PowerPoint-Programmierung 2
Grundlagen und Überblick 2
Quellcode eingeben 2
Objekte in PowerPoint 3
Änderungen gegenüber PowerPoint 97 und PowerPoint 2000 3
PowerPoint manipulieren 4
Fenstergröße verändern und abfragen 4
Systemdaten ermitteln 5
Fenstertitel ändern 7
Aktive Objekte ermitteln 8
Wichtige Methoden 9
Präsentationen 10
Öffnen und Schließen 10
Erstellen und Löschen von Präsentationen 12
Manipulation von Präsentationen 14
Verwenden von Präsentationen 21

PowerPoint-Programmierung

Mithilfe von PowerPoint können Sie nicht nur Präsentationen erstellen und abspielen lassen, sondern auch interaktive Präsentationen gestalten und Lernprogramme erstellen. Dazu trägt neben neuen Funktionen für Präsentationen - etwa interaktive Schaltflächen zur Steuerung der Präsentation - auch die in PowerPoint integrierte IDE bei, mit deren Hilfe auch PowerPoint programmiert werden kann. Dieses Kapitel soll die grundlegenden Objekte, Methoden und Eigenschaften von PowerPoint vorstellen. Dabei wird jedoch vorausgesetzt, dass Sie sich mit PowerPoint genügend auskennen, um Präsentationen erstellen und ausführen zu können.

Grundlagen und Überblick

Die PowerPoint-Bibliothek gehört mit zu den größten VBA-Bibliotheken des Office-Pakets. Ein gewisser Überblick ist daher unbedingt notwendig, um effizient programmieren zu können. Daher sollen nachfolgend die wichtigsten Objekte und ihre Position in der Objekthierarchie erläutert werden, bevor anhand zahlreicher Beispiele die PowerPoint-Programmierung gezeigt wird.

Quellcode eingeben

Genauso wie in Excel wird auch in PowerPoint der Quellcode in der IDE eingegeben. Diese kann mit Alt + F11 aufgerufen werden und unterscheidet sich auch in der Bedienung nicht. Allerdings sind die Eingabehilfe wie Parameterinfo und Konstantenliste noch nicht für alle Objekte und Methoden von PowerPoint implementiert. Dies irritiert anfangs bei der Eingabe, da man immer das Gefühl hat, einen Fehler gemacht zu haben.

Objekte in PowerPoint

Oberstes Objekt der Hierarchie ist auch hier wieder das Objekt *Application*, das die Anwendung PowerPoint darstellt. Ihm direkt untergeordnet sind die Objekte *Addins*, *Assistant*, *Commandbars*, *DocumentWindows*, *FileSearch*, *FileFind*, *Presentations*, *SlideShowWindows* und *VBE*. Die wichtigsten dieser Objekte sind *Presentations*, das die Liste aller Präsentationen darstellt und *SlideShowWindows*, das die Liste aller Fenster mit aktiven Bildschirmpräsentationen enthält. Die Auflistung *Presentations* enthält *Presentation*-Objekte, welche die einzelne PowerPoint-Präsentation darstellen.

Die wichtigsten seiner untergeordneten Objekte sind *Slides*, die Liste aller vorhandenen Folien einer Präsentation und *PrintOptions*, das die Druckeinstellungen der Präsentation steuert.

Das Objekt *Application* selber hat auch einige wichtige Eigenschaften und Methoden, deren Anwendung durch Beispiele noch verdeutlicht werden wird. Sie können z.B. mit Hilfe der *Caption*-Eigenschaft bestimmen, welcher Text im Fenstertitel von PowerPoint erscheinen soll oder Sie legen mit der *Windowstate*-Eigenschaft die Größe des Fensters fest, in dem PowerPoint läuft.

Änderungen gegenüber PowerPoint 97 und PowerPoint 2000

Neben den Änderungen, die sich alleine durch die Versionsänderung der VBA-Bibliotheken von 5.0 zu 6.0 ergeben, hat sich auch in PowerPoint einiges getan. Sie können nun auch in PowerPoint den Makrovirenschutz aktivieren, VBA-Makros digital signieren oder Makros ganz deaktivieren. Darüber hinaus gibt es auch in PowerPoint 2002 natürlich weitere Änderungen und Ergänzungen. Vor allem Ereignisse gibt es nun in PowerPoint. Diese können jedoch nur zusammen mit der WithEvents-Anweisung und Klassenmodulen genutzt werden. Beide Anweisungen werden im Kapitel 9 "Klassen programmieren und einsetzen" näher erläutert.

Der vorhandene PowerPoint 97/2000 Code läuft aber einwandfrei auch in PowerPoint 2002, so dass hier keine Probleme mit der Kompatibilität auftreten. Die einzige Ausnahme sind Auto_Open() und Auto_Close()-Prozeduren. Sie werden nicht mehr ausgeführt, da stattdessen ja die Ereignisse von PowerPoint verwendet werden können.

PowerPoint manipulieren

Gerade PowerPoint bietet viele Möglichkeiten, mit VBA die Anwendung selbst zu beeinflussen, weil das Objekt *Application* sehr viele untergeordnete Objekte und viele Eigenschaften hat, die eingestellt werden können. In diesem Abschnitt wird nun gezeigt, welche Einstellungen Sie per VBA vornehmen und wie Sie die Anwendung PowerPoint sonst noch beeinflussen können.

Fenstergröße verändern und abfragen

Sie können mithilfe der Eigenschaft *WindowState* des *Application*-Objektes die Fenstergröße manipulieren und abfragen. Das folgende Beispiel reduziert das PowerPoint-Fenster auf Symbolgröße und legt es unten in der Statusleiste von Windows ab.

```
Sub Fenstergröße()
  Application.WindowState = ppWindowMinimized
End Sub
```

Als Konstanten für *WindowState* stehen folgende Werte zur Verfügung:

Konstante	Wert	Beschreibung
ppWindowMaximized	3	Vollbildmodus
ppWindowMinimized	2	Symbolgröße
ppWindowNormal	1	Weder maximiert noch minimiert

Tab. 1: Konstanten der *WindowsState*-Eigenschaft

Hinweis: Laut Hilfetext gibt es auch die Konstante *ppWindowMixed*, tatsächlich tritt jedoch bei Verwendung ein Laufzeitfehler auf. Auch im Objektkatalog ist die Konstante nicht zu finden. Aus diesem Grund sollten Sie davon ausgehen, dass die Konstante nicht existiert und somit auch nicht verwendet werden kann.

Sie können natürlich auch über die Konstanten oder deren Wert abfragen, welchen Status das Anwendungsfenster hat und diesen bei Bedarf ändern. Das folgende Beispiel zeigt, wie PowerPoint maximiert werden kann, wenn es minimiert oder in Normalgröße dargestellt wird.

```
Sub Maximieren_wenn_nötig()
  If (Application.WindowState = _
    ppWindowMinimized) Or _
    (Application.WindowState = 1) Then
    Application.WindowState = _
    ppWindowMaximized
  End If
End Sub
```

Systemdaten ermitteln

In PowerPoint können Sie genauso wie in Excel und Access bestimmte Systemdaten ermitteln. Dazu gehören neben anwendungsspezifischen Daten auch Angaben über das Betriebssystem, das Installationsverzeichnis und Name und Version der Anwendung.

Das folgende kleine Makro zeigt, wie Sie diese Informationen ermitteln und in einem Dialog ausgeben.

```
Sub Infos()
  Dim strText  As String
  strText = "Betriebssystem: " & _
    Application.OperatingSystem _
    & Chr(10)
  strText = strText & "Verzeichnis: " _
    & Application.Path & Chr(10)
```

```
  strText = strText & "Anwendungsname: " _
    & Application.Name & Chr(10)
  strText = strText & "Version: " & _
    Application.Version
  MsgBox strText, vbInformation, _
    "Programm-Info!"
End Sub
```

Das Beispiel gibt folgende Meldung aus, wenn es in PowerPoint 2000 ausgeführt wird.

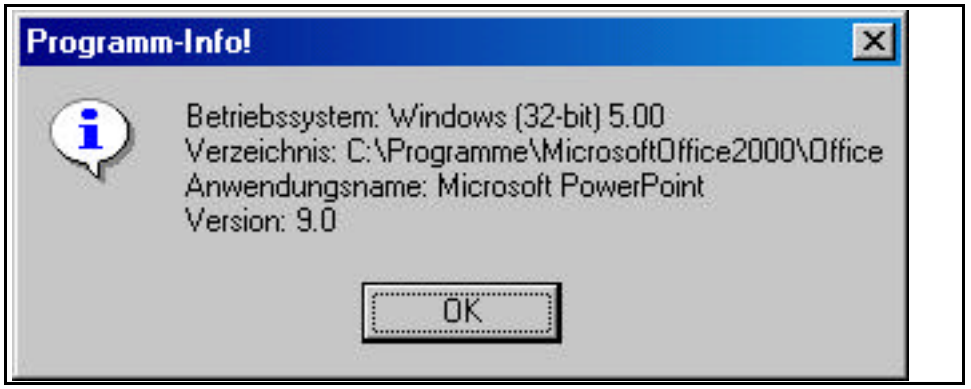

Abb. 1: Ausgabe der ermittelten Systemeigenschaften

Die Eigenschaft *Path* liefert das Verzeichnis, in dem sich die Datei POWERPNT.EXE befindet. Wenn Sie die Eigenschaft *Path* eines *Presentation*-Objektes abfragen, liefert sie das Verzeichnis, in dem die Präsentation gespeichert ist. In beiden Fällen erfolgt die Pfadangabe ohne den abschließenden Backslash "\".

Die *Name*-Eigenschaft liefert bei Anwendung auf das Objekt *Application* den Namen der Anwendung, bei Abfrage der *Name*-Eigenschaft eines *Presentation*-Objektes liefert Sie den Dateinamen der Präsentation ohne Pfadangabe.

Hinweis: Wenn Sie also die *Path*- und *Name*-Eigenschaft verwenden möchten, um den vollständigen Namen einer Präsentation einschließlich Verzeichnis zu ermitteln, dann müssen Sie zwischen *Name* und *Path*

noch einen "\" einfügen. Sie können aber in den meisten Fällen dazu die Eigenschaft *Fullname* verwenden. Sie ermittelt den Dateinamen einschließlich Verzeichnis.

```
Sub NameUndPfad()
  MsgBox Application.ActivePresentation.FullName
End Sub
```

Fenstertitel ändern

Mithilfe der Eigenschaft *Caption* des *Application*-Objektes können Sie den Fenstertitel ändern. Nachdem Sie das folgende VBA-Makro ausgeführt haben, sieht er wie folgt aus:

Abb. 2: Geänderter Fenstertitel

Um diese Änderung zu bewirken, ist genau eine Anweisung notwendig.

```
Sub Anwendungstitel()
    Application.Caption = _
    "Mein erstes PP Programm"
End Sub
```

Diese Änderung kann sehr wichtig sein, wenn Sie die Methode *AppActivate* verwenden. Sie aktiviert eine Anwendung mit einem bestimmten Text in der Titelzeile. Um Fehler auszuschließen, können Sie den Text mit *Caption* festlegen, um dann sicherzustellen, dass der Text auch mit dem an *AppActivate* übergebenen Text übereinstimmt.

Aktive Objekte ermitteln

Die Eigenschaften *ActivePresentation*, *ActivePrinter* und *ActiveWindow* geben die aktive Präsentation, den aktuell eingestellten Drucker und das aktive Fenster zurück. Während *ActivePrinter* eine Zeichenkette mit dem Druckernamen zurückgibt, geben die anderen beiden Eigenschaften ein Objekt vom Typ *Presentation* bzw. *DocumentWindow* zurück, dessen *Name*- bzw. *Caption*-Eigenschaft dann noch ermittelt werden muss.

Sie haben mit diesen Eigenschaften die Möglichkeit, vor dem Druck einer Präsentation oder einer Folie zu prüfen, ob der richtige Drucker aktiviert ist, oder ob vor dem Schließen der aktiven Präsentation auch die richtige Präsentation aktiviert ist. Das folgende Beispiel gibt diese drei Eigenschaften in einem Dialog aus.

```
Sub Aktives()
  Dim strText As String
  strText = "Aktive Präsentation: " _
    & ActivePresentation.Name & _
    Chr(10)
  strText = strText & "Aktiver Drucker : " _
    & Application.ActivePrinter _
    & Chr(10)
  strText = strText & "Aktives Fenster: " _
    & ActiveWindow.Caption
  MsgBox strText, vbInformation, _
    "Status-Info!"
End Sub
```

Sie sehen am Quelltext, dass auch PowerPoint Default-Objekte kennt (*Application*). Während für die Eigenschaften *ActiveWindow* und *ActivePresentation Application* das Default-Objekt ist, müssen Sie es für *ActivePrinter* explizit angeben.

Wichtige Methoden

Außer den bisher behandelten Eigenschaften des Objektes *Application* gibt es noch drei wichtige Methoden. Dies sind *Help*, *Quit* und *Run*.

Help ruft die PowerPoint-Hilfe auf, wenn nicht die beiden Parameter *HelpFile* und *ContextID* angegeben werden. Sie bestimmen die Hilfedatei und die Hilfeseite, die angezeigt werden soll. Entfällt nur der Parameter *ContextID*, so wird die Seite mit der Nr. 1 gezeigt, die in der Regel die Inhaltsübersicht enthält. Das folgende Beispiel zeigt die Hilfe zu PowerPoint an.

```
Sub Help_Methode()
  Application.Help
End Sub
Sub Run_Methode()
  Application.Run "Help_Methode"
End Sub
```

Die vorstehende Prozedur verwendet die *Run*-Methode, um eine VBA-Prozedur auszuführen. Sie können jedoch eine solche Prozedur auch ohne *Run* starten, indem Sie deren Namen angeben. In diesem Fall sähe das Makro wie folgt aus:

```
Sub Run_Methode()
  Help_Methode
End Sub
```

Die *Quit*-Methode beendet die Anwendung PowerPoint. Wenn Sie Änderungen an geöffneten Präsentationen nicht zuvor mit *Save* oder *SaveAs* gespeichert haben, fragt PowerPoint vor dem Beenden, ob die Änderungen gespeichert werden sollen. Der folgende Quelltext beendet PowerPoint.

```
Sub Quit_Methode()
  Application.Quit 'beendet PP
End Sub
```

Dies sind selbstverständlich nicht alle Eigenschaften und Methoden des *Application*-Objektes. Eine vollständige Darstellung würde sicherlich schon ein halbes Buch füllen. Das hier Dargestellte reicht jedoch vorerst aus, um erfolgreich Präsentationen manipulieren und erstellen sowie kleine Anwendungen entwickeln zu können.

Präsentationen

In diesem Abschnitt finden Sie Informationen darüber, wie PowerPoint-Präsentationen geöffnet, geschlossen und manipuliert werden und wie Bildschirmpräsentationen gestartet und beeinflusst werden können.

Öffnen und Schließen

Bevor Sie Präsentationen mit VBA bearbeiten können, müssen Sie diese erst einmal öffnen können. Dazu verwenden Sie die *Open*-Methode. Mithilfe des Parameters *Filename* bestimmen Sie die zu öffnende Datei. Das folgende Beispiel öffnet die Datei TEST.PPT.

```
Sub PR_öffnen()
  Application.Presentations.Open _
    FileName:="test.ppt"
End Sub
```

Für die *Open*-Methode können die in folgender Tabelle aufgelisteten Parameter verwendet werden:

Parameter	**Typ**	**Angabe notwendig**	**Beschreibung**
FileName	String	Ja	Gibt den Namen der Datei an, die geöffnet werden soll.
ReadOnly	Long	nein	Wenn die Datei schreibgeschützt geöffnet werden soll, ist hier *True* anzugeben. Wird der Parameter weggelassen oder *False* angegeben, so wird der Schreibschutz für die Datei nicht aktiviert.
Untitled	Long	nein	Soll eine Kopie der Datei geöffnet werden, geben Sie hier *True* an. In

			diesem Fall können Sie mit *SaveAs* die Datei unter einem beliebigen Namen speichern. Default-Wert ist *False.*
WithWindow	Long	nein	Soll die Datei in einem sichtbaren Fenster geöffnet werden, dann geben Sie hier *True* an. Dies ist auch der Default-Wert. Bei Angabe von *False* wird dir Datei in einem ausgeblendeten Fenster geöffnet.

Tab. 2: Parameter der *Open*-Methode

Fehler können hier auftreten, wenn sich die Datei nicht im aktuellen Verzeichnis befindet. Diese Fehlerquelle können Sie beseitigen, indem Sie als *Filename* nicht nur den Dateinamen, sondern auch das Verzeichnis angeben oder vor der Verwendung der *Open*-Methode in das Verzeichnis wechseln. Mit dem folgenden Befehl wechseln Sie zum Beispiel in das Verzeichnis *Temp* und öffnen anschließend die Präsentation *Test.ppt.*

```
Sub PR_öffnen_Ord()
  ChDir "C:\Temp"
  Application.Presentations.Open _
    FileName:="test.ppt"
End Sub
```

Zur Vermeidung dieses Fehlers können Sie in PowerPoint auch das Verzeichnis bestimmen, in dem standardmäßig Dateien gespeichert und auch gesucht werden. Dazu aktivieren Sie die Anwendung PowerPoint und wählen im Menü *Extras / Optionen.* Im Blatt *Speichern* des Dialogs geben Sie im Feld *Standardarbeitsordner* das Verzeichnis ein, in dem sich die Datei befindet bzw. in dem alle Dateien automatisch gespeichert werden. Klicken Sie dann auf *OK* um die Änderungen zu übernehmen und den Dialog zu schließen.

Um eine geöffnete Präsentation zu schließen, verwenden Sie die *Close*-Methode. Sie schließt die durch das Objekt angegebene Präsentation. Eine Angabe von Parametern ist nicht möglich. Das folgende Beispiel schließt die Präsentation mit dem Namen *Test.ppt.*

```
Sub PR_schließen()
  Application.Presentations("test.ppt").Close
End Sub
```

Erstellen und Löschen von Präsentationen

Um Präsentationen zu erstellen, stellt VBA die Methode *Add* des *Presentations*-Objektes zur Verfügung, mit der ein neues *Presentation*-Objekt erzeugt wird und an die *Presentations*-Auflistung angehängt wird. Der einzige Parameter der Methode *WithWindow* gibt an, ob die Präsentation sichtbar oder unsichtbar erzeugt werden soll.

Die dazu notwendige VBA-Prozedur lautet:

```
Sub Neue_PR()
  Application.Presentations.Add _
    withwindow:=True
End Sub
```

Hinweis: Sie sollten jedoch besser eine Objektvariable vom Typ *Presentation* zu Hilfe nehmen, wenn Sie eine neue Präsentation erzeugen, da Sie dann eine effiziente Möglichkeit haben, auf die neue Präsentation zuzugreifen, auch wenn Sie deren Namen nicht kennen. Der Name einer neuen Präsentation richtet sich nämlich immer nach der in einer PowerPoint-Sitzung bereits erzeugten Anzahl von Präsentationen. Die erste Präsentation bekommt den Namen *Präsentation1*, die zweite *Präsentation2* etc. Sie müssen also mit der ersten Methode genau wissen, wie die neue Präsentation heißt, um auf sie zugreifen zu können. Das nächste Beispiel zeigt, wie Sie es besser machen können.

```
Sub Neue_PR2()
  Dim PR As Presentation
  Set PR = Application.Presentations.Add
End Sub
```

Auch in PowerPoint gibt es keine Methode zum Löschen von Dateien. Sie müssen deshalb die VBA-Anweisung *Kill* verwenden.

```
Sub PR_Löschen()
  Kill "Test2.ppt"
End Sub
```

Hinweis: Hier kommt es zu Fehlern , wenn die Präsentation noch von PowerPoint geöffnet ist, wenn Sie versuchen, die Präsentation mit *Kill* zu löschen oder wenn sie nicht im aktuellen Verzeichnis gefunden werden kann. Die letztgenannte Fehlerquelle können Sie dadurch ausschließen, dass Sie den Dateinamen mit Verzeichnis und Laufwerk angeben. Um die erste Fehlerquelle zu vermeiden, sollten Sie die Datei vor dem Verwenden der *Kill*-Anweisung schließen.

Wenn Sie eine Präsentation erstellt haben, müssen Sie diese natürlich auch speichern können. Dazu bietet VBA die *Save*-Methode oder *SaveAs*-Methode an. Mit Hilfe der *SaveAs*-Methode können Sie eine Präsentation unter einem als Parameter *FileName* anzugebenden Namen speichern.

Sie können die *SaveAs*-Methode im Gegensatz zur *Save*-Methode dazu nutzen, eine neue Präsentation zum ersten Mal zu speichern, oder eine vorhandene Präsentation unter einem anderen Namen zu speichern. Wenn Sie lediglich die Änderungen an einer geöffneten Präsentation speichern möchten, verwenden Sie dazu die *Save*-Methode. Hier können Sie keine Parameter angeben.

Das folgende Beispiel zeigt, wie Sie die aktive Präsentation speichern können, wenn Sie sie bereits einmal gespeichert haben. Dies können Sie feststellen, indem Sie die *Fullname*- oder *Path*-Eigenschaft der Präsentation abfragen. Sie geben den Wert "" (=leere Zeichenfolge) zurück, wenn die Präsentation noch nicht gespeichert wurde.

```
Sub AktivePräsentation_speichern()
  ActivePresentation.Save
End Sub
```

Hinweis: Mit Hilfe der Methode *SaveCopyAs* können Sie eine Kopie der angegebenen Präsentation speichern. Den Namen der Kopie müssen Sie als *FileName* angeben.

Manipulation von Präsentationen

Bei der Manipulation von Präsentationen sind die Gestaltung einzelner Folien und die Manipulation der gesamten Präsentation durch bestimmte Einstellungen zu unterscheiden. Zunächst wird es darum gehen, wie Folien erstellt, gelöscht, verschoben und formatiert werden können.

Zuweisen von Vorlagen für die Präsentation

Auch in PowerPoint werden Präsentationen mithilfe von Präsentationsvorlagen formatiert. Diese tragen die Dateinamenserweiterung .POT und bestimmen die Hintergrundformatierung, die auf jeder Folie enthaltenen Felder wie Datum und Folien-Nr. etc.

Sie können einer neuen oder vorhandenen Präsentation eine Vorlage zuweisen, indem Sie die Methode *ApplyTemplate* verwenden. Der Parameter *FileName* bestimmt dabei die Formatvorlage, die verwendet werden soll. Befindet Sie sich nicht im aktuellen Verzeichnis, müssen Sie das Verzeichnis und das Laufwerk mit angeben.

```
Sub Vorlage_zuweisen()
  Dim PR As Presentation
  Set PR = ActivePresentation
  PR.ApplyTemplate FileName:="Eigene.pot"
End Sub
```

Hinweis: Wenn Sie einer Präsentation, die noch keine Folien enthält, eine Vorlage zuordnen, sehen Sie das Ergebnis, nämlich die Formatierung erst, wenn Sie die erste Folie in die Präsentation eingefügt haben.

Manipulation von Folien

Um eine neue Folie erstellen zu können, muss ein Objekt vom Typ *Slide* an die *Slides*-Auflistung des *Presentation*-Objektes angehängt werden. Dafür ist die *Add*-Methode des *Slides*-Objektes zuständig.

Die *Add*-Methode kennt die zwei Parameter *Index* und *Layout*. *Index* bestimmt die Position einer Folie innerhalb der Präsentation und somit auch die Folien-Nummer und darf nicht höher sein als die Anzahl Folien plus 1. Sie können die Anzahl der vorhandenen Folien über die Eigenschaft *Count* des *Slides*-Objektes ermitteln.

Layout bestimmt die Gestaltung der Folie. Diese entspricht den beim Einfügen einer Folie über das Menü auszuwählenden Folienlayouts. Das folgende Beispiel zeigt, wie eine Präsentation erstellt wird und dort nach Zuweisung der Vorlage eine neue Folie mit *Titel*-Layout eingefügt wird.

```
'Variablendeklarationen a. Modulebene
Dim PR As Presentation
Dim FOL As Slide, Icon As Shape
Dim SCHR As Shape
Dim BT As Shape


Sub Präsentation_erstellen()
  'Aktives Verzeichnis festlegen
  ChDir Application.ActivePresentation.Path
  'Erstellen der Präsentation
  Set PR = Application.Presentations.Add
  PR.SaveAs FileName:="Präsent1.ppt", _
    EmbedTruetypefonts:=True
  'Vorlage zuweisen
  PR.ApplyTemplate FileName:="Eigene.pot"
  'Erste Folie einfügen
  Set FOL = PR.Slides.Add(Index:=1, _
    Layout:=ppLayoutTitle)
  'Gestaltung der Folie
  '...
```

```
End Sub
```

Das Aussehen der Folie nach Ausführung dieser Anweisungen richtet sich im Wesentlichen nach der verwendeten Vorlage. Sie können mit den Anweisungen

```
Set PR = ActivePresentation
  PR.Slides.Add _
    Index:=PR.Slides.Count + 1, _
    Layout:=ppLayoutBlank
```

eine weitere Folie am Ende der aktiven Präsentation einfügen. Bei der ersten Folie brauchen Sie die Anzahl der Folie nicht zu ermitteln, da sie in diesem Fall bekannt ist. Möchten Sie aber zusätzliche Folien einfügen und wissen nicht, wie viele Folien schon vorhanden sind, können Sie die Anzahl mit *Count* ermitteln.

Einfügen von Objekten in die Folien

Da leere Folien in einer Präsentation natürlich ziemlich sinnlos sind, soll nun erläutert werden, wie Sie Objekte in Folien einfügen, formatieren und manipulieren können. Auf die Objekte der Folien wird über die *Shapes*-Auflistung zugegriffen. Sie beinhaltet auch die Platzhalter der Folienvorlage z.B. für den Folientitel. Auf diese Platzhalter kann jedoch auch über die *Placeholders*-Auflistung zugegriffen werden. In beiden Fällen erfolgt der Zugriff über den Index des Objektes.

Das folgende Beispiel zeigt, wie Sie den Folientitel der aktuellen Folie ändern können. Dazu wird über die Eigenschaft *Title* auf den Platzhalter *Titel* zugegriffen. Für den Untertitel besteht eine solche Eigenschaft leider nicht. Hier musste daher die Auflistung *Placeholders* verwendet werden, um den Untertitel festzulegen.

```
Sub Präsentation_erstellen()
...
  'Gestaltung der Folie
  'Folientitel festlegen
  FOL.Shapes.Title.TextFrame.TextRange.Text _
  = "Microsoft Office 2000"
  FOL.Shapes.Placeholders(2).TextFrame _
```

```
    .TextRange.Text = "Ein erster Überblick"
...
```

Das Ergebnis der Prozedur sieht folgendermaßen aus:

Abb. 3: Folie nach Festlegung von Titel und Untertitel

Um ein neues Objekt einzufügen, verwenden Sie eine *Add*...-Methode des *Shapes*-Objektes. Das folgende Beispiel zeigt, wie Sie mithilfe der *AddPicture*-Methode eine Grafik in die Folie einfügen, die an der Stelle in der Folie erscheint, an der bisher das Rechteck mit Rahmen zu sehen war. Der Parameter *FileName* legt die Datei fest, die eingefügt werden soll, *LinkToFile* bestimmt, ob die Grafik verknüpft oder eingebunden werden soll und die letzten vier Parameter *Top*, *Left*, *Width* und *Height* legen Position und Größe des Bildes fest.

```
...
  'Einfügen des Bildes in den Rahmen
  Set Icon = FOL.Shapes.AddPicture( _
    FileName:="Office.bmp", _
    LinkToFile:=msoFalse, _
    SaveWithDocument:=msoTrue, _
    Left:=648, Top:=24, _
    Width:=56, Height:=56)
    Icon.Name = "Bild"
...
```

Hinweis: Problematisch wird die genaue Bestimmung von Position und Größe eines Objektes in der Folie. Diese Position können Sie durch Probieren herausfinden. Eine weitere Möglichkeit besteht darin, das Objekt an der gewünschten Stelle einzufügen und dann dessen Eigenschaften im Direktfenster ausgeben zu lassen, um sie in der *Add*-Methode zu verwenden.

Für den späteren Zugriff auf bestimmte Objekte können Sie diesen Namen geben. Dazu setzen Sie einfach die *Name*-Eigenschaft des Objektes.

Die Anweisung *Icon.Name="Bild"* gibt z.B. dem im letzten Beispiel eingefügten Bitmap, auf das die Objektvariable *Icon* verweist, den Namen *Bild*. Die Eigenschaft *Shadow* gibt ein Objekt vom Typ *ShadowFormat* zurück, über dessen Eigenschaften der Schatten eines Objektes festgelegt werden kann. Die Anweisung *Icon.Shadow.Visible = msoFalse* sorgt dafür, dass dem Objekt kein Schatten hinzugefügt wird.

Sie können mithilfe der *AddTitle*-Methode den Platzhalter für den Folientitel wieder einfügen, wenn er gelöscht wurde. Dazu können Sie die *HasTitle*-Eigenschaft abfragen, um festzustellen, ob ein Titel vorhanden ist oder nicht. Ist kein Titel vorhanden, so gibt die *HasTitle*-Eigenschaft *False* zurück.

Das folgende Beispiel zeigt, wie Sie mit *AddTextEffect* ein WordArt-Objekt auf der zweiten Folie der neuen Präsentation einfügen können. Die Konstanten des Parameters *PresetTextEffect* entsprechen den im Auswahldialog dargestellten Effekten. Sie werden beginnend mit 1 durchnummeriert und heißen somit *msoPresetTextEffect1* bis *msoPresetTextEffect30*. Der Quelltext, der ein solches

WordArt-Objekt erzeugt und in eine Folie der Präsentation einfügt, sieht folgendermaßen aus:

```
...
  'Erstellen der zweiten Folie
  Set FOL = PR.Slides.Add(2, ppLayoutBlank)
  'Formatierung der zweiten Folie
  'Kopieren des Bildes von der ersten F.
  Icon.Copy
  FOL.Shapes.Paste
  Set SCHR = FOL.Shapes.AddTextEffect( _
    PresetTextEffect:=msoTextEffect27, _
    Text:="Dies ist WordArt " & Chr(10) & _
    "im Office 2000" & _
    Chr(10) & "Verfügbar in Powerpoint, " _
    & Chr(10) & "Excel und Word !", _
    FontName:="Arial Rounded MT Bold", _
    FontSize:=60, Left:=60 _
    , Top:=100, FontBold:=True, FontItalic:= _
    False)
  SCHR.Name = "Schriftzug"
...
```

Zunächst wird dazu mit der *Add*-Methode der *Slides*-Auflistung eine neue Folie erstellt, in die dann mit der *Copy*-Methode des *Shape*-Objektes das Bild aus der ersten Folie kopiert und mit der *Paste*-Methode der *Shapes*-Auflistung eingefügt wird. Danach wird mit der *AddTextEffect*-Methode das *WordArt*-Objekt erstellt und ein Verweis darauf in der Objektvariablen *SCHR* gespeichert, die auf Modulebene deklariert wurde.

Sie müssen alle Parameter der Methode angeben. Ihre Bedeutung wird aus ihrem Namen deutlich und bedarf daher keiner weiteren Erläuterung. Anschließend wird dem neuen Objekt mit der *Name*-Eigenschaft ein Name zugeordnet, über den Sie dann auch ohne eine Objektvariable oder einen bekannten Index auf das Objekt zugreifen können.

Im nächsten Beispiel wird eine *AutoForm* in die dritte Folie der Präsentation eingefügt. Dazu wird die Folie wieder mit der *Add*-Methode erstellt und das

Bild mit dem Office-Symbol wird dort eingefügt, indem es von der ersten Folie kopiert wird. Da aber jetzt die Objektvariable keinen Verweis mehr auf das Bild auf der ersten Folie enthält, wird mithilfe des Namens über die *Shapes*-Auflistung auf das Bild zugegriffen.

```
...
  'Erstellen der dritten Folie
  Set FOL = PR.Slides.Add(3, ppLayoutBlank)
  'Formatierung der dritten Folie
  'Kopieren des Bildes von der ersten F.
  PR.Slides(1).Shapes("Bild").Copy
  FOL.Shapes.Paste
  Set BT = FOL.Shapes.AddShape( _
    msoShapeActionButtonSound, _
    600, 350, 70, 50)
...
```

Danach wird mit der Methode *AddShape* eine AutoForm eingefügt. Mit dieser Methode können Sie bei Auswahl der richtigen Konstante alle Formen einfügen, die Sie über das Menü *AutoForm* der *Zeichnen*-Symbolleiste erstellen können.

Dort finden Sie in PowerPoint auch eine Kategorie *Interaktive Schaltflächen*. Damit können Sie Schaltflächen in Ihre Folien einfügen, die nicht nur die Steuerung der Präsentation durch den Anwender ermöglichen, sondern auch das Ausführen von Makros und Anwendungen. Eine solche Schaltfläche zum Abspielen von Sounddateien fügt das Beispiel in die dritte Folie ein.

Über die Eigenschaft *ActionSettings*, die ein *ActionSetting*-Objekt zurückgibt, können Sie für die Schaltfläche festlegen, welche Aktion beim Klicken auf die Schaltfläche ausgelöst werden soll. Wenn Sie z.B. Videos in der Präsentation starten möchten, müssen Sie dazu ein *Media*-Objekt einfügen. Dazu verwenden Sie die *AddMediaObjekt*-Methode des *Shapes*-Objektes.

Das folgende Beispiel zeigt, wie Sie in die dritte Folie der Präsentation ein solches Video einfügen können.

```
...
```

```
  'Einfügen eines Videos
  FOL.Shapes.AddMediaObject "praesgr.avi", _
    40, 100
End Sub
```

Verwenden von Präsentationen

In der Regel werden Präsentationen erstellt, um Sie entweder am Bildschirm als Bildschirmpräsentationen ausführen zu lassen oder um sie auf Folien zu drucken, die dann am Overheadprojektor gezeigt werden.

Präsentationen ausführen

Präsentationen werden mit Hilfe der *Run*-Methode ausgeführt. Die einfachste Möglichkeit, eine fertige Präsentation auszuführen, ist die folgende:

```
Sub Starten()
  Set PR = Presentations.Open(FileName:= _
    "Präsent1.ppt", WithWindow:=msoFalse)
  PR.SlideShowSettings.Run
End Sub
```

Dazu wird die Präsentation zunächst geöffnet. Der Parameter *WithWindow* der *Open*-Methode sorgt aber dafür, dass die Präsentation unsichtbar bleibt.

Mit Hilfe der *Run*-Methode des *SlideShowSettings*-Objektes wird dann die Bildschirmpräsentation mit den voreingestellten Effekten und Zeiten gestartet. Der folgende Quelltext zeigt einen Ausschnitt aus den möglichen Einstellungen für die Bildschirmpräsentation.

Zunächst wird die Präsentation wieder unsichtbar geöffnet. Danach werden bestimmte Eigenschaften des *SlideShowSettings*-Objektes gesetzt. *AdvanceMode* legt fest, dass für die Präsentation die festgelegten Zeiten verwendet werden sollen. Der Wert *msoTrue* der Eigenschaft *LoopUntilStopped*

bestimmt, dass die Präsentation so lange immer wieder von vorne beginnt, bis Esc gedrückt wird. *Pointercolor* bestimmt die Farbe des Stiftes, der während der Präsentation anstatt des Pfeils eingeblendet werden kann. Mit *With PR.SlideShowSettings.Run.View* wird das Fenster bearbeitet, in dem die Präsentation abläuft. Dazu wird mit *Pointertype* der Pfeil durch den Stift ersetzt und anschließend mit *GotoSlide(4)* zur vierten Folie gesprungen, um mit dieser die Präsentation zu beginnen. *PR.Close* schließt zum Schluss wieder die Präsentation.

```
Sub Starten2()
  Set PR = Presentations.Open(FileName:= _
    "Präsent1.ppt", withWindow:=msoFalse)
  With PR.SlideShowSettings
    .AdvanceMode = ppSlideShowUseSlideTimings
    .LoopUntilStopped = msoTrue
    .PointerColor.RGB = RGB(255, 0, 0)
  End With
  With PR.SlideShowSettings.Run.View
    .PointerType = ppSlideShowPointerPen
    .GotoSlide (4)
  End With
  PR.Close
End Sub
```

Präsentationen drucken

Das folgende Beispiel zeigt, wie Sie eine Präsentation oder Ausschnitte davon drucken können, welche wichtigen Einstellungen es gibt und wie Sie sie vornehmen.

Zunächst wird der Objektvariablen *PR* ein Verweis auf die aktive Präsentation zugewiesen, bevor in einer *With*-Anweisung die Einstellungen für den Druck vorgenommen werden.

FitToPage = *msoTrue* bewirkt, dass die Foliengröße an die Seitengröße angepasst wird. Dies führt dazu, dass der gesamte verfügbare Platz auf der auszudruckenden Seite ausgefüllt wird. Die Eigenschaft *FrameSlides* des

PrintOptions-Objektes bestimmt, ob ein Rahmen um die Folie gedruckt werden soll oder nicht. Im Beispiel wird ein Rahmen gedruckt.

Mit der Eigenschaft *PrintColorType* können Sie festlegen, ob farbig oder schwarz-weiß gedruckt werden soll. Hierzu stehen folgende Konstanten zur Verfügung:

Konstante	**Bedeutung**
ppPrintBlackAndWhite	Druck erfolgt in Schwarz-weiß mit Graustufen. Diese Möglichkeit ist insbesondere für Laserdrucker geeignet, die Farbverläufe dann als Graustufen wiedergeben.
ppPrintColor	Farbdruck mit der durch den Drucker und Druckertreiber bestimmten maximalen Anzahl Farben.
ppPrintPureBlackAndWhite	Ausdruck erfolgt nur in den Farben Schwarz und Weiß. Es gibt keine Grauabstufungen.

Tab. 3: Konstanten für die PrintColorType -Eigenschaft

Hinweis: Sollten Sie z.B. beim Drucken eines WordArt-Objektes mit Füllmuster einen Ausdruck vorfinden, bei dem das Füllmuster nicht die Buchstaben, sondern stattdessen den Schrifthintergrund füllt, so hat dies nichts damit zu tun, welche Druckoptionen Sie gewählt haben. Versuchen Sie den Druck mit einem anderen Druckertreiber, der mit Ihrem Drucker kompatibel ist. Es gibt offensichtlich bei einigen Treibern Probleme mit Füllmustern von *WordArt*-Objekten. Wenn Sie z.B. einen Epson EPL 5500 haben, können Sie statt des Originaltreibers von Epson den mit Windows 95 mitgelieferten Treiber für den Epson EPL 5200 verwenden. Mit diesem erfolgt der Druck einwandfrei, aber leider nur mit 300 dpi.

PrintFontsAsGraphics = msoCTrue legt fest, dass Schriften als Grafiken gedruckt werden. Dies verhindert, dass verschiedene Drucker TrueType-Schriften durch ihre Druckerschriften ersetzen. Dies machen natürlich nicht alle

Drucker und es geschieht in der Regel auch nur, wenn Sie es im Druckertreiber oder am Drucker einstellen. Sie können mit dieser Einstellung jedoch auch Probleme beim Druck des €-Zeichens vermeiden. Mithilfe der Eigenschaft *OutputType* können Sie bestimmen, in welcher Form Sie die Präsentation drucken möchten. Dazu stehen verschiedene Konstanten zur Verfügung.

Konstanten	**Beschreibung**
ppPrintOutputBuildSlides	Druckt Folien
ppPrintOutputNotesPages	Druckt Notizen
ppPrintOutputOutline	Druckt Gliederung
ppPrintOutputSixSlideHandouts	Druckt Handzettel mit 6 Folien je Seite
ppPrintOutputSlides	Druckt Folien und ist Default-Wert
ppPrintOutputThreeSlideHandouts	Druckt Handzettel mit 3 Folien je Seite
ppPrintOutputTwoSlideHandouts	Druckt Handzettel mit 2 Folien je Seite.

Tab. 4: Konstanten für die OutputType-Eigenschaft

Mithilfe der Eigenschaft *PrintHiddenSlides* können Sie bestimmen, ob auch ausgeblendete Folien gedruckt werden sollen. Im Beispiel ist dies der Fall, allerdings enthält die Präsentation keine ausgeblendeten Folien. *PrintInBackground* legt fest, ob der Druck im Hintergrund erfolgen soll. Default-Wert ist hier *True*. Der Hintergrunddruck hat den Vorteil, dass Sie während des Ausdrucks, der je nach Drucker und Größe der Präsentation sehr lange dauern kann, weiter mit dem Rechner arbeiten können.

Nun erfolgt der eigentliche Ausdruck mithilfe der *Printout*-Methode. Die Bedeutung Ihrer Parameter finden Sie in folgender Tabelle. Alle Parameter sind optional.

Parameter	**Typ**	**Beschreibung**
From	Long	Gibt die Seitenzahl der ersten Seite an, die gedruckt werden soll. Wird dieses Argument nicht angegeben, beginnt der

		Druck am Anfang der Präsentation. Mit der Angabe der *To*- und *From*-Argumente werden die Inhalte des *PrintRanges*-Objekts und der Wert der *RangeType*-Eigenschaft für die Präsentation festgelegt.
To	Long	Gibt die Seitenzahl der letzten Seite an, die gedruckt werden soll. Wird dieses Argument nicht angegeben, wird die Präsentation bis zum Ende gedruckt. Mit der Angabe der *To*- und *From*-Argumente werden die Inhalte des *PrintRanges*-Objekts und der Wert der *RangeType*-Eigenschaft für die Präsentation festgelegt.
PrintToFile	String	Gibt den Namen der Datei, an die die Druckausgabe weitergeleitet werden soll. Wenn Sie dieses Argument angeben, wird die Datei nicht an den Drucker weitergeleitet, sondern in eine Datei umgeleitet. Wird dieses Argument nicht angegeben, wird die Datei an einen Drucker weitergeleitet.
Copies	Long	Bestimmt die Anzahl der zu druckenden Kopien. Wird dieses Argument nicht angegeben, wird nur eine Kopie gedruckt. Mit der Angabe dieses Arguments wird der Wert der *NumberOfCopies*-Eigenschaft festgelegt.
Collate	Long	Dieses Argument ist *True*, wenn eine vollständige Kopie der Präsentation gedruckt werden soll, bevor die erste Seite der nächsten Kopie gedruckt wird. Wird dieses Argument nicht angegeben, werden mehrere Kopien sortiert. Mit der Angabe dieses Arguments wird der Wert der *Collate*-Eigenschaft festgelegt.

Tab. 5: Parameter der Printout-Methode (Quelle: Online-Hilfe)

Obwohl der Parameter *Collate* vom Typ *Long* ist, kann ihm ein boolescher Wert, *True* oder *False* zugewiesen werden. Besser ist jedoch, Sie verwenden die vordefinierten Microsoft-Office-Konstanten *msoTrue* und *msoFalse*, die vom Typ *Long* sind.

```
Sub Präsentation_drucken()
```

```
  Set PR = ActivePresentation
  With PR.PrintOptions
    .FitToPage = msoCTrue
    .FrameSlides = msoCTrue
    .PrintColorType = ppPrintBlackAndWhite
    .PrintFontsAsGraphics = msoCTrue
    .OutputType = ppPrintOutputSlides
    .PrintHiddenSlides = msoCTrue
    .PrintInBackground = msoTrue
  End With

  PR.PrintOut From:=1, To:=1, Copies:=1, _
  Collate:=msoFalse
End Sub
```

Zusammenfassung, Fragen und Übungen

?? Eine PowerPoint-Präsentation wird durch das *Presentation*-Objekt dargestellt.

?? Ihm untergeordnet ist die *Slides*-Auflistung, die alle Folien der Präsentation darstellt.

?? Die einzelne Folie wird durch das *Slide*-Objekt repräsentiert.

?? Präsentationen werden mit der *Open*-Methode geöffnet und mit der *Close*-Methode geschlossen.

?? Neue Präsentationen werden mit *SaveAs*, andere mit *Save* gespeichert.

?? Der Folientitel wird durch die *Title*-Eigenschaft zurückgegeben.

Fragen und Übungen

1. Wozu dient das Schlüsselwort *Run* ?

2. Wie kann eine Präsentation geöffnet werden, ohne dass sie auf dem Bildschirm angezeigt wird? Erstellen Sie den entsprechenden Quellcode dazu.

3. Erstellen Sie ein VBA-Makro, das eine Präsentation öffnet und eine Kopie davon erstellt.
4. Erstellen Sie Quellcode, der in ein Listenfeld eines Formulars die Titel aller Folien der aktuellen Präsentation schreibt. Hat die Folie keinen Titel, soll stattdessen der Index eingefügt werden.
5. Schreiben Sie ein VBA-Makro, das die aktive Präsentation startet.