

HANSER

René Martin

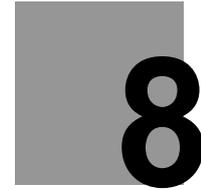
Microsoft Visio 2007-Programmierung

ISBN-10: 3-446-41084-8

ISBN-13: 978-3-446-41084-8

Leseprobe

Weitere Informationen oder Bestellungen unter
<http://www.hanser.de/978-3-446-41084-8>
sowie im Buchhandel



8 Bewegung in Visio

Sicherlich ist Visio nicht das beste Programm, um Bewegung im Sinne einer Animation oder eines Computerspiels zu erzeugen. Wenn es Ihnen um schnelle Abläufe geht, um frei definierte Pfade, um Bewegung im Internet, dann sind Programme wie Flash oder andere sicherlich besser geeignet. Um einen zentralen Nachteil von Visio zu benennen: Die VBA-Methode `DoEvents` baut den gesamten Bildschirm auf. Es ist in Visio leider nicht möglich, nur einen Teil des Bildschirms neu zu zeichnen. Jedoch: Um Änderungen von Zuständen in technischen oder naturwissenschaftlichen Zeichnungen zu simulieren, tut Visio gute Dienste.

8.1 Autorennen

An einem kleinen Beispiel soll der Aufbau eines Programms erläutert werden. Es geht dabei um ein Shape, das auf dem Zeichenblatt verschoben wird. Mithilfe der Vorlage „Wegbeschreibung 3D“ wird eine Straße gezeichnet. Dazu wird das Straßen-Shape auf ein Zeichenblatt platziert. Mit gedrückter [STRG]-Taste wird es verschoben und dupliziert. Nun kann dieser Befehl wiederholt werden, sodass eine Straße entsteht. Auf der Straße wird ein Auto bereitgestellt.

Im Menü `FORMAT | OBJEKTDATEN` wird der Name „Auto1“ festgelegt. Seine Lage wird über das Größe- und Position-Fenster bestimmt: Es befindet sich bei (10 mm / 240 mm). Im Code-Editor wird ein neues Modul angelegt – dort wird eine Prozedur erzeugt, mit der auf das Shape zugegriffen wird:

```
Sub Fahrt()  
  
    Dim pagSeite As Page  
  
    Dim shpAuto1 As Shape  
  
        Set pagSeite = Application.ActivePage  
  
    Set shpAuto1 = pagSeite.Shapes("Auto1")
```

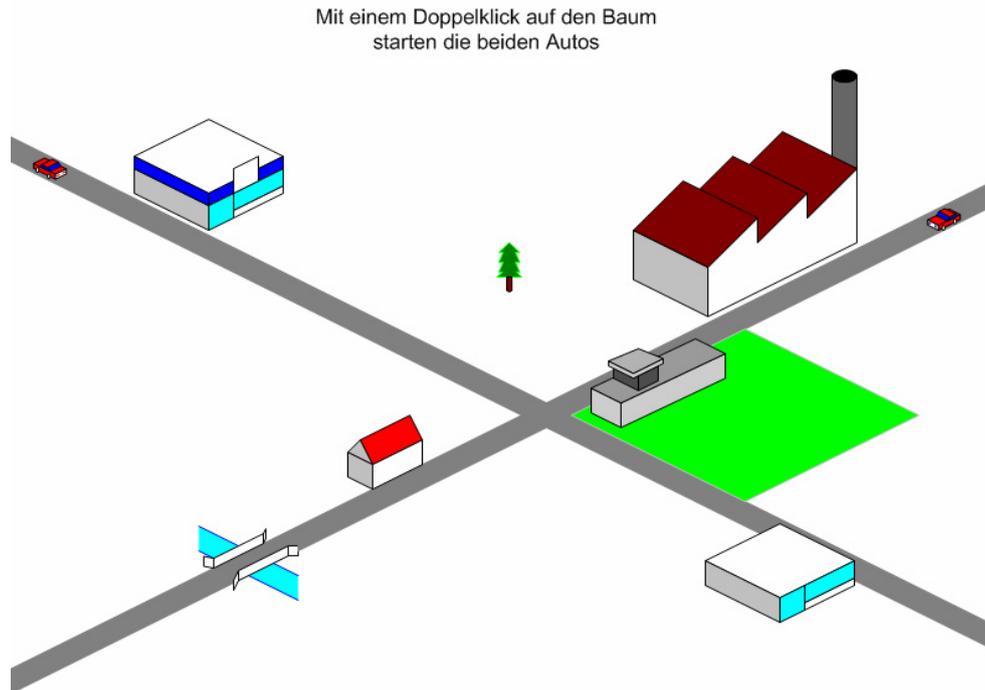


Abbildung 8.1 Eine Straße mit Auto

Um es zum Fahren zu bringen, müssen die beiden Koordinaten „PinX“ und „PinY“ verändert werden. Da ein Straßen-Shape eine Breite von 20 mm und eine Höhe von 15 mm besitzt, beträgt die Bewegung nach rechts und unten ein Verhältnis 4 : 3. Also könnte gestartet werden:

```
Sub Fahrt()  
  
    Dim pagSeite As Page  
  
    Dim shpAuto1 As Shape  
  
    Dim i As Integer  
  
    Set pagSeite = Application.ActivePage  
  
    Set shpAuto1 = pagSeite.Shapes("Auto1")  
  
    For i = 0 To 100  
  
        shpAuto1.Cells("PinX").Result("mm") = 2 * i + 10  
  
        shpAuto1.Cells("PinY").Result("mm") = 240 - i  
  
    End For  
End Sub
```

```
DoEvents
```

```
Next
```

Der Befehl `DoEvents` ist nötig, damit der Bildschirm sich erneut aufbaut. Wird er weggelassen, so wird zwar das Makro ausgeführt, aber es ist nichts sichtbar. Damit das Auto am Ende wieder zum Ausgangspunkt zurückkehrt, wird es auf die Startposition gesetzt:

```
shpAuto1.Cells("PinX").Result("mm") = 10
```

```
shpAuto1.Cells("PinYY").Result("mm") = 240
```

Wird die Prozedur aus VBA gestartet, so können Sie leider nichts sehen. Deshalb muss das Makro entweder aus Visio über EXTRAS | MAKRO | BASRENNEN | FAHRT oder über EXTRAS | MAKROS | MAKROS aufgerufen werden. Oder Sie binden es an eine Schaltfläche oder an ein Shape. Neben die Straße wird ein Baum-Shape platziert. Damit das Makro per Doppelklick startet, muss die Datei gespeichert werden und sollte das Modul einen vernünftigen Namen erhalten. Dann kann im Menü FORMAT | VERHALTEN | DOPPELKLICKEN das Makro an das entsprechende Shape gebunden werden. Dies ist wichtig, denn wenn die Datei nicht gespeichert ist, dann lautet die Prozedur beispielsweise:

```
AutoRennen.basRennen.Fahrt
```

Ein Speichern der Datei oder ein Umbenennen des Moduls führt dazu, dass das Doppelklickverhalten nicht mehr funktioniert. Dann müsste erneut das korrekte Makro eingestellt werden. Deshalb ist es geschickter, zu Beginn die korrekten Namen zu vergeben.

Das Beispiel soll noch um ein zweites Fahrzeug („Auto2“) erweitert werden, das von rechts oben nach links unten fährt. Der ganze Code sieht dann wie folgt aus:

```
Sub Fahrt()
    Dim pagSeite As Page
    Dim shpAuto1 As Shape
    Dim shpAuto2 As Shape
    Dim i As Integer

    Set pagSeite = Application.ActivePage
    Set shpAuto1 = pagSeite.Shapes("Auto1")
    Set shpAuto2 = pagSeite.Shapes("Auto2")

    For i = 0 To 100
        shpAuto1.Cells("PinX").Result("mm") = 2 * i + 10
```

```
shpAuto1.Cells("PinY").Result("mm") = 240 - i  
shpAuto2.Cells("PinX").Result("mm") = 190 - 2 * i  
shpAuto2.Cells("PinY").Result("mm") = 230 - i  
  
DoEvents  
  
Next  
  
shpAuto1.Cells("PinX").Result("mm") = 10  
shpAuto1.Cells("PinY").Result("mm") = 240  
shpAuto2.Cells("PinX").Result("mm") = 190  
shpAuto2.Cells("PinY").Result("mm") = 230  
  
End Sub
```

Wenn Ihnen die Fahrt zu schnell ist, dann könnten Sie die Variable `i` von Typ `Double` deklarieren und die Schleife um eine kleinere Schrittweise erhöhen:

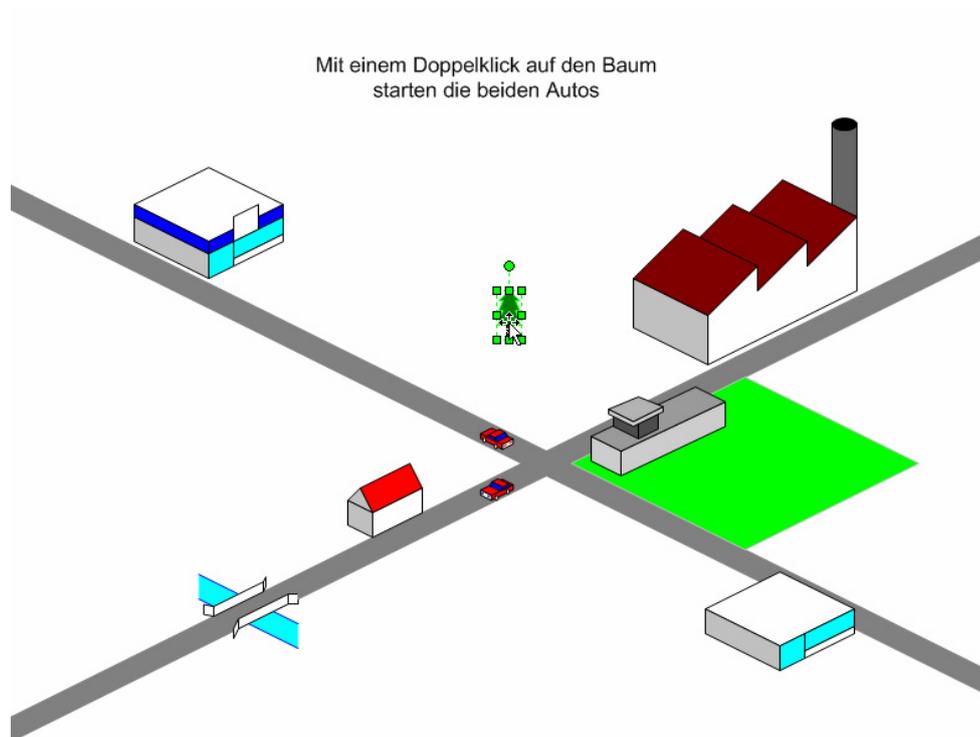


Abbildung 8.2 Die beiden Autos rasen über die Straße ($i = 45$).

```
Dim i As Double
```

```
...
```

```
For i = 0 To 100 Step 0.4
```

Analog kann man das Auto entlang einer „gekrümmten“ Linie verschieben:

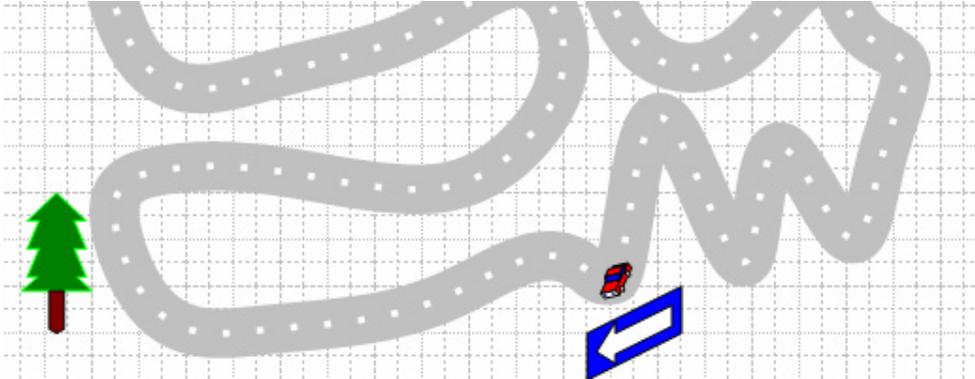


Abbildung 8.3 Das Auto folgt einer Kurve.

8.2 Der Viertaktmotor

Auf einem neuen Zeichenblatt wird ein Objekt gezeichnet, das aus folgenden sechs Teilen besteht, wie Sie in Abbildung 8.4 sehen.

Da die „Kurbel“ immer unterhalb des „Deckels“ liegt, werden im ShapeSheet der Kurbel folgende Werte eingetragen:

Der „LocPinX“ befindet sich immer in der Mitte:

```
=GUARD (Width*0,5)
```

Der „Pin“ wird ans obere Ende platziert, also sitzt der „LocPinY“ auf:

```
=GUARD (Height)
```

Die x-Koordinate stimmt mit der des „Deckels“ überein – folglich lautet „DrehbezX“:

```
=GUARD (Deckel!PinX)
```

Und die y-Koordinate liegt in einem bestimmten Abstand zum „Deckel“. Also könnte die Formel beispielsweise lauten:

```
=GUARD (Deckel!PinY-7 mm.)
```

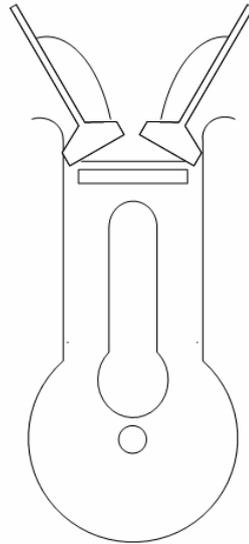


Abbildung 8.4 Der konstruierte Viertaktmotor

Damit wird gewährleistet, dass Bewegungen des „Deckels“ auf die „Kurbel“ übertragen werden. Nun kann die Programmierung beginnen. Hinter einer Befehlsschaltfläche liegt folgender Code:

```
Option Explicit

Private Sub cmdStart_Click()

    Dim i As Integer, j As Integer

    For j = 1 To 2

        For i = 0 To 90

            ActivePage.Shapes("Deckel").Cells("PinY").Formula = _

                "=" & Chr(34) & "-i / 3 + 110 & " mm" & Chr(34)

            DoEvents

        Next i

    Next j

End Sub
```

```

For i = 90 To 0 Step -1

ActivePage.Shapes("Deckel").Cells("PinY").Formula = _

    "=" & Chr(34) & -i / 3 + 110 & "mm" & Chr(34)

DoEvents

Next i

Next j

End Sub

```

Die äußere Schleife wird zweimal ausgeführt, der Deckel dabei nach unten gesenkt und nach oben gehoben. Dies geschieht in der inneren Schleife, die von 0 bis 90 hoch- oder von 90 bis 0 heruntergezählt wird. Nun soll sich die „Kurbel“ bewegen. Da die „Kurbel“ nach links oder rechts bewegt wird, muss über eine Verzweigung abgefragt werden, in welche Richtung der Ausschlag geht. Die rechte Hälfte sieht dann wie folgt aus:

```

For i = 0 To 90

ActivePage.Shapes("Deckel").Cells("PinY").Formula = _

    "=" & Chr(34) & -i / 3 + 110 & " mm" & Chr(34)

If i < 45 Then

ActivePage.Shapes("Kurbel").Cells("Winkel").Formula = _

    "=" & Chr(34) & Sin(i * 2 * 3.14159 / 180) * 20 & _

    " grad" & Chr(34)

Else

ActivePage.Shapes("Kurbel").Cells("Winkel").Formula = _

    "=" & Chr(34) & Cos((i - 45) * 2 * 3.14159 / 180) * _

    20 & " grad" & Chr(34)

End If

DoEvents

Next i

```

Analog auf der anderen Seite: Dort wird der Winkel negativ genommen. Nun soll sich noch das Ventil „Einlass“ in Phase 1 a nach unten bewegen, in Phase 1 b nach oben. Auch

hier muss abgefragt werden, wo sich der „Deckel“, oder genauer die Zählervariable i, befindet. Der erste Teil sieht nun wie folgt aus:

```
Private Sub cmdStart_Click()

    Dim i As Integer

    Dim j As Integer

    Dim dblXEInBeginn As Double, dblyBeginn As Double

    Dim dblXAusBeginn As Double

    dblXEInBeginn = Cdbl(Mid(ActivePage.Shapes("Einlass").Cells("PinX"). _
        .Formula, 1, InStr(1, _
        ActivePage.Shapes("Einlass").Cells("PinX").Formula, " ")))

    dblyBeginn = Cdbl(Mid(ActivePage.Shapes("Einlass").Cells("PinY") _
        .Formula, 1, InStr(1, _
        ActivePage.Shapes("Einlass").Cells("PinY").Formula, " ")))

    dblXAusBeginn = Cdbl(Mid(ActivePage.Shapes("Auslass").Cells("PinX"). _
        Formula, 1, InStr(1, _
        ActivePage.Shapes("Auslass").Cells("PinX").Formula, " ")))

    For j = 1 To 2

        For i = 0 To 90

            ActivePage.Shapes("Deckel").Cells("PinY").Formula = _
                "=" & Chr(34) & "-i / 3 + 110 & " mm" & Chr(34)

            If i < 45 Then

                ActivePage.Shapes("Kurbel").Cells("Winkel") _
                    .Formula = _
                        "=" & Chr(34) & Sin(i * 2 * 3.14159 / 180) * 20 _
                            & " grad" & Chr(34)

                If j = 1 Then
```

```
ActivePage.Shapes("Einlass").Cells("PinX") _  
    .Formula = "=" & dblXEInBeginn - i / 10 & " mm"  
ActivePage.Shapes("Einlass").Cells _  
    ("PinY").Formula = "=" & dblyBeginn - i / 5 & " mm"  
End If  
Else  
ActivePage.Shapes("Kurbel").Cells("Angle"). _  
Formula = "=" & Chr(34) & Cos((i - 45) * _  
2 * 3.14159 / 180) * 20 & " grad" & Chr(34)  
If j = 1 Then  
ActivePage.Shapes("Einlass").Cells("PinX"). _  
Formula = "=" & dblXEInBeginn - (90 - i) / 10 & " mm"  
ActivePage.Shapes("Einlass").Cells("PinY"). _  
Formula = "=" & dblyBeginn - (90 - i) / 5 & " mm"  
End If  
End If  
DoEvents  
Next i
```

Das Ergebnis kann sich sehen lassen:

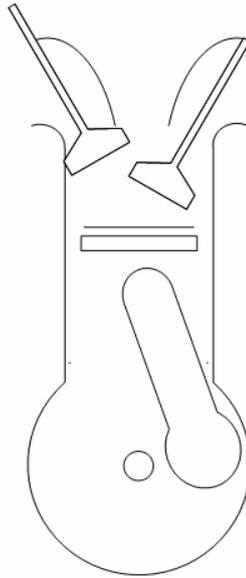


Abbildung 8.5 Ein Snapshot bei $i = 39$

Nun kann noch die Explosion dargestellt werden. Dazu wird ein weiteres Objekt („Flamme“) eingefügt, das dreimal in die Farben Weiß, Gelb und Rot wechselt.

```
If j = 1 Then
  For z = 0 To 3
    With ActivePage.Shapes("Flamme") _
      .Cells("FillForegnd")
      .Formula = "=5":      DoEvents
      .Formula = "=2":      DoEvents
      .Formula = "=1":      DoEvents
    End With
  Next
End If
```

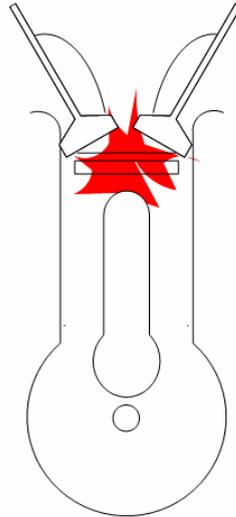


Abbildung 8.6 Die Explosion

Und schließlich kann man noch die Phasen in einem Textfeld beschreiben:

```
[...]
For j = 1 To 2

    If j = 1 Then
        ActivePage.Shapes("Beschreibung").Text = _
            "Viertaktmotor Phase 1" & vbCrLf & "Ansaugen"
    Else
        ActivePage.Shapes("Beschreibung").Text = _
            "Viertaktmotor Phase 3" & vbCrLf & "Arbeiten"
    End If

    For i = 0 To 90
[...]
```

**Viertaktmotor
Phase 2
Verdichten**

Start

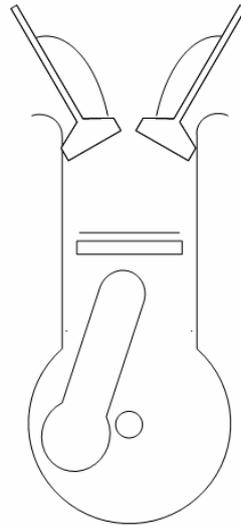


Abbildung 8.7 Eine Momentaufnahme des fertigen, laufenden Motors bei $j = 1$ und $i = 58$

8.3 Planetenumlaufbahn

Werden mehrere Objekte auf dem Zeichenblatt bewegt, die zueinander in einer bestimmten Konstellation stehen (beispielsweise ein Text, der unabhängig vom Shape in einem bestimmten Abstand zu diesem stehen soll), dann kann dies ohne Programmierung im ShapeSheet des entsprechenden Shapes festgelegt werden. Das Objekt, auf das Bezug genommen wird, wird per Namen angesprochen, und durch ein Ausrufezeichen wird der Name der Zelle, auf die referiert wird, festgelegt, beispielsweise:

```
=Erde!PinY-5 mm
```

Damit der Anwender nicht versehentlich das Shape verschieben kann, kann die Funktion GUARD verwendet werden:

```
=GUARD(Erde!PinY-5 mm)
```

Mit ein wenig elementarer Algebra beziehungsweise Trigonometrie können nun leicht Positionen verändert werden, wie beispielsweise bei den Umlaufbahnen der Planeten.

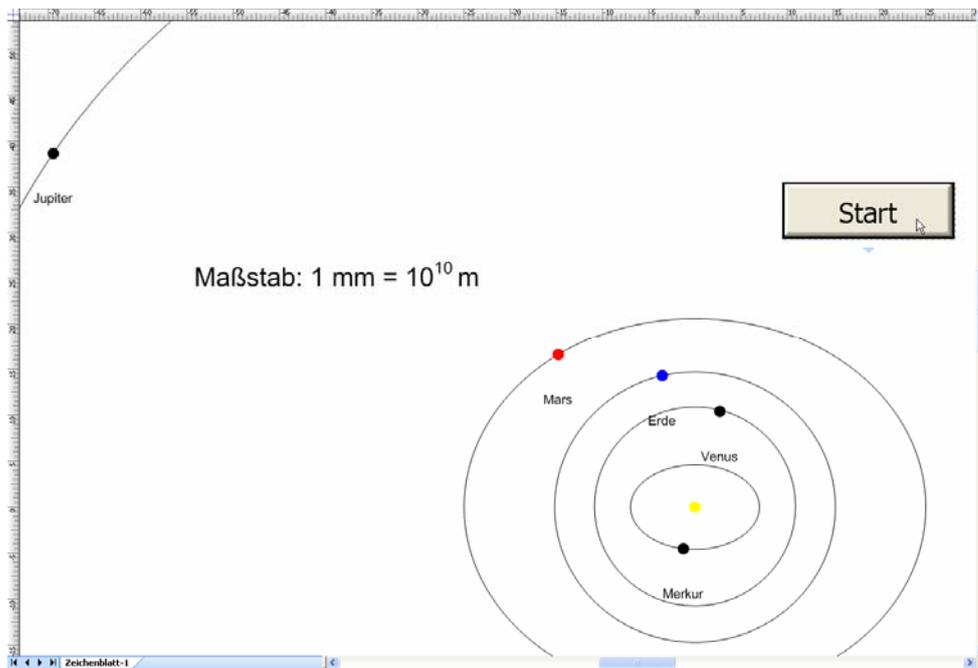


Abbildung 8.8 Die Planetenumlaufbahnen können in Visio leicht simuliert werden.

8.4 Fazit

Mit wenig Aufwand kann in Visio eine Zeichnung erstellt werden. Mit geringen VBA-Kenntnissen kann auf die Shapes zugegriffen werden. Sie können in ihrer Position und in ihrem Aussehen verändert werden. Parallel dazu können Texte platziert und ausgeblendet werden. Damit können leicht und schnell dynamische Verläufe aus den Bereichen Technik, Pneumatik und Hydraulik, Kybernetik, Naturwissenschaft, Astronomie ... dargestellt werden.

