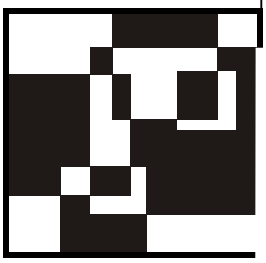
ГЛАВА 14



Программирование в PowerPoint

PowerPoint — это программа для работы с презентациями (т. е. с наборами графических изображений — слайдов, иногда со звуковым сопровождением). Использовать возможности VBA в PowerPoint на предприятиях приходится намного реже, чем возможности Word или Excel, однако иногда такая необходимость возникает. Часто специалисты используют презентации PowerPoint для сопровождения выступлений при демонстрации продуктов или услуг, отчетов о деятельности и т. п. Поскольку со слайдами можно связывать звуковое сопровождение, PowerPoint активно используется для целей обучения, например, для подготовки интерактивных уроков. Еще одна часто используемая возможность — создание звуковых книг с картинками для детей. При помощи PowerPoint можно создавать фотоальбомы со звуковым сопровождением, диафильмы со звуком, детские игры и многое другое. И как только данных становится много (например, цифровых фотографий может быть несколько сотен) сразу встает вопрос об автоматизации.

Чаще всего приходится программным способом выполнять следующие действия в PowerPoint:

- *автоматически создавать презентации* (например, на основе набора изображений в каталоге);
- *производить обработку презентаций* — менять формат изображений, добавлять или изменять аудиосопровождение и т. п. Чаще всего подобные действия приходится производить в тех ситуациях, когда презентации были связаны с внешними файлами и эти файлы изменились.

В PowerPoint система объектов выглядит следующим образом:

- объект самого высокого уровня — `Application` с набором свойств и методов, очень похожим на аналогичные объекты в Word и Excel;

- уровень ниже — коллекция `Presentations` с объектами `Presentation`. Можно сказать, что эти объекты по месту в иерархии примерно аналогичны объекту `Workbook` в Excel;
- в объект `Presentation` встроена коллекция `Slides` с объектами `Slide`. В качестве аналога можно привести листы `Worksheet` в книгах Excel;
- в объект `Slide` встроена коллекция `Shapes` с объектами `Shape`. Объекты `Shape` представляют собой все элементы слайда (всего их 22 типа: изображение, надпись, диаграмма, заголовок, таблица, автофигура и т. п.).

Вокруг этих четырех объектов — `Application`, `Presentation`, `Slide` и `Shape` — и строится вся объектная модель PowerPoint.

В этой главе мы не будем приводить справку по свойствам и методам различных объектов PowerPoint (их можно быстро найти при помощи макрорекордера), а вместо этого проиллюстрируем работу с PowerPoint на примерах из практики.

Предположим, что нам нужно создать презентацию PowerPoint на основе набора JPG-картинок, которые будут лежать в каталоге C:\Slides (например, они получены со сканера или цифрового фотоаппарата). Имена JPG-файлов следуют по порядку, например, с DSCN2440.JPG по DSCN2480.JPG. Файлов в каталоге может быть произвольное количество, поэтому нам нужно взять все файлы из этого каталога. Наша задача — поместить их в презентацию по порядку. Задача усугубляется тем, что JPG-файлы разного размера (по высоте и ширине), а слайды желательно сделать одинаковыми.

Как ни удивительно, код VBA для PowerPoint удобнее запускать не из PowerPoint, а из внешнего приложения, поддерживающего VBA, например, из Word или Excel. Таким образом, на момент запуска у нас гарантированно не будет активных презентаций и мы ничего не перепутаем при вставке.

Наше решение может выглядеть следующим образом:

1. Создаем новый документ в Word или Excel, в него помещаем кнопку или обеспечиваем другой графический интерфейс по вкусу. Главное — это не забыть добавить в проект ссылки на две объектные библиотеки:
   - *Microsoft PowerPoint 11.0 Object Library* (msppt.olb) — для объектов самого PowerPoint;
   - *Microsoft Scripting Runtime* (ScrRun.dll) — для того, чтобы можно было пользоваться объектом `FileSystemObject` и другими возможностями для работы с файловой системой. Эта библиотека, которая есть на любом компьютере начиная с Windows 2000, — самый удобный способ выполнения большинства действий в файловой системе.

   Далее можно приступать к созданию кода.

2. Конечно, первое, что нам потребуется — запустить PowerPoint. Делается это точно так же, как и для Word, Excel, Access и т. п.:

```
Dim oApp As New PowerPoint.Application
oApp.Activate
oApp.Visible = msoTrue
```

3. Следующее действие — создание новой пустой презентации:

```
Dim oPresent As PowerPoint.Presentation
Set oPresent = oApp.Presentations.Add()
```

   Все абсолютно стандартно, как будто мы создаем новый документ Word. А вот дальше начинаются моменты, специфические для PowerPoint.

4. Следующим действием должно быть создание слайда. Но нам придется создавать столько слайдов, сколько файлов находится в каталоге C:\Slides. Конечно, слайды будут создаваться в цикле. Вначале мы получаем при помощи библиотеки Scripting Runtime (можно было бы обойтись и средствами Office, но так проще) коллекцию всех файлов этого каталога:

```
Dim oFSO As New Scripting.FileSystemObject
Dim oFolder As Scripting.Folder
Dim oFile As Scripting.File

Set oFolder = oFSO.GetFolder("C:\Slides")
For Each oFile In oFolder.Files
    ...
Next
```

   Если мы вместо многоточия вставим строку, например, такого вида:

```
MsgBox oFile.Name
```

   то можно будет убедиться, что мы получили набор файлов в правильном порядке.

5. Далее нам все-таки нужно создать слайды. Делается это при помощи метода `Add()` коллекции `Slides`. В документации к русскому PowerPoint 2003 описание этого метода по непонятной причине отсутствует (несмотря на то, что справка по VBA все равно приводится на английском), но из всплывающей подсказки можно догадаться, что этот метод принимает два обязательных параметра: номер слайда в презентации (нумерация должна начинаться с 1), и одно из значений перечисления `ppSlideLayout` (из нескольких десятков), которое определяет шаблон слайда.

   Номер слайда придется обеспечивать счетчиком, а лучший для нас шаблон — пустой:

```
Dim nCounter As Integer
nCounter = 1
For Each oFile In oFolder.Files
    Set oSlide = oApp.ActivePresentation.Slides.Add(nCounter, _
        ppLayoutBlank)
    ...
    nCounter = nCounter + 1
Next
```

6. А теперь самое главное — вставляем в слайд изображение и настраиваем его размеры. Для этой цели можно использовать метод `AddPicture()` коллекции `Shapes` каждого слайда:

```
oSlide.Shapes.AddPicture FileName:="C:\Slides\" & oFile.Name, _
    LinkToFile:=msoFalse, SaveWithDocument:=msoTrue, _
    Left:=10, Top:=10, Width:=700, Height:=520
```

   Параметр `FileName` — это имя передаваемого файла. Именно он и будет меняться в цикле. Параметр `LinkToFile` определяет, будет ли файл изображения помещен внутрь презентации (`msoFalse`) или в презентацию будет помещена ссылка на него (`msoTrue`). Конечно, если вставляемые файлы не очень большие, то с точки зрения удобства и производительности презентации лучше помещать изображения внутрь презентации (файла PPT). Параметр `SaveWithDocument` определяет, сохранять ли изображения вместе с презентацией (в нашем случае сохранять). А параметры `Left`, `Top`, `Width` и `Height` нужны, чтобы сделать изображения одинакового размера (нужные значения подбираются вручную).

   Естественно, код этого пункта помещается вместо многоточия в цикл пункта 5. Чтобы не возиться с удалением обработанных файлов, я бы поместил в цикл еще одну очевидную строку:

```
oFile.Delete
```

Итоговый код для нашей задачи может выглядеть так:

```
Dim oApp As New PowerPoint.Application
oApp.Activate
oApp.Visible = msoTrue

Dim oPresent As PowerPoint.Presentation
Set oPresent = oApp.Presentations.Add()

Dim oFSO As New Scripting.FileSystemObject
Dim oFolder As Scripting.Folder
Dim oFile As Scripting.File
```

```
Set oFolder = oFSO.GetFolder("C:\Slides")
For Each oFile In oFolder.Files
    Set oSlide = oApp.ActivePresentation.Slides.Add(nCounter, _
        ppLayoutBlank)
    oSlide.Shapes.AddPicture FileName:="C:\Slides\" & oFile.Name, _
        LinkToFile:=msoFalse, SaveWithDocument:=msoTrue, _
        Left:=10, Top:=10, Width:=700, Height:=520
    oFile.Delete
Next
```

Несколько строк кода могут заменить часы нудной работы по копированию и вставке изображений вручную.

На практике код для создания эффектов анимации, звукового сопровождения, диапазонов фигур и т. п. может оказаться очень сложным. Найти в документации то, что вам нужно, не так-то просто. Рекомендуется для получения "наводящих указаний" активно использовать макрорекордер и анализировать созданный им код. Однако макрорекордер часто выбирает какие-то очень нетривиальные способы выполнения различных действий. Например, для вставки того же рисунка он предлагает использовать код типа:

```
ActiveWindow.Selection.SlideRange.Shapes.AddPicture ...
```

что, конечно, задачу не упрощает. Так что код макрорекордера всегда рекомендуется проверять и исправлять.

Задание для самостоятельной работы 14: Программное добавление элементов в слайды

ЗАДАНИЕ:

Создайте макрос в PowerPoint, который бы добавлял во все слайды активной презентации в правый нижний угол надпись "© Академия специальных курсов, 2005" (рис. 14.1).

Примечание

В реальной работе, возможно, удобнее было поместить этот макрос по внешнее приложение VBA, например, в документ Word или лист Excel, чтобы не копировать этот код для каждой новой презентации. Но в этой работе для простоты код будет выполняться из самого PowerPoint.

Рис. 14.1. Часть слайда с копирайтом

Ответ к заданию 14

Код макроса может быть, например, таким:

```
Public Sub InsertCopyRight()
    Dim oSlide As Slide
    Dim oShape As Shape

    'Проходим циклом по всем слайдам в презентации
    For Each oSlide In Application.ActivePresentation.Slides

        'Для каждого слайда создаем надпись (в виде объекта Shape)
        'Нужные значения для числовых параметров Left, Top, Width и
        'Height находим подбором или через макрорекордер
        Set oShape = oSlide.Shapes.AddTextbox(_
            msoTextOrientationHorizontal, 500, 510, 210, 40)

        'Доступ к тексту через вложенные объекты TextFrame и TextRange
        oShape.TextFrame.TextRange.Text = Chr(169) & _
            " Академия специальных курсов, 2005"

        'Занимаемся украшательством
        oShape.TextFrame.TextRange.Font.Size = 12
        oShape.TextFrame.TextRange.Font.Bold = msoTrue
    Next

End Sub
```