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

3.10 Die Layer

Anders als die Formatvorlagen sind die verwendeten Layer Teil des Seiten-Objekts. Deshalb muss über `Page` auf sie zugegriffen werden. Die Techniken sind bereits bekannt. Im folgenden Beispiel werden alle Layer angezeigt:

```
Sub LayerAnzeigen()

   Dim i As Integer, strListe As String

   For i = 1 To ActivePage.Layers.Count

      strListe = strListe & _

         vbCr & ActivePage.Layers(i).Name

   Next

   MsgBox strListe

End Sub
```

Oder über einen Objektzugriff:

```
Sub LayerAnzeigen2()

   Dim layLayer As Layer

   Dim strListe As String

   For Each layLayer In ActivePage.Layers

      strListe = strListe & vbCr & layLayer.Name

   Next

   MsgBox strListe

End Sub
```

Mit der Methode `Add` wird ein neuer Layer erzeugt:

```
   ActivePage.Layers.Add "MeinLayer01"
```

Mit der Methode `Delete` kann der Layer wieder gelöscht werden. Und die Gestaltung erfolgt ebenso über den Zellzugriff.

Achtung

Jedoch: So einfach wie bei der Sammlung Pages oder Documents verhält es sich bei Layers nicht. Dies soll auf den folgenden Seiten beschrieben werden.

Layer sind in Visio ein wichtiges Mittel, um Shapes zu kennzeichnen. Über den Menüpunkt FORMAT | LAYER kann ein Shape auf einen oder mehrere Layer gelegt werden.

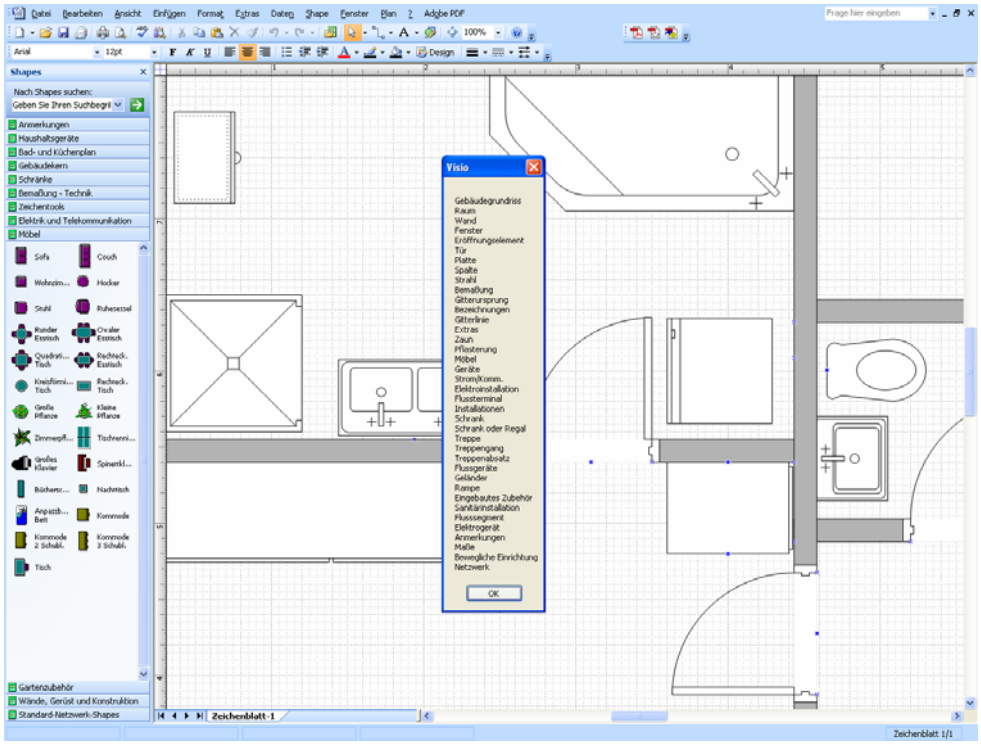

Abbildung 3.6 Die Liste der Layer

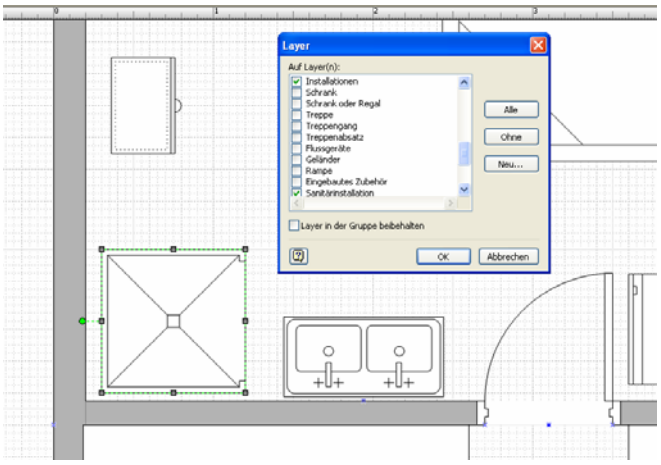

Abbildung 3.7 Ein Shape kann auf mehreren Layern liegen.

Über ANSICHT | LAYEREIGENSCHAFTEN können alle Layer eingesehen werden. Wird ein Shape auf einen Layer gelegt, kann es anschließend in eine Schablone gezogen werden und wird so zu einem Master-Shape. Jede neue Zeichnung erhält dann automatisch diesen

Layer zugewiesen, wenn das Shape aus der Schablone auf die Zeichnung gezogen wird. Achtung: Layer gelten nur für ein Zeichenblatt, sind also nicht in der gesamten Datei vorhanden.

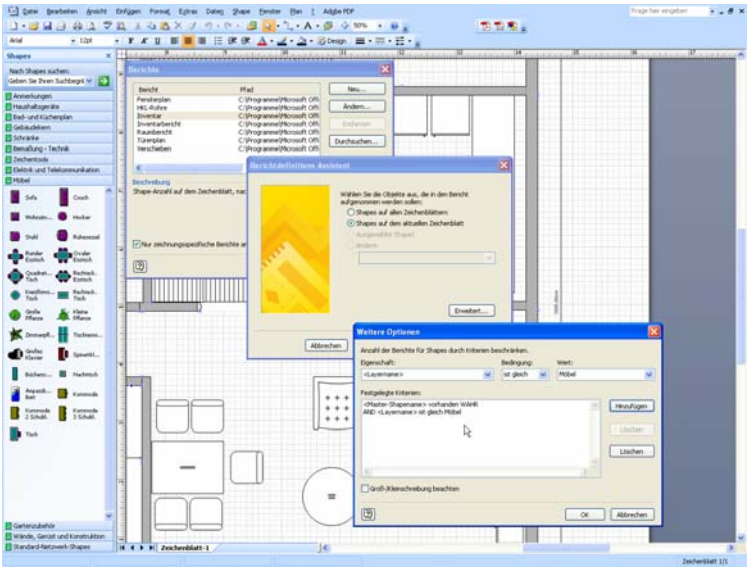

Abbildung 3.8 Viele Assistenten verwenden Layer (hier: DATEN | BERICHTE).

Viele Assistenten verwenden die Layer zur Auswahl bestimmter Shapes: Alle Shapes, die auf einem Layer liegen, können markiert werden (BEARBEITEN | AUSWAHL NACH TYP), in Berichte und Datenbanken können Informationen bestimmter Shapes geschrieben werden, die auf eigenen Layern liegen.

Doch die einfache Verwendbarkeit und der universelle Nutzen haben ihre Tücken:

3.10.1 Die Anzahl der Shapes pro Layer

Über das Menü ANSICHT | LAYEREIGENSCHAFTEN kann die Anzahl der Shapes angezeigt werden, die auf einem Layer liegen. Jedoch ist diese Zahl nicht immer korrekt: Wurden mehrere Shapes zu einer Gruppe zusammengefügt, dann wird jedes Kindelement plus die Gruppe (die ja schließlich auch ein Shape ist) gezählt. Umgekehrt kann ein Shape auf mehreren Layern liegen, sodass die Summe der Shapes nicht mit der angezeigten Summe übereinstimmen muss.

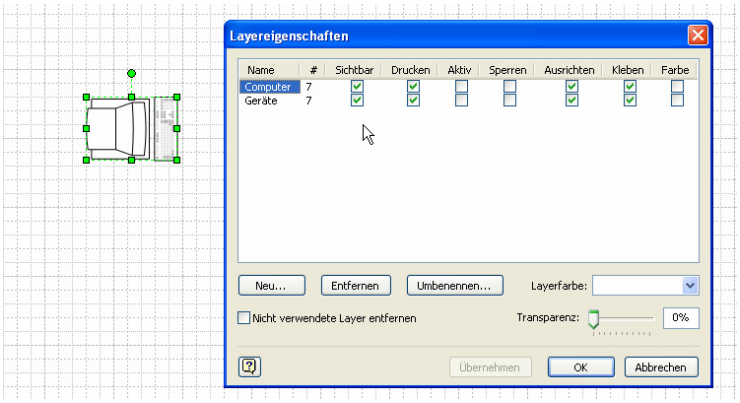

Abbildung 3.9 Die Anzahl der Shapes pro Layer ist nur bedingt brauchbar – hier: ein Shape, zwei Layer, # = 14.

3.10.2 Das ShapeSheet

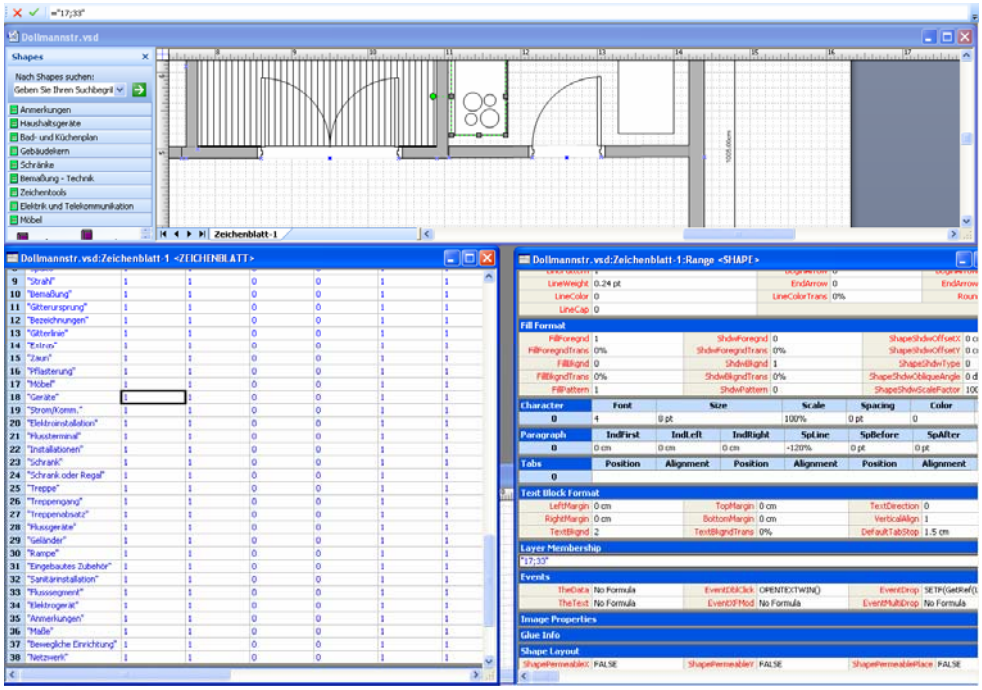

Abbildung 3.10 Die Nummer der Layer des Shapes + 1 ergeben die Nummer des Layers des Zeichenblattes (hier: bewegliche Geräte und Elektrogerät).

Alle verwendeten Layer werden im ShapeSheet des Zeichenblattes angezeigt – im Abschnitt „Layer“. Dabei werden neue Layer in der Liste unten eingefügt – die Reihenfolge ist nicht alphabetisch sortiert – anders als im Dialog „Layereigenschaften“.

Der Abschnitt „Layerzugehörigkeit“ der einzelnen Shapes greift nun auf diese Layer-Liste des Zeichenblatts zurück und listet die Layer auf – leider nicht namentlich, sondern per Nummern, zu deren Zahl 1 addiert werden muss:

```
="17"
```

steht also für die Zugehörigkeit zum siebten Layer des Zeichenblattes,

```
="17;33"
```

weist darauf hin, dass sich das Shape auf den Layern Nummer 18 und 34 befindet.

3.10.3 VBA

Noch komplizierter wird das Ganze, wenn Sie per Programmierung auf alle Layer des Zeichenblattes oder eines Shapes zugreifen. Das Objekt „Layer“ ist schnell gefunden, jedoch ist seine Verwendung nicht trivial. Zwar funktioniert:

```
Dim vsLayer As Layer

For Each vsLayer In ActivePage.Layers

   MsgBox vsLayer.Name

Next
```

Jedoch erstaunt, dass ein Shape nicht über die Sammlung „Layers“ verfügt. Man muss alle Layer mit einer Zählerschleife durchlaufen:

```
Dim i As Integer

With ActivePage.Shapes("Sheet.1")

   For i = 1 To .LayerCount

      MsgBox .Layer(i).Name

   Next

End With
```

Hier fällt auf, dass die Eigenschaft Layer eine Sammlung ist – auch ohne Pluralendung s, wie man vermuten würde. Darüber kann auch eine korrekte Zuweisung vorgenommen werden:

```
Set vsLayer = ActivePage.Shapes("Sheet.1").Layer(i)
```

Wird ein neuer Layer erstellt, muss überprüft werden, ob er bereits existiert. Da leider keine Funktion `LayersExists` oder Ähnliches zur Verfügung steht, muss mit einer Schleife gearbeitet werden:

```
Dim strStandort As String

Dim fLayer As Boolean

Dim vsLayer As Layer

strStandort = "München"

With Application.ActivePage

   For i = 1 To .Layers.Count

      If .Layers(i).Name = strStandort Then

         Set vsLayer = .Layers(i)

         fLayer = True

         Exit For

      End If

   Next

   If fLayer = False Then

      Set vsLayer = .Layers.Add(strStandort)

      ' -- überprüfe, ob es den Layer gibt

      ' - falls nicht, dann erzeuge ihn

   End If

End With
```

Obwohl die Zellnamen des ShapeSheets direkt mit ihrem deutschen oder englischen Namen angesprochen werden können (beispielsweise Breite und Höhe beziehungsweise Width und Height), stellt die Zelle Layerzugehörigkeit auch hier wieder eine Ausnahme dar. Der Zugriff erfolgt über die Zelle `LayerMember`), beispielsweise um das Shape von allen Layern zu entfernen:

```
Dim vsShape As Shape

Set vsShape = ActivePage.Shapes("Sheet.1")

vsShape.Cells("LayerMember").Formula = ""
```

Soll dieses Shape nun auf einen neuen Layer gelegt werden, dann wird die Methode `Add` nicht auf das Shape, sondern auf den Layer angewandt:

```
vsLayer.Add vsShape, 0
```

Der zweite Parameter `fPresMems` gibt an, ob bei Gruppen die Kindelemente ausgeschlossen werden sollen oder nicht. Analog kann mit der Methode `Remove` ein Layer eines Shapes gelöscht werden:

```
vsLayer.Remove vsShape, 0
```

Die Methode `Delete` dient zum Löschen eines Layers einer Seite. Sie verlangt einen Parameter `fDeleteShapes`. Ist er 1 oder True, werden alle Shapes, die noch auf diesem Layer liegen, mit dem Layer gelöscht, bei 0 (`False`) bleiben sie erhalten.

Mit diesem Wissen kann nun beim Verschieben von einem vorhandenen Shape und beim Erzeugen von neuen Shapes von einem übergeordneten „Standort“ der Name des Standortes ausgelesen und das Shape auf diesen Layer gelegt werden. Für Daten (Datenfelder oder benutzerdefinierte Eigenschaften) wäre dieser Aufwand nicht ganz so schwierig:

```
Option Explicit

Dim WithEvents pagobj As Visio.Page

Private m_shpObj As Shape


Private Sub Document_RunModeEntered(ByVal doc As Visio.IVDocument)

    Set pagobj = Visio.ActivePage

    ' -- beim Öffnen greife auf das Zeichenblatt zu

End Sub


Private Sub Document_ShapeAdded(ByVal Shape As Visio.IVShape)

   ' -- ein neues Shape wird generiert, aber kein Standort

   If Left(Shape.Name, 8) <> "Standort" Then

      Set m_shpObj = Shape

      Call Standort_Test

   End If

End Sub

Private Sub pagobj_CellChanged(ByVal Cell As IVCell)

   On Error Resume Next

   If Left(Cell.Name, 3) = "Pin" Or Left(Cell.Name, 4) = "Dreh" Then
```

```
    ' -- wird irgendein "altes" Shape verschoben
       If Left(Cell.Shape.Name, 8) <> "Standort" Then
          Set m_shpObj = Cell.Shape
          ' -- greife auf das Shape zu
          Call Standort_Test
       End If
    End If
End Sub

Private Sub Standort_Test()
    Dim vsShapeOnPage As Shape
    Dim dblTolerance As Integer
    Dim intSpatialRelation As VisSpatialRelationCodes
    Dim strSpatialRelation As String
    Dim strStandort As String

    On Error GoTo errHandler
    ' -- die Toleranz
    dblTolerance = 0.01

    For Each vsShapeOnPage In ActivePage.Shapes
        ' -- alle vorhandenen Shapes des Zeichenblattes werden durchlaufen
        If Left(vsShapeOnPage.Name, 8) = "Standort" Then
           If vsShapeOnPage.Name <> m_shpObj.Name Then
           ' -- alle außer dem Shape selbst
             If vsShapeOnPage.CellExists("Prop.Standort", True) Then
                strStandort = vsShapeOnPage.Cells("Prop.Standort").Formula
             Else
```

```
                    strStandort = ""
                End If
                  intSpatialRelation = _
                  vsShapeOnPage.SpatialRelation(m_shpObj, dblTolerance, _
                  visSpatialIncludeHidden)
                  ' -- das Verhältnis zu anderen Shapes

                  Select Case intSpatialRelation
                      Case VisSpatialRelationCodes.visSpatialContain
                         Call Standort_Zuweisen(m_shpObj, strStandort)
                         ' -- innerhalb
                         Exit For
                      Case VisSpatialRelationCodes.visSpatialContainedIn
                         ' -- umfasst
                      Case VisSpatialRelationCodes.visSpatialOverlap
                          Call Standort_Zuweisen(m_shpObj, strStandort)
                          Exit For
                          ' -- überlappt
                      Case VisSpatialRelationCodes.visSpatialTouching
                          ' -- bei Berührung nichts
                      Case Else
                          ' -- nicht in einem anderen Shape
                          Call Standort_Zuweisen(m_shpObj, "")
                  End Select
                End If
            End If
        Next
    errHandler:
```

```
   MsgBox "Es trat ein Fehler auf:" & vbCr & _
      Err.Number & ": " & Err.Description
End Sub

Private Sub Standort_Zuweisen(NeuesShape As Shape, strStandort As String)
   Dim i As Integer
   Dim fLayer As Boolean
   Dim fStandort As Boolean
   Dim vsLayer As Layer

   fLayer = False
   fStandort = False

   If Left(strStandort, 1) = """ Then
      strStandort = Mid(strStandort, 2)
   End If

   If Right(strStandort, 1) = """ Then
      strStandort = Left(strStandort, Len(strStandort) - 1)
   End If


   If NeuesShape.CellExists("Prop.Standort", True) Then
      For i = 1 To NeuesShape.Section(visSectionProp).Count
         If NeuesShape.Section(visSectionProp).Row(i - 1).Name = _
            "Standort" Then
            NeuesShape.Section(visSectionProp).Row(i - 1).Cell(0). _
               Formula = "=""" & strStandort & """
```

```
            fStandort = True
            Exit For
         End If
      Next
   End If
   If fStandort = False Then
      If NeuesShape.SectionExists(visSectionProp, True) = True Then
         For i = 1 To NeuesShape.Section(visSectionProp).Count
            If NeuesShape.Section(visSectionProp).Row(i - 1).Cell(2). _
               Formula = """Standort""" Then
               NeuesShape.Section(visSectionProp).Row(i - 1).Cell(0). _
                  Formula = "=""" & strStandort & """"
               fStandort = True
               Exit For
            End If
         Next i
      End If
   End If
   ' -- überprüfe, ob es ein Datenfeld "Standort gibt"
   ' -- falls ja, dann schreibe den neuen Standort hinein.
   If strStandort <> "" Then
      For i = 1 To Application.ActivePage.Layers.Count
         If Application.ActivePage.Layers(i).Name = strStandort Then
            Set vsLayer = Application.ActivePage.Layers(i)
            fLayer = True
            Exit For
         End If
      Next
```

```
        If fLayer = False Then
           Set vsLayer = Application.ActivePage.Layers.Add(strStandort)
           ' -- überprüfe, ob es den Layer gibt
           ' -- falls nicht, dann erzeuge ihn
        End If
        NeuesShape.Cells("LayerMitglied").Formula = ""
        ' -- lösche alle vorhandenen Layer des Shapes
        vsLayer.Add NeuesShape, 0
        ' -- lege das Shape auf den neuen Layer
    End If
End Sub
```

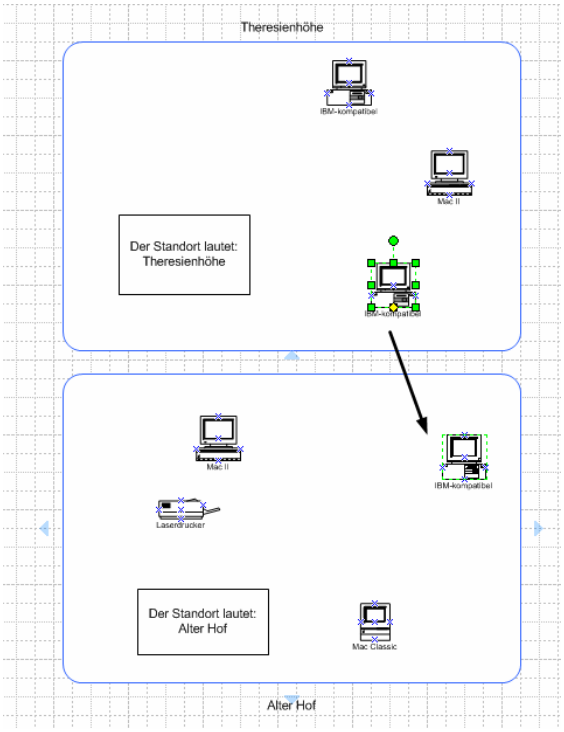

Abbildung 3.11 Beim Verschieben eines Shapes wird es auf einen anderen Layer gelegt. Parallel dazu werden die Datenfelder aktualisiert.