iAnimationsdurchlauf As Integer

    Sub Main

        oDokument = ThisComponent
        oSeite = oDokument.DrawPages(0)
        oPolygonForm =
oDokument.CreateInstance( "com.sun.star.drawing.PolyPolygonShape" )

        aKoordinaten(0).x = 5000
        aKoordinaten(0).y = 5000
        aKoordinaten(1).x = 7500
        aKoordinaten(1).y = 7500
        aKoordinaten(2).x = 10000
        aKoordinaten(2).y = 5000

        oPolygonForm.PolyPolygon = Array(aKoordinaten())

```

```
oSeite.add(oPolygonForm)

iAnimationsdurchlauf = 0
Do While iAnimationsdurchlauf < 10
    Wait(1000)
    aKoordinaten(1).y = 2500
    oPolygonForm.PolyPolygon = Array(aKoordinaten())

    Wait(1000)
    aKoordinaten(1).y = 7500
    oPolygonForm.PolyPolygon = Array(aKoordinaten())
    iAnimationsdurchlauf = iAnimationsdurchlauf + 1
Loop

'Alternativer Do While Loop:
'
'Do While iAnimationsdurchlauf < 10
'    Wait(1000)
'    If aKoordinaten(1).y = 2500 Then
'        aKoordinaten(1).y = 7500
'    Else
'        aKoordinaten(1).y = 2500
'    End If
'    oPolygonForm.PolyPolygon = Array(aKoordinaten())
'    iAnimationsdurchlauf = iAnimationsdurchlauf + 1
'Loop

'Das Polygon wieder von der Seite entfernen
```

```
oSeite.remove(oPolygonForm)
```

```
End Sub
```

1.1. Die Angabe der Koordinaten

Im Unterschied zum Rechteck geben Sie bei einem Dreieck oder anderen, frei wählbaren Polygonen die genauen Koordinaten für die Ecken an. Statt einem Punkt für die Position der linken oberen Ecke des Rechtecks wie im letzten Beispiel benötigen Sie diesmal eine Sammlung von Punkten, die Sie in einem Datenfeld (Array) speichern können.

```
Dim aKoordinaten(2) As New com.sun.star.awt.Point
```

Das Array wird genau wie eine einfache Variable deklariert. Da Sie jedoch mehrere Werte unterbringen müssen, wird die Größe des Datenfelds in () bereits hier angegeben. Im Fall des Dreiecks handelt es sich logischerweise um drei Punkte, Sie benötigen also als Datentyp wieder den Punkt (Point) und zwar drei Mal. Da der Index eines Arrays standardmäßig mit 0 beginnt, steht der hier angegebene Wert 2 für den höchsten Index der Werte 0,1,2, somit also für 3 Werte.

Für die kleine Animation in diesem Makro benötigen Sie eigentlich nichts Neues, Sie können sie genauso auch in einer For-Next-Schleife wie im letzten Beispiel laufen lassen. Hier wurde jedoch mit einem Do-While-Loop eine weitere Form der Schleife eingeführt. Für die Abbruchbedingung dieser Schleife wird hier die Integer-Variable `iAnimationsdurchlauf` verwendet.

```
Dim iAnimationsdurchlauf As Integer
```

Der Zugriff auf das Dokument und die Zeichenseite läuft genauso ab wie im letzten Beispiel. Auch wird wiederum ein Form-Objekt erzeugt, in diesem Fall `PolyPolygonShape`. Dies ist eine Form, die mehrere frei wählbare Vielecke (Polygone) enthalten kann, in diesem Fall enthält sie allerdings nur ein Polygon.

```
oPolygonForm =  
oDokument.CreateInstance("com.sun.star.drawing.PolyPolygonShape")
```

Wie im letzten Beispiel erfolgen die Koordinaten-Angaben wieder Hundertstelmillimetern und zwar jeweils von der linken oberen Ecke des Dokuments aus gerechnet. Statt eines Punktes, der in einer Variablen gespeichert wird, handelt es sich nun um drei Punkte, die jeweils in einem Element des Arrays `aKoordinaten` gespeichert werden. Um welches Element es sich jeweils handelt, geben Sie über den Index in `()` an.

Der erste Punkt im ersten Element `aKoordinaten(0)` hat als Struct Point - wie bereits bekannt - zwei Eigenschaften, nämlich den x- und den y-Wert. Dasselbe gilt natürlich auch für die zwei weiteren Punkte.

```
aKoordinaten(0).x = 5000  
aKoordinaten(0).y = 5000  
aKoordinaten(1).x = 7500  
aKoordinaten(1).y = 7500  
aKoordinaten(2).x = 10000  
aKoordinaten(2).y = 5000
```

Nachdem das Objekt `PolyPolygonShape` grundsätzlich mehrere Polygone enthalten kann, erwartet seine `PolyPolygon`-Eigenschaft ein Array, das selbst mehrere Arrays mit den Koordinaten der einzelnen Polygone beinhaltet. Obwohl im Beispiel nur ein Polygon vorhanden ist, muss der von der Eigenschaft erwartete Datentyp geliefert werden. D.h. es muss ein Array mit einem Element, hier dem Array `aKoordinaten()` erzeugt werden. Dafür benutzen Sie die BASIC-Funktion `Array()`.

```
oPolygonForm.PolyPolygon = Array(aKoordinaten())
```

Damit Ihr Polygon sichtbar wird, muss es wie im letzten Beispiel der Zeichenseite hinzugefügt werden.

```
oSeite.add(oPolygonForm)
```

Die Koordinaten können auch nach dem Hinzufügen zur Seite noch geändert werden. Da es sich hierbei um Manipulationen am sichtbaren Polygon handelt, eignet sich diese Vorgehensweise auch für Animationen.

1.2. Die Do-While-Loop-Schleife

Da das Umklappen des Dreiecks mehrfach, hier konkret zehnmal stattfinden soll, benötigen Sie wieder eine Schleife. Der Do-While-Loop prüft zunächst eine Bedingung, hier ob der Intergerwert von `iAnimationsdurchlauf` kleiner als 10 ist, bevor der Inhalt der Schleife (vor der Zeile `Loop`) ausgeführt wird. Ist die Bedingung wahr (`true`), wird der Code in der Schleife ausgeführt, solange bis die Bedingung falsch (`false`) ist, hier also bei einem Integer-Wert größer als 10.

Anders als bei der For-Next-Schleife wird der Schleifenzähler nicht im Schleifenkopf initialisiert. Hier geschieht dies entsprechend vor der Schleife mit `iAnimationsdurchlauf = 0`, d.h. der Integervariablen wird erstmalig ein Wert, nämlich 0 zugewiesen. Ebenso wird der Schleifenzähler nicht automatisch erhöht, dies übernehmen Sie selbst in der Zeile

```
iAnimationsdurchlauf = iAnimationsdurchlauf + 1
```

vor dem Ende der Schleife. Würden Sie den Schleifenzähler nicht erhöhen, lief die Animation unendlich weiter, da die Variable immer den Wert 0 behalten würde und somit die Bedingung immer `true` ist. In diesem Fall können Sie das Makro im Makro-Editor über die Abbrechen-Schaltfläche stoppen.

```
iAnimationsdurchlauf = 0
Do While iAnimationsdurchlauf < 10
  Wait(1000)
  aKoordinaten(1).y = 2500
  oPolygonForm.PolyPolygon = Array(aKoordinaten())

  Wait(1000)
  aKoordinaten(1).y = 7500
  oPolygonForm.PolyPolygon = Array(aKoordinaten())
  iAnimationsdurchlauf = iAnimationsdurchlauf + 1
Loop
```

Um das „Klappen“ des Dreiecks zu erzeugen, genügt es eine Koordinate einer Ecke zu

verändern. Hier wird der y-Wert von aKoodinaten(1) dafür wechselweise auf 2500 bzw. 7500, also den Ausgangswert, gesetzt.

Das Array der geänderten Koordinaten muss nun jeweils wieder in ein Array gepackt werden.

1.2.1. Variation im Programmierstil

In der bereits vorgestellten Do-While-Loop-Schleife müssen Sie genau nachvollziehen, an welchem Punkt Sie sich gerade befinden, d.h. welchen Wert die Variable aktuell besitzt.

Alternativ gibt es für das „Klappen“ des Dreiecks noch die elegantere Lösung eines „Schalters“. Als dieser fungiert eine If-Else-Verzweigung innerhalb der Do-While-Loop-Schleife. Hierbei überprüft der Code in der Schleife den Wert der Variablen und setzt sie auf den jeweils anderen Wert.

```
'Do While iAnimationsdurchlauf < 10
'   Wait(1000)
'   If aKoordinaten(1).y = 2500 Then
'       aKoordinaten(1).y = 7500
'   Else
'       aKoordinaten(1).y = 2500
'   End If
'   oPolygonForm.PolyPolygon = Array(aKoordinaten())
'   iAnimationsdurchlauf = iAnimationsdurchlauf + 1
'Loop
```

Der Unterschied besteht darin, dass hier pro Schleifendurchlauf nur eine Änderung der Koordinate erfolgt. Im zuerst beschriebenen, anderen Fall wird die Koordinate dagegen stets zweimal geändert.

1.2.2. Entfernen des Polygons

Nach dem Ende der Animation soll das Polygon wieder von der Seite entfernt werden. Andernfalls würden Sie nach mehrmaligem Starten des Makros immer mehr Polygone übereinander erhalten.

Genauso einfach wie Sie ein Polygon einer Seite mit der Methode add() hinzufügen, entfernen Sie es mit der Methode remove(), der das Polygon-Objekt als Parameter übergeben wird.

```
oSeite.remove(oPolygonForm)
```