

Formen erzeugen

Table of contents

1 Farbige Rechtecke automatisch generiert.....	2
1.1 Variablen als Objekte deklarieren.....	3
1.2 Seite im Dokument ansprechen.....	5
1.3 Die For-Next-Schleife.....	6
1.4 Die Gestaltung der Rechtecke.....	7
1.5 Die Methoden Wait() und add().....	9

1. Farbige Rechtecke automatisch generiert

Das vorliegende [Beispiel-Makro](#) greift auf die erste Seite eines in diesem Fall bereits vorhandenen Draw-Dokuments zu.

Danach generiert es mit Hilfe zweier, ineinander verschachtelter For-Next-Schleifen die gewünschte Anzahl von Rechtecken und legt ihre Eigenschaften fest, bevor sie dem Dokument hinzugefügt werden. Wie Sie beim Ausführen des Makros sehen, verändern sich die Farbfüllungen von links nach rechts und von oben nach unten. Dadurch entsteht der Eindruck einer Animation. Nachdem alle 35 Rechtecke gezeichnet sind, erhalten Sie eine Meldung, dass das Makro fertig ist.

In diesem Ausschnitt aus den 35 Rechtecken sehen Sie die Message-Box mit der Abschlussmeldung

Note:

Falls Sie das Makro einmal im Editor starten und die Ausführung nicht sehen, ist einfach nur das Dokumenten-Fenster durch den Editor verdeckt.

Den Code des gesamten Makros zeigt Ihnen die anschließende Übersicht. Sie können sich das heruntergeladene Makro aber auch im Basic-Editor öffnen, um zum Code zu gelangen.

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

Sub Main

Dim oDokument As Object
Dim oSeite As Object
Dim oRechteck As Object
Dim Punkt As New com.sun.star.awt.Point
Dim Groesse As New com.sun.star.awt.Size
Dim iZeile As Integer
Dim iSpalte As Integer

oDokument = ThisComponent
oSeite = oDokument.DrawPages(0)

For iZeile = 1 To 7
  For iSpalte = 1 To 5
    Punkt.x = iSpalte * 3000
    Punkt.y = iZeile * 3000
    Groesse.Width = 2000
    Groesse.Height = 2000
    oRechteck =
```

```
oDokument.CreateInstance("com.sun.star.drawing.RectangleShape")
    oRechteck.Size = Groesse
    oRechteck.Position = Punkt
    'oRechteck.FillColor = RGB(255/iSpalte,iSpalte * 10,iZeile * 35)
'Rot
    'oRechteck.FillColor = RGB(iSpalte * 10,255/iSpalte,iZeile * 35)
'Grün
    oRechteck.FillColor = RGB(iSpalte * 10,iZeile * 35,255/iSpalte)
'Blau
    Wait(100)
    oSeite.add(oRechteck)
    Next iSpalte
Next iZeile
Wait(1000)
MsgBox "Fertig !!!"

End Sub
```

1.1. Variablen als Objekte deklarieren

Aus dem smalltalk-Beispiel kennen Sie bereits die hier ebenfalls vorgenommene Deklaration von Variablen als Integer, um Ganzzahlen aufzunehmen. Dies gilt hier für die Variablen iZeile und iSpalte.

Neu ist hingegen der Datentyp Object, der Ihnen schon beim ersten Blick auf den Code der Variablendeklaration verdeutlicht, dass dieses Makro auch Objekte verwendet.

```
Dim oDokument As Object
Dim oSeite As Object
Dim oRechteck As Object
Dim Punkt As New com.sun.star.awt.Point
Dim Groesse As New com.sun.star.awt.Size
Dim iZeile As Integer
Dim iSpalte As Integer
```

1. Zunächst werden drei Variablen als Objekte definiert, nämlich oDokument, oSeite und oRechteck. Das bedeutet, diese Variablen sollen später die konkreten Objekte aufnehmen, mit denen das Makro arbeitet.
2. Etwas anders sieht es bei den Variablen Punkt und Groesse aus, die als Eigenschaftskonstruktionen (Structs) von Form-Objekten (Shapes) definiert werden. Konkret geht es hier um die Form Rechteck (RectangleShape), die später erzeugt werden soll.

1.1.1. Die Funktion von Structs

Grundsätzlich handelt es sich bei Structs um Bündelungen zusammengehöriger Eigenschaften (Properties). So fasst z.B. Size (Größe) die Eigenschaften Height (Höhe)

und Width (Breite) einer Form zusammen.

Die Übersicht veranschaulicht am Beispiel der Form Rechteck die Zusammenhänge zwischen Form-Objekten, Structs und Eigenschaften.

Auf den ersten Blick erscheint die Verwendung der Structs nicht unbedingt einleuchtend. Immerhin existieren die gebündelten Eigenschaften Size und Position ebenfalls, die Sie als Feature Position und Größe vielleicht auch aus der Draw-Anwendung kennen.

Sie benötigen allerdings die Structs, um die Eigenschaften x und y, width und height mit Werten belegen zu können. Dies liegt schlichtweg daran, dass Sie sich diese Eigenschaften „ausleihen“ müssen und zwar aus dem „abstract windowing toolkit“.

Dieses Paket beinhaltet Klassen für das Erzeugen von Fenstern und deren Elementen, zu denen unter anderem auch Punkte und Größen gehören. Eben diese Elemente verwenden Sie auch bei der Positionierung und Gestaltung Ihrer Form, um die o.g. Werte an Eigenschaften von Objekten zu übergeben.

Der Sinn für diese Wiederverwendung von Code liegt hier schlichtweg in einer effizienteren Programmierung. D.h. der Code wird durch das Herunkopieren von Elementen nicht unnötig aufgebläht.

1.1.2. Structs erzeugen

Damit Sie in Ihrem Makro also auf gebündelte Eigenschaften zugreifen können, müssen Sie zunächst die Structs aus den UNO-Klassen des API erzeugen lassen. Die UNO (Uniform Network Objects)-Klassen sind der Bauplan, nach dem konkrete, aber vorläufig noch „leere“ Objekte gebildet werden. D.h. diese Objekte enthalten zwar bereits Eigenschaften, die Sie aber noch mit Werten füllen müssen.

So besitzt das Objekt Rechteck (RectangleShape) z.B. bereits eine Füllfarbeneigenschaft (FillColor), die konkrete Farbe müssen Sie allerdings aussuchen.

Note:

Manche Objekte, wie z.B. auch das Rechteck, verfügen zwar über Standardeigenschaften, diese sind aber nicht unbedingt erwünscht. So wäre das Rechteck ohne zusätzliche Angaben hellblau, jeweils 1 Millimeter breit und hoch und würde winzigklein an der Position 0,0 in der linken oben Ecke des Dokuments erscheinen.

Aus einer UNO-Klasse lassen sich stets mehrere Instanzen, d.h. mehrere konkrete Objekte herstellen.

Mit New weisen Sie das API an, ein neues konkretes Objekt zur Verfügung zu stellen.

Mit `com.sun.star.awt.Point` bzw. `com.sun.star.awt.Size` geben Sie jeweils den Ort der Klasse für die Structs `Point` bzw. `Size` an.

Note:

Der Ort einer Klasse stellt eine Art Paketpfad dar, dabei beinhalten Pakete stets mehrere Pakete oder Klassen wie z.B. das Paket `star` die `StarOffice`-Pakete, das wiederum das bereits erwähnte Paket `awt` „abstract windowing toolkit“ enthält. `com.sun` wird nach einer für Java entwickelten Namenskonvention vorangestellt. Um Namensgleichheiten zu vermeiden, soll die rückwärtsgelesene Internetdomain (hier `sun.com`) des Autors der Klassen am Beginn des Pfades stehen. Am Ende stehen hier die gewünschten Structs `Point` oder `Size`.

1.2. Seite im Dokument ansprechen

Logischerweise benötigen Sie ein Dokument, um Ihre Formen darstellen zu können. In diesem Fall haben Sie bereits ein `Draw`-Dokument vorliegen, auf das das darin enthaltene Makro zugreifen muss.

- Hierzu stellt die `BASIC`-Laufzeitumgebung ein Standard-Objekt zur Verfügung und zwar `ThisComponent`, das der Variablen `oDokument` zugewiesen wird.

```
oDokument = ThisComponent
```

Dieses Objekt steht jeweils für die Umgebung, in die das Makro eingebettet ist, hier also das Zeichendokument.

Ist ein Dokument vorhanden, benötigen Sie eine Seite, um darauf zeichnen zu können. Da sich im `Draw`-Dokument mehrere Zeichenseiten in einem Dokument befinden können, muss die gewünschte Seite explizit angesprochen werden. Dies gilt auch, wenn das Dokument nur eine Seite enthält. Die Nummerierung der Seiten beginnt bei 0.

Die Zeichenseiten des Dokuments befinden sich einem Datenfeld, `Array` genannt. Ein `Array` ist eine Sammlung von Variablen, die einer speziellen `Array`-Variablen untergebracht sind. Die im `Array` enthaltenen Variablen werden als Elemente bezeichnet und können über einen Index angesprochen werden.

- In diesem Fall enthält das `Array` `DrawPages` alle Zeichenseiten des `Draw`-Dokuments. Möchten Sie die erste, in diesem Fall auch die einzige Seite des Dokuments ansprechen, setzen Sie den Index, hier 0, in runden Klammern() hinter den Variablennamen.

```
oSeite = oDokument.DrawPages(0)
```

- Um eine Eigenschaft eines Objekts anzusprechen, setzen Sie hinter der Objektvariablen, hier `oDokument` einen Punkt, gefolgt von der gewünschten Eigenschaft `DrawPages()`.

Wie im Fall von DrawPages(0) kann eine Eigenschaft selbst wieder ein Objekt sein, das selbst wiederum Eigenschaften enthält. Eine solche Eigenschaft könnte hier z.B. DrawPage, also eine einzelne Seite sein, die als Objekt z.B. wiederum die Eigenschaft Name hat.

1.3. Die For-Next-Schleife

Einer der großen Vorteile der Programmierung liegt darin, dass Sie nicht jeden Schritt immer wieder explizit anordnen müssen. Andernfalls müssten Sie bei diesem Beispiel-Makro jedes der 35 Rechtecke zuerst erzeugen und dann jeweils die Eigenschaften vergeben.

Die For-Next-Schleife eröffnet Ihnen die Möglichkeit Code, der wiederholt ausgeführt werden soll, nur einmal definieren zu müssen. Dazu legen Sie mit Hilfe einer Bedingung fest, wie oft die Schleife durchlaufen werden soll. Im Beispiel handelt es sich um zwei ineinander verschachtelte Schleifen.

```
For iZeile = 1 To 7
  For iSpalte = 1 To 5 Next iSpalte
  (....)
Next iSpalte
Next iZeile
```

- In diesem Fall wird der Code zwischen For und Next ausgeführt, solange iZeile einen Wert zwischen 1 und 7 besitzt. Dabei wird zu Beginn der Schleife der Wert der Variablen, hier iZeile, automatisch mit 1 belegt und bei Erreichen von Next automatisch um 1 erhöht. Eine solche Variable wird deshalb auch als Schleifenzähler bezeichnet.

Ihren besonderen Nutzwert erhält eine Schleife dadurch, dass nicht immer genau derselbe Code ausgeführt werden muss. Im Beispiel erweist es sich für die unterschiedliche Gestaltung der Füllfarben als praktisch, dass der Schleifenzähler in jedem Durchlauf einen anderen Wert hat, hier jeweils um 1 erhöht.

Note:

Benötigen Sie in einer Schleife andere Zählsschritte oder auch einmal ein Abwärts-Zählen, geben Sie dies über die Step-Anweisung an, wie z.B. For i = 300 To 100 Step -3. In diesem Fall wird in 100 Schleifendurchläufen i um jeweils 3 verringert.

- Im Beispiel wird die äußere Schleife 7-mal durchlaufen. Bei jedem Durchgang wird die innere Schleife gestartet, die dann jeweils 5-mal durchlaufen wird, d.h. insgesamt also 35-mal. Auf diese Weise werden 7 Zeilen mit jeweils 5 Spalten erzeugt.

- Nur in der inneren Schleife werden die Rechtecke erzeugt und ihre Eigenschaften mit

Werten versehen.

1.4. Die Gestaltung der Rechtecke

Die genaue Reihenfolge, in der Sie innerhalb der Schleife die Eigenschaften des Rechtecks mit Werten belegen, ist unerheblich. Allerdings muss das jeweilige Objekt bereits existieren, dem Sie Eigenschaften zuweisen. Entsprechend können die bereits vor der Schleife erzeugten Structs Punkt und Groesse jetzt problemlos mit Werten für Koordinaten bzw. Breite und Höhe belegt werden.

```
Punkt.x = iSpalte * 3000
Punkt.y = iZeile * 3000
Groesse.Width = 2000
Groesse.Height = 2000
```

- Dabei gilt als Bezugspunkt für das Koordinatensystem standardmäßig die linke, obere Ecke der Zeichenseite und als Bezugspunkt für das Rechteck dessen linke, obere Ecke.
- Die Werte geben Sie, ebenso wie bei der Groesse, in Hundertstelmillimetern an, die hier verwendeten Werte 3000 und 2000 entsprechen also 3 bzw. 2 Zentimetern. Wie bereits erwähnt, sind x und y bzw. Width und Height dabei jeweils Eigenschaften der Structs. Aufgrund der hier identischen Werte für Breite und Höhe wird das anschließend erzeugte Rechteck ein Quadrat.

Die Abstände zwischen den Formen kommen dadurch zustande, dass die x- bzw. Y-Werte jeweils Vielfaches von 3000 Hundertstelmillimetern betragen, während die Breite bzw. Höhe der Rechtecke jeweils 2000 Hundertstelmillimeter ist. Somit ergibt sich jeweils ein Abstand von 1000 Hundertstelmillimetern.

Das Objekt oRechteck muss bereits existieren, bevor die zuvor mit Werten belegten Structs Groesse und Punkt den kombinierten Eigenschaften Size und Position zugewiesen werden können. Dasselbe gilt logischerweise für die Fülleigenschaft FillColor.

```
oRechteck =
oDokument.createInstance("com.sun.star.drawing.RectangleShape")
oRechteck.Size = Groesse
oRechteck.Position = Punkt
'oRechteck.FillColor = RGB(255/iSpalte,iSpalte * 10,iZeile * 35)
'Rot
'oRechteck.FillColor = RGB(iSpalte * 10,255/iSpalte,iZeile * 35)
'Grün
oRechteck.FillColor = RGB(iSpalte * 10,iZeile * 35,255/iSpalte)
'Blau
```

Erzeugt wird das Objekt oRechteck etwas anders als die bereits bei der Deklaration instanziierten Objekte Punkt und Groesse. Es handelt sich diesmal um kein abstraktes

Objekt, sondern um eine Zeichenform des neu erzeugten Dokuments oDokument.

- In diesem Fall wird mit der Methode `createInstance()` des Dokument-Objekts eine Rechteck-Zeichenform nach dem Muster der UNO-Klasse `RectangleShape` erzeugt. Dazu wird die Zeichenfolge `"com.sun.star.drawing.RectangleShape"` als String-Argument der Methode übergeben.

Note:

`com.sun.star.drawing` bezeichnet wieder den Paketpfad für den Ort dieser Klasse. Hier enthält das Paket `star` das Paket `drawing` und dieses wiederum die Klasse `RectangleShape`.

1.4.1. Füllfarbe festlegen

Ist das Objekt `oRechteck` erzeugt, legen Sie die Füllfarbeneigenschaft `FillColor` fest. Nachdem OpenOffice.org-BASIC die `RGB()`-Methode zur Angabe von RGB (Rot-,Grün,-Blau)-Werten bereitstellt, können Sie diese Werte direkt an die `FillColor`-Eigenschaft übergeben.

Im Beispiel sollen die Rechtecke nicht alle gleichfarbig sein. Deshalb werden die Farbwerte mit Hilfe von Variablen variiert, hier mit den Schleifenzählern `iSpalte` und `iZeile`.

- Beim ersten Durchlauf beider Schleifen sähen die RGB-Werte folgendermaßen aus: `RGB(1*10, 1*35, 255/1)`. Das `*` steht für Multiplikation, der `/` für Division, also ergeben sich in diesem Fall die Werte RGB (10, 35, 255).

- Entsprechend würden beim ersten Durchlauf der äußeren, aber zweiten Durchlauf der inneren Schleife folgende Werte entstehen: `RGB(2*10, 1*35, 255/2)`. Ergeben sich dabei nicht ganzzahlige Rechenergebnissen (wie hier z.B. 127,5) rundet die RGB-Funktion automatisch.

Der Kommentar am Ende der Codezeile 'Blau' beschreibt die vorherrschende Farbe der Füllung.

Wie Sie vielleicht bereits bemerkt haben, befinden sich noch zwei weitere, auskommentierte Variationen dieser Codezeile im Listing. Möchten Sie das Ergebnis auf Basis von mehr rot oder grün sehen, kommentieren Sie die Blau-Zeile aus und entfernen stattdessen entweder das Kommentarzeichen vor der Zeile Rot oder Grün.

Die blaue und die rote Variante desselben Makros

1.5. Die Methoden Wait() und add()

Die im Beispiel zwei Mal eingeschobene BASIC-Methode Wait() hält das Ausführen des Makros um so viele Millisekunden wie jeweils angegeben auf, also 100 bzw. 1000. Dies dient nur einer gefälligeren Darstellung und ist nicht zwingend notwendig.

```
...Wait(100)
  oSeite.add(oRechteck)
Next iSpalte
Next iZeile
Wait(1000)
MsgBox "Fertig !!!"
```

- Ist die erste Wartezeit um, wird das Rechteck auf Ihrer Seite platziert, was durch die Methode add() der Zeichenseite (DrawPage) oSeite geschieht (Zeile 30). Im Gegensatz zur BASIC-Methode Wait() handelt es sich bei add() wieder um eine Methode, die fest einem Objekt, hier also DrawPage zugeordnet ist. Generell werden mit der Methode add() einer Seite weitere Komponenten hinzugefügt.
- Die zweite Wartezeit legt das Makro ein, nachdem alle Formen erzeugt sind, also nach der gewünschten Anzahl der Schleifendurchläufe. Nach der kurzen Pause erscheint jetzt die Message-Box mit der Abschlussmeldung „Fertig!“.