Visual Basic
简明教程

计 算 机 教 程
青 苹 果 电 子 图 书 系 列

Visual Basic 简明教程

石　磊　　编著
张　晋　　审校

前　言

Visual Basic 是开发 Windows 应用程序最为强大的工具之一，利用它可以开发出功能强大的 Windows 应用程序，无论是 Windows 程序设计的初学者，还是有经验的 Windows 程序员，利用 Visual Basic 都可以迅速地开发出自己满意的应用程序。

为了满足广大读者的愿望，迅速地掌握 Windows 环境下的 Visual Basic 编程方法，作者编写本书。书中融合了利用 Visual Basic 开发 Windows 下应用程序的理论和实践，全面和深入地介绍了利用 Visual Basic 开发应用程序的常用方法和技巧，由于书中附带了很多的程序开发实例，所以实用性很强，书中各章的结构安排如下所示：

第一章，介绍了利用 Visual Basic 编制应用程序的一些基础知识；

第二章，学习使用 Visual Basic 中的可视化控件编程，这是读者必须掌握的内容；

在 Visual Basic 的程序设计过程中，对话框是程序与用户最直接的交互方式，所以在第三章中讲解对话框的编程技巧；

第四章，向读者介绍了 Visual Basic 中常用窗体的设计技巧；

第五章，介绍了 Visual Basic 中的文件处理技术；

第六章，笔者用一章的篇幅向读者介绍多媒体程序设计，其中包括动画程序制作，如何制作动画播放器等；

数据库设计和开发是 Visual Basic 非常吸引人的地方，第七章重点讲解数据库程序开发，让用户更好学习开发这方面的内容；

在 Visual Basic 程序设计中，只需要简单地设置控件属性就可以做到与网络的接口操作，在本书的第八章，读者可以学习到网络应用方面的内容。

作为 Windows 操作系统中不可缺少的一部分，API 函数在 Visual Basic 中的作用在一段时间内仍然不可替代，在第九章中比较详细说明了 Visual Basic 中调用 API 函数的常用方法与技巧；

全书融合了利用 Visual Basic 开发 Windows 下应用程序的理论和实践，全面深入地介绍了利用 Visual Basic 开发应用程序的常用方法和技巧。在本书的编著过程中，我们尽量注意减少冗长无味的说明，代之以具体实用的例题演示。通过例题，引导读者把握 Visual Basic 的精髓所在。

全书各章均含有大量的示例，包括丰富的程序代码、事物名称、图形图片、数据信息等，内容如有雷同，纯属巧合，未经许可，不得沿用。最后，限于作者水平，书中的错误和不足之处在所难免，竭诚欢迎广大读者对本书提出批评和建议。

编　者

内 容 提 要

Visual Basic 是 Windows 98/2000、Windows NT 应用程序的新一代可视化开发环境。本书是在面向对象编程思想的基础上，介绍了 Visual Basic 中 VCL 控件使用技巧、图像和文件处理、多媒体程序设计、窗体与对话框编程技巧、API 函数的调用、数据库设计开发、网络应用以及 Acitve X 技术高级编程等方面的内容。

全书内容简洁明了，使用简明的语言并辅之以实例，介绍使用 Visual Basic 进行程序设计的各种方法和常用技巧。本书中的程序都是笔者在实践中总结提炼出来的，具有很强的实用性。

本书适合各个级别的用户使用，主要作为 Visual Basic 课程的培训教材，也可作为专业程序人员的参考资料，对广大学习编程的爱好者也是一本很好的学习指南。

目　　录

第一章　编制简单的程序

随着编程概念的更新，可视化编程已经成为人们关注的焦点，由 Basic 发展而来的 Visual Basic 程序设计语言就是一种典型的可视化编程语言，由于它继承了 Basic 语言的所有优点，如简单、灵活等特点，同时又包括了面向对象等先进的程序设计技术，为用户提供了开发 Microsoft Windows 应用程序的最迅速、最简捷的方法。

利用 Visual Basic 程序设计语言，可以很方便的设计出在 Windows 环境下运行的应用程序来。本章，我们将带领读者学习最新版本的 Visual Basic 的新功能和新特性，并且掌握简单的 Windows 编程基础知识，为后面的学习打下基础。

1.1　Visual Basic 编程特点

Visual Basic 程序设计语言是基于 Windows 的一种高级程序设计语言，Visual Basic 语言的出现为 Windows 下的编程提出了一个新的概念，利用 Visual Basic 的动态数据交换、对象的链接和嵌入、动态链接库、ActiveX 技术和开放式数据库访问技术可以很方便的设计出功能强大的应用程序。下面，我们首先谈谈利用 Visual Basic 编程的几个特点：

1.1.1　可视化程序设计

在 Visual Basic 中开发的应用程序，不但有丰富的图形界面，同时由用户为开发图形界面而添加的代码真是少而又少，因为在 Visual Basic 中设计图形界面的过程中只需要设置 Active X 控件的属性值即可。如图 1-1 所示即为一个设计好的 Visual Basic 图形用户界面，在其中不用用户另行添加一行代码。

图 1-1　Visual Basic 的可视化界面

1.1.2　多任务

在传统的 MS-DOS 环境中，每次只能够执行一个任务，只有从一个任务中退出才能够执行下一个任务，这样在客观上就浪费了很多的资源，但是在 Windows 操作环境中，多个应用程序可以同时的运行，每个应用程序在屏幕上都有一个显示的窗口，如图 1-2 所示即为 Windows 的多任务环境。

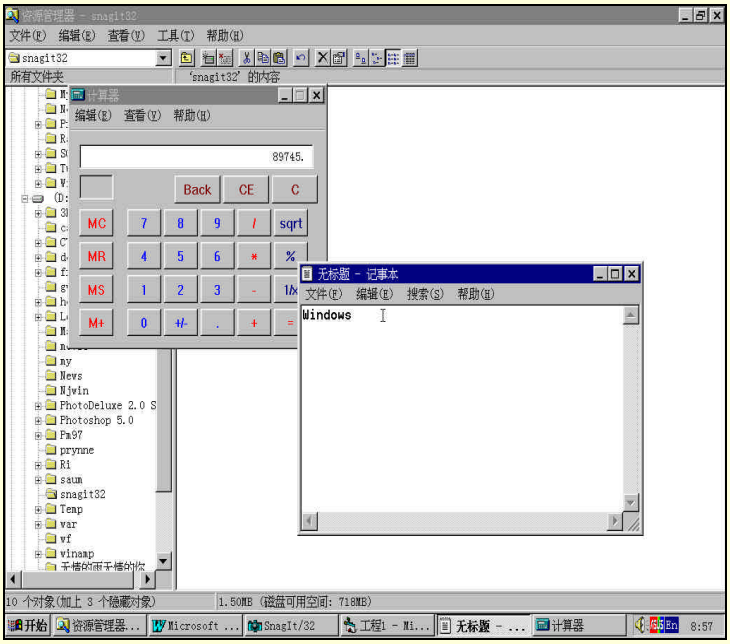


图 1-2 Windows 的多任务环境

1.1.3 资源共享

使用 Visual Basic 进行 Windows 编程，应用程序之间可以方便安全的实现共享资源。方式共有三种：剪贴板、DDE 和 OLE。

- **剪贴板可以把一个应用程序中的信息(文本、图形等)拷贝或者剪切下来，然后在切换到另外的应用程序中，把所要的信息粘贴到适当的位置；**
- **DDE 即是动态的数据交换技术，它的作用是在应用程序之间建立一条动态的数据交换的通道，使得应用程序在运行的过程中可以相互的交换信息；**

OLE 即是对象的嵌入和链接技术，与 DDE 不同，它不是在应用程序之间建立一个桥梁，而是把每个应用程序都看作是一个对象，通过对象之间的相互协作和协议来共同的完成任务。

1.1.4 数据库和网络功能

随着 Visual Basic 语言的向前发展，在数据库和网络方面的功能优势就愈加明显，利用 Visual Basic 中的 ODBC——开放式的数据库访问技术可以很方便的开发出自己的数据库应用程序，利用 Visual Basic 自带的可视化数据管理器和报表生成器，完全可以在 Visual Basic 就完成数据库的开发工作。

如图 1-3 所示即为一个开发的数据库应用程序。

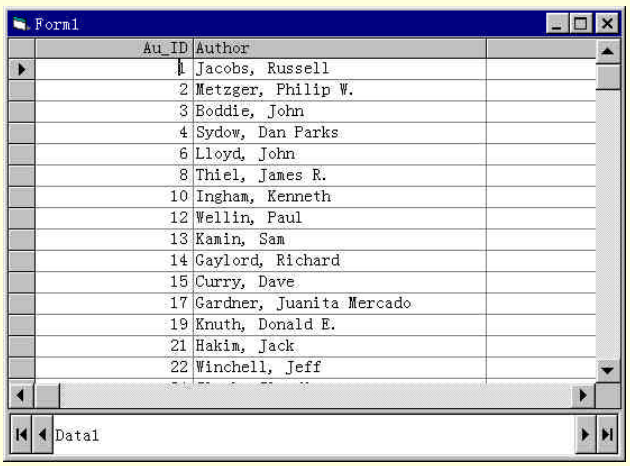


图 1-3 利用数据控件开发的应用程序

在 Visual Basic 中可以自己独立的开发 Active X 控件，而且可以制作独立在 Internet 上发行的 Active X 文档，在本书中就开发了一个“网络计算器”的应用程序，读者可以在后面学习到。

1.1.5 其他特性

在 Visual Basic 以前的版本中，由于仍然摆脱不了解释执行的代码运行机制，所以在相当的程度上制约了 Visual Basic 的发展。

Visual Basic 6.0 版本开始，在 Visual Basic 中制作的应用程序都改变为编译执行，使得 Visual Basic 的代码效率有了很大的提高，同时执行的速度也加快了 30%（同 Visual Basic 5.0 相比）。

当然在 Visual Basic 中的特性还不止这些，还有其他众多的特性，例如:

- 面向对象的编程语言；
- 结构化程序设计；
- 事件驱动的程序设计；
- 支持动态链接库；
- 应用程序之间的资源共享。

对于这些我们就不加赘述了，读者可以在后面的学习中深刻体会到。

1.2 Visual Basic 的编程环境

想要学习 Visual Basic 的可视化编程，在熟悉 Visual Basic 语法结构的基础上，熟练的利用 Visual Basic 的集成开发环境也是十分必要的。

如图 1-4 所示为 Visual Basic（6.0 版本）启动后的初始画面。

在 Visual Basic 启动后的初始画面中包含有工具栏、工具箱、工程资源窗口和属性窗口，下面将分别的加以介绍。

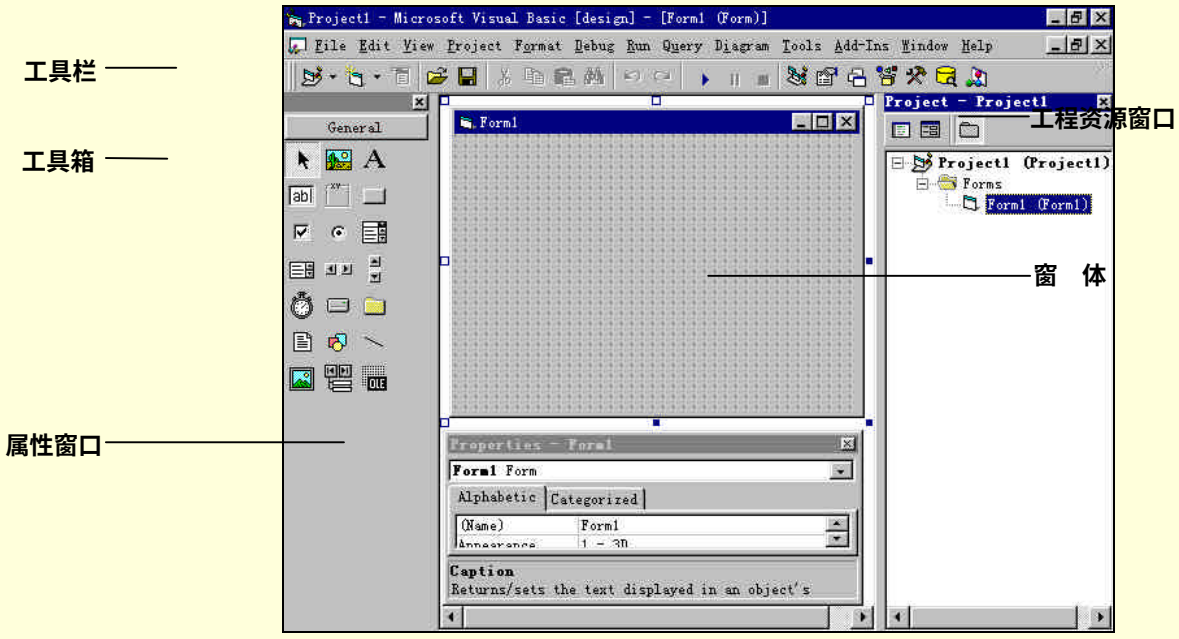


图 1-4 Visual Basic 6.0 的初始画面

1.2.1 工具栏

如图 1-5 所示为在 Visual Basic 的集成开发环境中的标准工具栏。

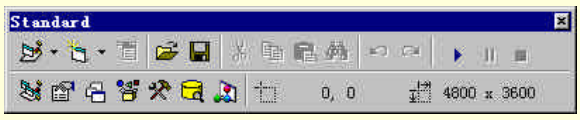


图 1-5 工具栏

工具栏上各个按钮及其说明如表 1-1 所示。

表 1-1 工具箱中常用快捷键

按钮图标	功能说明	按钮图标	功能说明
	Add Standard EXE Project		Start
	Add Form		Break
	Menu Editor		End
	Open Project		Project Explorer
	Save Project		Properties Window
	Cut		Form Layout Window
	Copy		Object Browser

续 表

按钮图标	功能说明	按钮图标	功能说明
	Paste		Toolbox
	Find		Data View Window
	Undo		Visual Component Manager
	Redo		

在 Visual Basic 的集成开发环境中还可以自己定制工具栏，选择菜单 View/Toolbars/ Customize，就会弹出一个如图 1-6 所示的定制工具栏的对话框。

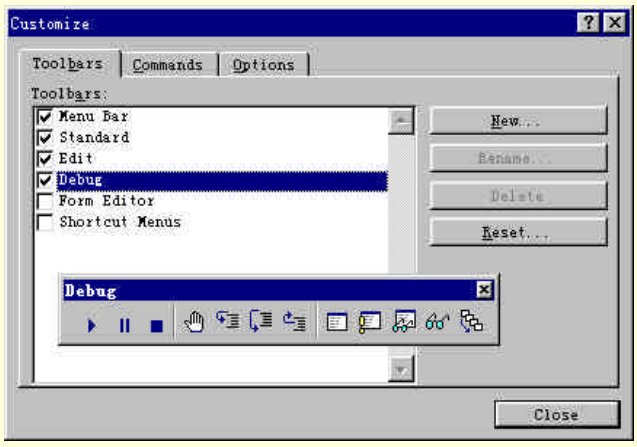


图 1-6 自定义工具栏

1.2.2 工具箱

为了方便用户的编程，Visual Basic 把常用的控件集中的放置在工具箱上，而不是特别常用的控件则不会出现在缺省的工具箱上。

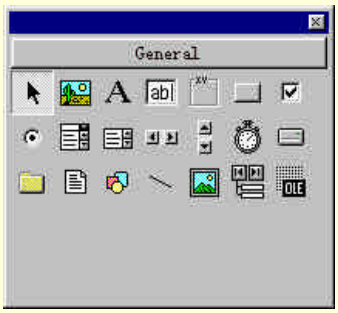


图 1-7 缺省工具箱

在设计阶段，通过使用一些快捷键可以提高编程的效率如表 1-2 所示。

表 1-2 工具箱中常用快捷键

快捷键	功 能
ENTER	把所选控件放到活动窗体中
DOWN ARROW	选择与已选工具同一列中下一个工具
UP ARROW	选择与已选工具同一列中上一个工具
LEFT ARROW	选择已选工具左边的工具
RIGHT ARROW	选择已选工具右边的工具
TAB	从左到右依次通过工具箱
SHIFT+TAB	从右到左依次通过工具箱
END	选择工具箱的最后一个工具
HOME	选择指针工具
ALT+F4	关闭工具箱

同样可以把一些自己需要的控件随时的加载到工具箱上，选择菜单 Project/Components，就会弹出如图 1-8 所示的对话框。

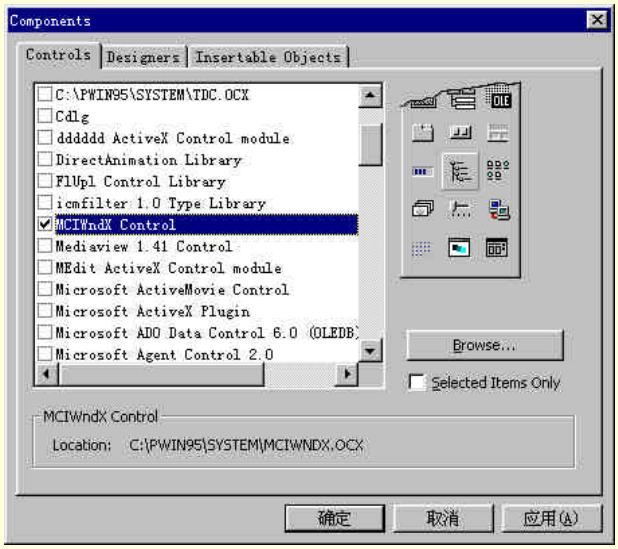


图 1-8 添加控件对话框

例如要添加一个 MCIWndX 控件，就可以在添加控件的对话框中选择 MCIWndX Control 项，单击“确定”按钮，就向缺省的工具箱中添加了 MCIWndX 控件。

1.2.3 代码窗口

Visual Basic 中的代码窗口是输入程序代码和程序调试的地方，在程序设计的任何阶段双击窗体就会弹出如图 1-9 所示的代码窗口。

它是由以下几个部分组成的：

- 标题条，用来显示应用程序的项目名称和窗体名称；

- 对象框，用来显示和选择窗体中现有的对象列表和当前正在编辑的对象名；
- 事件框，用来显示和选择对象可用的事件；
- 代码框，用来输入代码。

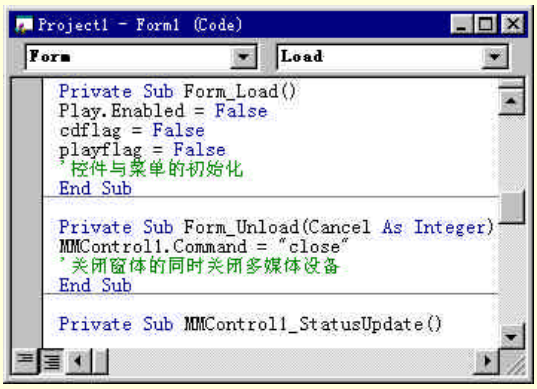


图 1-9 代码窗口

在代码窗口中通过使用快捷键可以快速的访问下列命令,从而提高了编程的效率,在代码窗口中常用的快捷键如表 1-3 所示。

表 1-3 代码窗口中常用的快捷键

快捷键	功 能
F7	查看代码窗口
F2	查看“对象浏览器”
CTRL+F	查找
CTRL+H	替换
F3	查找下一个
SHIFT+F3	查找前一个
CTRL+DOWN ARROW	下一个过程
CTRL+UP ARROW	前一个过程
SHIFT+F2	查看定义
CTRL+PAGE DOWN	到下一个屏幕
CTRL+PAGE UP	到上一个屏幕
CTRL+SHIFT+F2	到以前光标处
CTRL+HOME	到模块开头处
CTRL+END	到模块结尾处
CTRL+RIGHT ARROW	右移一词
CTRL+LEFT ARROW	左移一词
CTRL+Y	删除当前行
CTRL+DELETE	删至行尾
CTRL+SHIFT+F9	清除所有断点
CTRL+F2	切换书签
SHIFT+F10	查看快捷键菜单

1.2.4 项目窗口

项目窗口有时也称作工程资源窗口，它主要是用来显示项目文件中所包含的所有文件，如图 1-10 所示即为一个典型的项目窗口。

通常项目窗口由以下几个部分组成：

- 标题条，用来显示项目名称；
- View Project 按钮，用来显示窗体的显示界面；
- View Code 按钮，用来显示程序代码；
- 文件列表框，用来显示项目中所包含的所有文件。

图 1-10 项目窗口

在项目窗口中也有一些快捷键供用户使用，如表 1-4 所示。

表 1-4 项目窗口中常用的快捷键

快捷键	功 能
ENTER	从列表中打开选定文件，或是展开及折叠列表以显示下级项目
SHIFT+ENTER	为选定文件打开代码窗口
F7	为选定文件打开代码窗口
SHIFT+F10	查看快捷键菜单
HOME	选定列表中的第一个文件
END	选定列表中最后一个文件
UP ARROW	向上移一个项目
DOWN ARROW	向下移一个项目

1.2.5 属性窗口

属性窗口就是用来描述对象属性的窗口，用户可以在属性窗口中来对对象进行设置和修改，如图 1-11 所示.

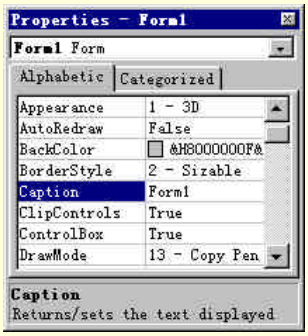


图 1-11 属性窗口

一个典型的属性窗口由以下几个部分组成：

- 标题条，显示项目名称和窗体名称；
- 对象列表框，用来显示和选择当前窗体中所有对象；
- 设置框，用户可以在其中设置和修改对象的属性设置。

在属性窗口有一些快捷键如表 1-5 所示。

表 1-5 属性窗口中常用的快捷键

快捷键	功 能
PAGE DOWN	在属性列表中下移
PAGE UP	在属性列表中上移
DOWN ARROW	向下移一个属性
UP ARROW	向上移一个属性
RIGHT ARROW	向下移一个属性
LEFT ARROW	向上移一个属性
END	到列表中最后一个属性
HOME	到列表中第一个属性

ALT+F6	在最后两个活动窗口之间切换
TAB	在属性、属性设置框及对象框中移动插入点
CTRL+SHIFT+ALPHA	移动到列表中以该字母开头的属性

属性窗口的显示方式有两种，即属性按字母顺序方式排列和属性按照分类的方式排列，如图 1-12 所示为按照分类次序排列的属性。

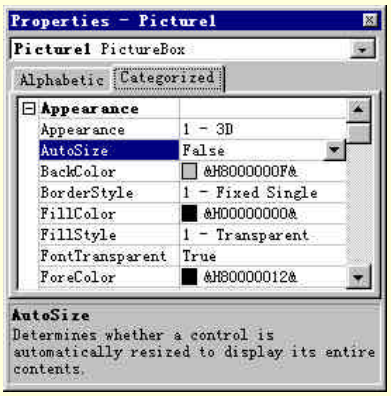


图 1-12 属性按分类排列

1.3 制作第一个程序

下面就以一个简单的示例程序来说明，在 Visual Basic 6.0 中进行应用程序开发的一般步骤和基本的方法，为以后的程序设计打一个基础。

编制示例程序的基本步骤如下所示：

1. 开始工作

首先要激活 Visual Basic 6.0 应用程序，在 Visual Basic 6.0 的集成开发环境中用鼠标选择“文件”菜单中的“新建工程”选项，在屏幕上就会弹出一个如图 1-13 所示的“新建工程”对话框。

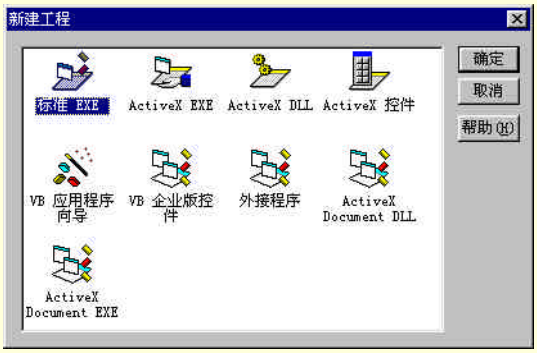


图 1-13 “新建工程”对话框

在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 6.0 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form Form1
    Caption         =   "Hello Visual Basic"
    ClientHeight    =   3195
    ClientLeft      =   60
    ClientTop       =   345
    ClientWidth     =   4680
    ScaleHeight     =   3195
```

```
    ScaleWidth      =   4680
    StartUpPosition =   2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- **窗体的标题栏中显示文本“**Hello Visual Basic**”；**
- **窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮。**

2．**添加控件**

在程序设计的过程中，向当前空白的窗体添加两个 CommandButton **控件和两个** TextBox **控件，它们的作用在后面会加以说明，添加控件后的窗体如图** 1-14 **所示。**

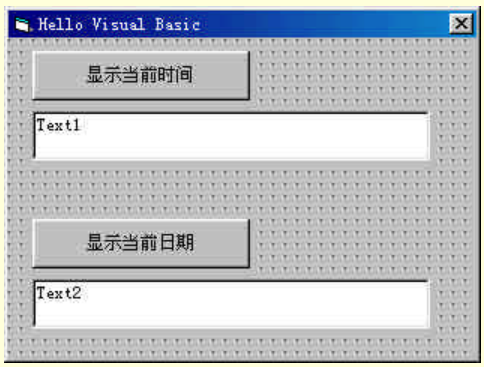


图 1-14　**添加控件后的窗体**

各个控件的属性设置如下所示：

```
Begin Visual Basic.TextBox Text2
Height      =     495
Left  =     360
Text  =     "Text2"
Top   =     2280
Width       =     2295
End
Begin Visual Basic.TextBox Text1
Height      =     495
Left  =     360
Text  =     "Text1"
Top   =     960
Width       =     2295
End
Begin Visual Basic.CommandButton Command2
Caption     =     "显示当前日期："
Height      =     495
Left  =     360
Top   =     1680
Width       =     1815
End
Begin Visual Basic.CommandButton Command1
Caption     =     "显示当前时间："
Height      =     495
Left  =     360
Top   =     360
```

```
Width     =     1815
End
End
```

经过以上属性设置后的 VCL **控件的特性及作用如下所示：**

- Text1 **控件：在程序运行的过程中充当系统时间显示的容器；**
- Text2 **控件：在程序运行的过程中充当系统日期显示的容器；**
- Command1 **控件：激活控件** Text1 **显示系统时间；**
- Command2 **控件：激活控件** Text2 **显示系统日期。**

3. **程序初始化**

经过上面的界面设置，下面我们开始进行程序的初始化工作。

注意：

> **在本示例程序中，所谓程序初始化，指的就是为窗体** Private Sub Form_Load()**事件和窗体声明段中所添加的响应代码。**

在程序设计的过程中，用鼠标左键双击窗体上的空白处，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的事件列表中选择窗体 Form**，在对应的事件列表中选择事件** Private Sub Form_Load()**，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：**

```
Private Sub Form_Load()
Form1.Text1.Text = Time
Form1.Text2.Text = Date
'显示时间和日期
End Sub
```

程序说明：

在程序运行的初期，事件 Private Sub Form_Load()**中的代码就会被执行，程序通过** Form1.Text1.Text = Time、Form1.Text2.Text = Date **两条语句在窗体的** Text1 **和** Text2 **中显示程序运行初期的系统时间和日期。**

4. **响应按钮操作**

在示例程序运行的过程中，如果用户在两个按钮上按下鼠标的左键，在对应的文本框中就会显示当前系统的时间和日期。

为此，在程序设计的过程中，用鼠标左键双击窗体上的两个按钮控件，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中分别选择按钮控件 Command1 **和** Command2**，在对应的事件列表中选择事件** Private Sub Command1_Click()**和** Private Sub Command2_Click()**。**

把光标移动到相应事件的处理过程中，添加如下所示的事件响应代码：

```
Private Sub Command1_Click()
Form1.Text1.Text = Time
'显示当前时间
End Sub

Private Sub Command2_Click()
Form1.Text2.Text = Date
'显示当前日期
End Sub
```

程序说明：

在程序运行的过程中，当用户在按钮控件 Command1 **上按下鼠标左键时，就会激活控件的** Private Sub Command1_Click()**事件。**

程序通过 Form1.Text1.Text = Time **语句在控件** Text1 **上显示系统当前时间，如果按钮控件** Command2 **上按下**

鼠标左键时，就会激活控件的 Private Sub Command2_Click()事件，程序通过 Form1.Text2.Text = Date 语句在控件 Text2 上显示系统当前日期。

5. 运行程序

按照附后的源程序清单添加剩余的代码，设置项目的启动窗体为主窗体 Form1，存储文件，运行程序，程序运行结果如图 1-15 所示。

图 1-15 程序运行结果

单击按钮“显示当前时间”，在界面的第一个文本框中就会出现计算机系统的当前时间。第二个按钮、文本框的功能和前者类似。

提示：

运行编辑好的 Visual Basic 程序有两种方式，最简捷的方法就是直接按键盘上的 F5 功能键。

附程序完整程序清单如下所示：

程序清单

```
Begin Visual Basic.Form Form1
    Caption         =   "Hello Visual Basic"
    ClientHeight    =   3195
    ClientLeft      =   60
    ClientTop       =   345
    ClientWidth     =   4680
    ScaleHeight     =   3195
    ScaleWidth      =   4680
    StartUpPosition =   2
    Begin Visual Basic.TextBox Text2
        Height      =   495
        Left        =   360
        TabIndex    =   3
        Text        =   "Text2"
        Top         =   2280
        Width       =   2295
    End
    Begin Visual Basic.TextBox Text1
        Height      =   495
        Left        =   360
        TabIndex    =   2
```

```
        Text            =   "Text1"
        Top             =   960
        Width           =   2295
    End
    Begin Visual Basic.CommandButton Command2
        Caption         =   "显示当前日期："
        Height          =   495
        Left            =   360
        TabIndex        =   1
        Top             =   1680
        Width           =   1815
    End
    Begin Visual Basic.CommandButton Command1
        Caption         =   "显示当前时间："
        Height          =   495
        Left            =   360
        TabIndex        =   0
        Top             =   360
        Width           =   1815
    End
End
Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False
Private Sub Command1_Click()
Form1.Text1.Text = Time
'显示当前时间
End Sub
Private Sub Command2_Click()
Form1.Text1.Text = Date
'显示当前日期
End Sub
Private Sub Form_Load()
Form1.Text1.Text = Time
Form1.Text1.Text = Date
'显示时间和日期
End Sub
```

1.4 快速生成应用程序

在 Visual Basic 6.0 **中系统为用户提供了许多个向导应用程序，利用这些向导可以节省很多的劳动，可以少用甚至不用添加代码就可以快速生成应用程序。**

下面就利用 Visual Basic **应用程序向导来生成一个简单的应用程序，具体的程序设计步骤如下所示。**

1．开始工作

首先启动一个新的项目，选择“文件”菜单中的“新建工程”选项，在 Visual Basic 6.0 的集成开发环境中就会弹出一个标题为“新建工程”的对话框，用户在其中可以选择各种向导程序和窗体形式等。

选择“新建工程”对话框中的“Visual Basic 应用程序向导”，单击“确定”按钮进入下一步。

接下来就会显示一个“应用程序向导--介绍”对话框，在其中可以读取以前存储过的设置，单击“下一步”按钮。

注意：

✍ 以上对话框基本不用任何设置，所以介绍略国，也不作图解。

2．选择界面类型

在由向导生成的应用程序中，用户可以在如图 1-16 所示的窗体界面类型选择对话框中选择界面的类型。

图 1-16　窗体界面类型选择对话框

在窗体界面类型选择对话框中用户可以为自己的应用程序选择一种界面类型，如 MDI（多文档界面）、SDI（单文档界面）或者是资源管理器样式等。

这里，用户可以选择界面类型为 MDI，单击“下一步”按钮。

注意：

✍ MDI、SDI 窗体界面设计是 Visual Basic 6.0 可视化编程的重要内容，我们在稍后一章将详细介绍。

3．选择菜单

在如图 1-17 所示的菜单选择对话框中，用户可以选择文件菜单、编辑菜单、视图菜单、窗口菜单和帮助菜单以及其中的子菜单项。

图 1-17　菜单选择对话框

4．设计完毕

接下来，用户只要按照向导的提示进行操作就可以了，不再赘述。

最后经过以上各步，完成的应用程序窗体如图 1-18 所示。

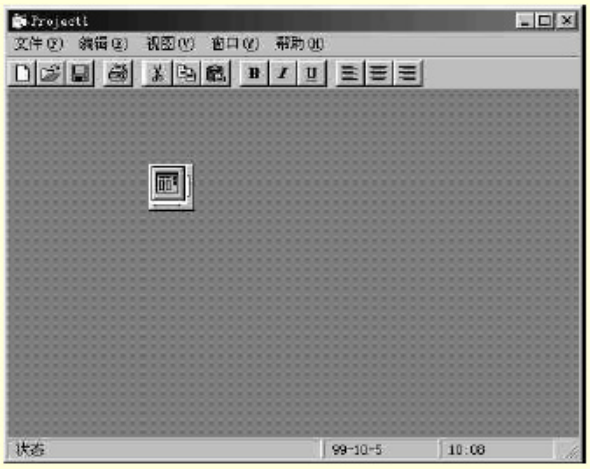


图 1-18 选择程序路径和设置

在这里，我们没有填写一行代码的应用程序中，但是已经包括了菜单、工具栏和状态条等 Windows 应用程序的标准设置了，而且还具有一定的功能。

5. 运行程序

做完以上的工作后，选择 Visual Basic 6.0 工具栏上的“保存”按钮，在弹出的对话框中选择合适的文件名保存文件，然后按键盘上的功能键 F5 运行程序。

程序的运行的结果如图 1-19 所示。

图 1-19 程序运行结果

在这个应用程序中，可以完成新建文件、打开指定文件、存储文件，对文本文件进行编辑等操作。但是由于还没有添加各个事件的详细响应代码，所以它的某些功能还不能够实现。

注意：

- 总的来说，上面我们生成的是一个标准的 Windows 程序框架，用户可以在此基础上添加自己的代码。

由向导生成的程序完整源代码如下所示：

程序清单

```
Private Sub MDIForm_Load()
    Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
    Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
    Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
    Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)
    LoadNewDoc
```

```
End Sub

Private Sub LoadNewDoc()
    Static lDocumentCount As Long
    Dim frmD As frmDocument

    lDocumentCount = lDocumentCount + 1
    Set frmD = New frmDocument
    frmD.Caption = "Document " & lDocumentCount
    frmD.Show
End Sub

Private Sub MDIForm_Unload(Cancel As Integer)
    If Me.WindowState <> vbMinimized Then
        SaveSetting App.Title, "Settings", "MainLeft", Me.Left
        SaveSetting App.Title, "Settings", "MainTop", Me.Top
        SaveSetting App.Title, "Settings", "MainWidth", Me.Width
        SaveSetting App.Title, "Settings", "MainHeight", Me.Height
    End If
End Sub

Private Sub mnuHelpAbout_Click()
    'To Do
    MsgBox "在此处添加"关于"对话框代码！"
End Sub

Private Sub mnuViewOptions_Click()
    'To Do
    MsgBox "在此处添加"选项"对话框代码！"
End Sub

Private Sub mnuViewStatusBar_Click()
    If mnuViewStatusBar.Checked Then
        sbStatusBar.Visible = False
        mnuViewStatusBar.Checked = False
    Else
        sbStatusBar.Visible = True
        mnuViewStatusBar.Checked = True
    End If
End Sub

Private Sub mnuViewToolbar_Click()
    If mnuViewToolbar.Checked Then
        tbToolBar.Visible = False
        mnuViewToolbar.Checked = False
    Else
```

```
            tbToolBar.Visible = True
            mnuViewToolbar.Checked = True
        End If
End Sub

Private Sub tbToolBar_ButtonClick(ByVal Button As ComctlLib.Button)
        Select Case Button.Key
            Case "New"
                LoadNewDoc
            Case "New"
                mnuFileNew_Click
            Case "Open"
                mnuFileOpen_Click
            Case "Save"
                mnuFileSave_Click
            Case "Print"
                mnuFilePrint_Click
            Case "Cut"
                mnuEditCut_Click
            Case "Copy"
                mnuEditCopy_Click
            Case "Paste"
                mnuEditPaste_Click
            Case "Bold"
                'To Do
                MsgBox "在此处添加"粗体"代码！"
            Case "Italic"
                'To Do
                MsgBox "在此处添加"斜体"代码！"
            Case "Underline"
                'To Do
                MsgBox "在此处添加"下划线"代码！"
            Case "Left"
                'To Do
                MsgBox "在此处添加"左对齐"代码！"
            Case "Center"
                'To Do
                MsgBox "在此处添加"居中对齐"代码！"
            Case "Right"
                'To Do
                MsgBox "在此处添加"右对齐"代码！"
        End Select
End Sub

Private Sub mnuHelpContents_Click()
        Dim nRet As Integer
```

```
    '如果这个工程没有帮助文件，显示消息给用户
    '可以在"工程属性"对话框中为应用程序设置帮助文件
    If Len(App.HelpFile) = 0 Then
        MsgBox "无法显示帮助目录，该工程没有相关联的帮助。", vbInformation, Me.Caption
    Else
        On Error Resume Next
        nRet = OSWinHelp(Me.hwnd, App.HelpFile, 3, 0)
        If Err Then
            MsgBox Err.Description
        End If
    End If
End Sub

Private Sub mnuHelpSearch_Click()
    Dim nRet As Integer
    '如果这个工程没有帮助文件，显示消息给用户
    '可以在"工程属性"对话框中为应用程序设置帮助文件
    If Len(App.HelpFile) = 0 Then
        MsgBox "无法显示帮助目录，该工程没有相关联的帮助。", vbInformation, Me.Caption
    Else
        On Error Resume Next
        nRet = OSWinHelp(Me.hwnd, App.HelpFile, 261, 0)
        If Err Then
            MsgBox Err.Description
        End If
    End If
End Sub

Private Sub mnuWindowArrangeIcons_Click()
    Me.Arrange vbArrangeIcons
End Sub

Private Sub mnuWindowCascade_Click()
    Me.Arrange vbCascade
End Sub
Private Sub mnuWindowNewWindow_Click()
    'To Do
    MsgBox "此处添加"新建窗口"代码！"
End Sub

Private Sub mnuWindowTileHorizontal_Click()
    Me.Arrange vbTileHorizontal
End Sub

Private Sub mnuWindowTileVertical_Click()
    Me.Arrange vbTileVertical
```

```
End Sub

Private Sub mnuViewRefresh_Click()
    'To Do
    MsgBox "此处添加"刷新"代码！"
End Sub

Private Sub mnuEditCopy_Click()
    'To Do
    MsgBox "此处添加"复制"代码！"
End Sub

Private Sub mnuEditCut_Click()
    'To Do
    MsgBox "此处添加"剪切"代码！"
End Sub

Private Sub mnuEditPaste_Click()
    'To Do
    MsgBox "此处添加"粘贴"代码！"
End Sub

Private Sub mnuEditPasteSpecial_Click()
    'To Do
    MsgBox "此处添加"特殊粘贴"代码！"
End Sub

Private Sub mnuEditUndo_Click()
    'To Do
    MsgBox "此处添加"撤消"代码！"
End Sub

Private Sub mnuFileOpen_Click()
    Dim sFile As String
    With dlgCommonDialog
        'To Do
        '设置 common dialog 控件的标志和属性
        .Filter = "所有文件 (*.*)|*.*"
        .ShowOpen
        If Len(.filename) = 0 Then
            Exit Sub
        End If
        sFile = .filename
    End With
    'To Do
    '处理打开的文件
```

```
End Sub

Private Sub mnuFileClose_Click()
    'To Do
    MsgBox "此处添加"关闭"代码！"
End Sub

Private Sub mnuFileSave_Click()
    'To Do
    MsgBox "此处添加"保存"代码！"
End Sub

Private Sub mnuFileSaveAs_Click()
    'To Do
    '在调用 ShowSave 之前设置 common dialog 控件
    dlgCommonDialog.ShowSave
End Sub

Private Sub mnuFileSaveAll_Click()
    'To Do
    MsgBox "此处添加"全部保存"代码！"
End Sub

Private Sub mnuFileProperties_Click()
    'To Do
    MsgBox "此处添加"属性"代码！"
End Sub

Private Sub mnuFilePageSetup_Click()
    dlgCommonDialog.ShowPrinter
End Sub

Private Sub mnuFilePrintPreview_Click()
    'To Do
    MsgBox "此处添加"打印预览"代码！"
End Sub

Private Sub mnuFilePrint_Click()
    'To Do
    MsgBox "此处添加"打印"代码！"
End Sub

Private Sub mnuFileSend_Click()
    'To Do
    MsgBox "此处添加"发送"代码！"
End Sub
```

```
Private Sub mnuFileMRU_Click(Index As Integer)
    'To Do
    MsgBox "此处添加"MRU"代码！"
End Sub

Private Sub mnuFileExit_Click()
    '卸载窗体
    Unload Me
End Sub

Private Sub mnuFileNew_Click()
    LoadNewDoc
End Sub
   '模块文件中的响应代码
Private Sub Form_Load()
    Form_Resize
End Sub

Private Sub Form_Resize()
    On Error Resume Next
    txtText.Move 100, 100, Me.ScaleWidth - 200, Me.ScaleHeight - 200
End Sub
```

1.5 小 结

本章通过几个简单的应用程序示例揭开了 Visual Basic 6.0 的神秘面纱，让用户对 Visual Basic 6.0 下的可视化编程有一个感性认识，为今后的学习打下坚实的基础。

通过本章的学习，读者基本上可以领会 Visual Basic 6.0 到底是干什么的，如何学习它。但是，更加深入的东西还是需要读者在编程实践中自己去体会，在今后的学习中自己去领悟，在此就不加赘述了。

第二章 常用可视化控件的使用

想要学习 Viusal Basic 的可视化编程，在熟悉 Viusal Basic 语法结构的基础上，熟练的利用 Viusal Basic 中常用的控件是非常必要的，所以本章我们将带领读者来学习 Visual Basic 6.0 中的这些基本控件。

Viusal Basic 为了方便用户开发应用程序，提供了很多的可视化控件，在本章只是介绍一些常用的基本控件，至于高级的 Active X 控件的应用，在随后的各章中会逐一的加以介绍，在此就不加叙述了。

2.1 使用图片框控件

图片框（PictureBox）控件可以显示来自位图、图标或者元文件，以及来自增强的元文件、JPEG 或 GIF 文件的图形。如图 2-1 所示即为一个加载有位图文件的 PictureBox 控件。

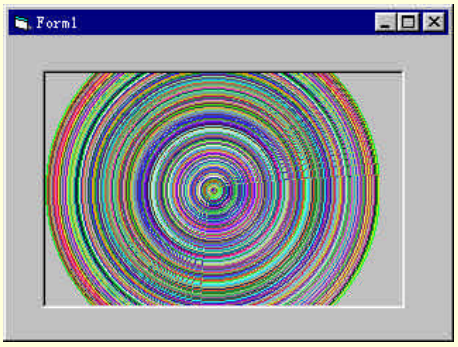


图 2-1 加载位图的控件

PictureBox 控件在 Viusal Basic 的图形图像处理中占有很重要的地位，同时它本身也有很多独特的属性、事件和方法，下面就对 PictureBox 控件的特殊属性和方法作一个系统的介绍。

2.1.1 Align 属性

Align 属性返回或设置一个值，确定对象是否可在窗体上以任意大小、在任意位置上显示，或是显示在窗体的顶端、底端、左边或右边，而且自动改变大小以适合窗体的宽度。

它的语法结构如下：

object.Align [= number]

其中 Align 属性的设置值及其说明如表 2-1 所示。

表 2-1 Align 属性的设置

设 置	数字值	说 明
VbAlignNone	0	无，可以在设计时或在程序中确定大小和位置
VbAlignTop	1	顶部，对象显示在窗体的顶部，其宽度等于窗体的 ScaleWidth 属性设置值。
VbAlignBottom	2	底部，对象显示在窗体的底部，其宽度等于窗体的 ScaleWidth 属性设置值
VbAlignLeft	3	左边，对象在窗体的左面，其宽度等于窗体的 ScaleWidth 属性设置值。
VbAlignRight	4	右边，对象在窗体的右面，其宽度为窗体的 ScaleWidth 属性设置值

如图 2-2 所示为 PictureBox 控件的 Align 属性属性设置为 4 时的显示情况，控件位于窗体的右侧，宽度为窗体的 ScaleWidth 属性设置值。

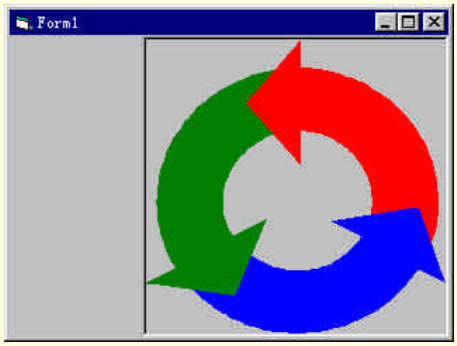


图 2-2 Align 属性属性设置为 4

下面的示例代码就是当程序运行时，在窗体上单击鼠标时，程序会自动的判断鼠标的按下位置，并且根据位置的不同来设置 PictureBox 控件的 Align 属性。用 Align 属性可以很快地在窗体的顶部或底部创建工具栏或状态栏。

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If X < Form1.Width / 2 Then
    If Y < Form1.Height / 2 Then
        Picture1.Align = 1
        控件显示在窗体的顶部
    Else
        Picture1.Align = 2
        控件显示在窗体的底部
    End If
Else
    If Y < Form1.Height / 2 Then
        Picture1.Align = 3
        控件显示在窗体的左面
    Else
        Picture1.Align = 4
        控件显示在窗体的右面
    End If
End If
End Sub
```

注意：

✍ 当 Align 值设置为 1 或 2 时，用户改变窗体的大小，控件会自动地改变大小以适合窗体的尺寸。

2.1.2 Appearance 属性

Appearance 属性用于设置窗体上 PictureBox 控件在设计时的绘图风格。

它有两个设置值如下所示：

0 平面绘制控件和没有可视化效果的窗体；

1 带有三维效果的绘制控件。

如图 2-3 所示即为 Appearance 属性设置为不同的值时的显示情况，其中左图为 Appearance 属性设置为 1，而右图为 Appearance 属性设置为 0。

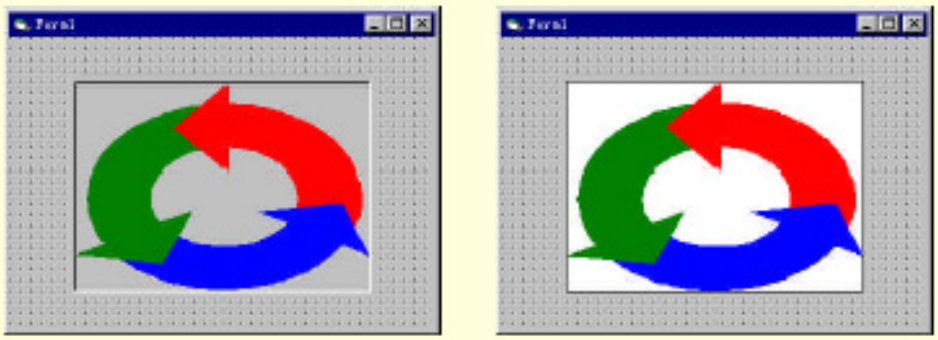


图 2-3 Appearance 属性设置

注意：

✍ Appearance 属性在运行时是只读的。

2.1.3 AutoRedraw 属性

AutoRedraw 属性用于设置控件从图形方法到持久图形的输出，也即用来设置控件是否具有自动重绘的功能。

它的语法结构如下：

object.AutoRedraw [= boolean]

其中 AutoRedraw 属性的设置值及其说明如下所示：

- True：PictureBox 控件的自动重绘有效，图形和文本输出到屏幕的同时存储在内存的图像中；
- False：使对象的自动重绘无效，且将图形或文本只写到屏幕上。

下面就以一个示例来说明 AutoRedraw 属性的应用，步骤如下：

1. 开始工作

首先启动一个新的项目，在空白的窗体上添加一个 PictureBox 控件，控件与窗体的属性设置如表 2-2 所示。

表 2-2 窗体与控件的属性设置

窗体或控件	属 性	设 置
窗体	(Name)	Form1
	AutoRedraw	False
	Caption	Form1
	Height	3600
	Left	0
	Top	0
	Width	4800
控件	(Name)	Picture1
	Align	0 - None
	AutoRedraw	False
	Height	3000
	Left	840
	Picture	(None)
	Top	120
	Width	3000

添加控件后的窗体如图 2-4 所示。

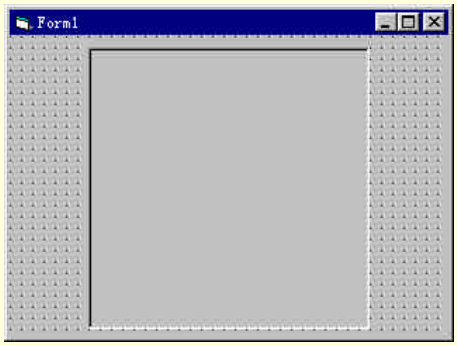


图 2-4 添加控件后的窗体

2. 添加代码

在窗体的设计阶段双击窗体，在弹出的代码窗口中找到窗体的 Form_Load()事件，并且在其中添加下列代码：

```
Private Sub Form_Load()
    Dim X, Y As Integer
    '声明坐标变量
    Dim r As Integer
    '声明半径变量
    X = Picture1.ScaleWidth / 2
    '设置圆心坐标 X 位置
    Y = Picture1.ScaleHeight / 2
    '设置圆心坐标 Y 位置
    For r = 0 To Picture1.ScaleWidth / 2
    '设置半径
      Picture1.Circle (X, Y, r, RGB(Rnd * 255, Rnd * 255, Rnd * 255) )
      '在控件上绘制圆
    Next r
    End Sub
```

程序说明：

在程序开始运行后，首先定义两个坐标变量和一个存储圆半径的变量，通过两条赋值语句设置圆心的坐标，然后通过一个循环语句来控制圆的半径，最后通过一条语句 Picture1.Circle (X, Y, r, RGB(Rnd * 255, Rnd * 255, Rnd * 255))来以随机的颜色在控件上绘制圆。

3. 运行程序

选择菜单 File 中的 Save 选项存储文件，按键盘上的功能键 F5 键运行程序，程序运行的初始画面如图 2-5 所示。

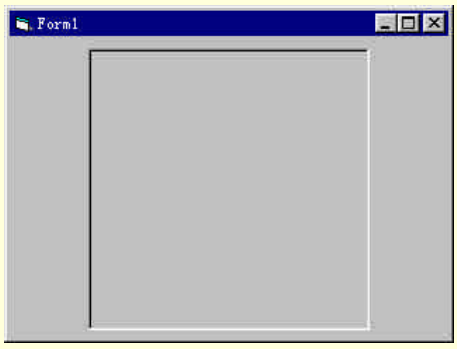


图 2-5 程序运行的初始画面

但是程序的运行结果并不像我们所预料的那样在控件上绘制一系列的圆，所以一定是程序代码或者控件的

设置有问题。

4. 修改代码

经过仔细的分析之后不难发现，控件的 AutoRedraw 属性设置为 False，这样图形就不能够正常的显示在控件上，所以修改代码如下：

```
Private Sub Form_Load()
    Dim X, Y As Integer
    '声明坐标变量
    Dim r As Integer
    '声明半径变量
    Picture1.AutoRedraw = True
    '设置 AutoRedraw 处于有效的状态
    X = Picture1.ScaleWidth / 2
    '设置圆心坐标 X 位置
    Y = Picture1.ScaleHeight / 2
    '设置圆心坐标 Y 位置
    For r = 0 To Picture1.ScaleWidth / 2
    '设置半径
      Picture1.Circle (X, Y), r, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
      '在控件上绘制圆
    Next r
    Picture1.AutoRedraw = False
    ' 关闭 AutoRedraw
End Sub
```

5. 运行程序

存储文件，重新运行程序，结果如图 2-6 所示。

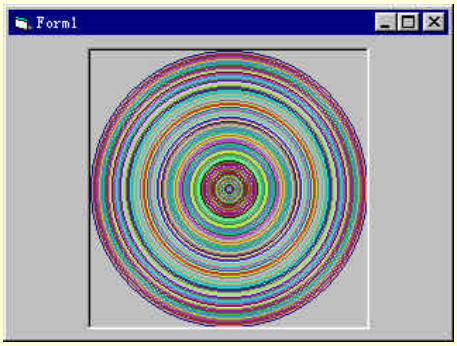


图 2-6 在控件上绘制圆

2.1.4 AutoSize 属性

AutoSize 属性用于设置控件是否具有自动改变大小的能力，以适应图像文件的尺寸，显示图像文件的全部内容。

它的语法结构如下：

```
object.AutoSize [= boolean]
```

其中 AutoSize 属性的设置值及其说明如下所示：

- True：自动改变控件大小以显示全部内容；
- False：保持控件大小不变，超出控件区域的内容被裁剪掉。

如图 2-7 所示即为 AutoSize 属性设置为 False 时的显示情况，在程序的运行过程中将保持控件的大小不变，

超出控件区域的内容被裁剪掉。

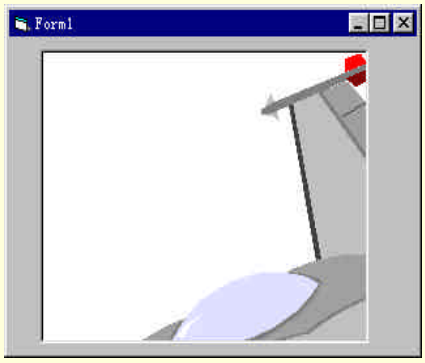


图 2-7 AutoSize 属性设置为 False

2.1.5 Image 属性

Image 属性用于返回持久图形的句柄，该句柄由 Microsoft Windows 运行环境提供，它的语法结构如下：

object.Image

其中 object 代表一个对象表达式。

下面就以一个示例程序来说明 PictureBox 控件中 Image 属性的应用，本示例的作用是在 PictureBox 控件上以随机色绘制出一系列同心圆，同时把绘制的图形保存到一个指定的位图文件中，步骤如下：

1. 开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加一个 PictureBox 控件，窗体及控件的属性设置如表 2-3 所示。

表 2-3 窗体与控件的属性设置

窗体或控件	属 性	设 置
窗体	(Name)	Form1
	BorderStyle	3 - Fixed Dialog
	Caption	Form1
	Height	3570
	Left	0
	Moveable	False
	Picture	(None)
	Top	0
	Width	4770
控件	(Name)	Picture1
	Align	0 - None
	AutoRedraw	True
	Height	3000
	Left	720
	Picture	(Metafile)
	Top	120
	Width	3000

这样设置的窗体具有如下的特性：

- 窗体在程序的运行过程中不能够改变大小；
- 运行程序，窗体始终位于屏幕的中央；
- 窗体在程序的运行过程中不能够移动。

如表 2-2 设置的 PictureBox 控件具有如下的特性：

- 控件的名称为 Picture1；
- 控件可以在设计时或在程序中确定大小和位置；
- 控件的自动重绘处于有效的状态；
- 控件加载一个元文件。

添加控件后的窗体如图 2-8 所示。

2．添加代码

首先来添加程序的初始化代码，在窗体的设计阶段双击窗体，在窗体的 Form_Load()事件中添加下列代码：

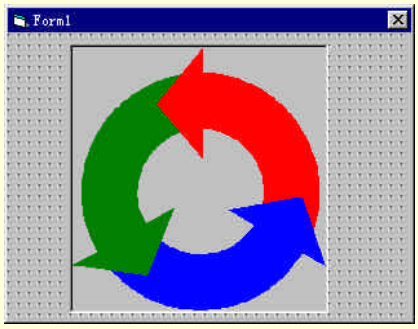


图 2-8　添加控件后的窗体

```
Private Sub Form_Load()
    Picture1.AutoRedraw = True
    '设置控件的 AutoRedraw 属性处于有效的状态
    Picture1.Picture = LoadPicture("")
    '清空控件上的图形
End Sub
```

程序说明：

运行程序时，首先执行 Form_Load()事件中的程序初始化代码，通过语句 Picture1.AutoRedraw = True 来设置控件的 AutoRedraw 属性处于有效的状态，这样输出到控件上的图形同时存储在内存中，然后通过语句 Picture1.Picture = LoadPicture("")来清空控件上的图形。

添加完程序的初始化代码后，在代码窗口中找到窗体的 Form_Click()事件，并且在其中添加下列代码：

```
Private Sub Form_Click()
    Dim X, Y, R As Integer
    '声明变量
    X = Picture1.ScaleWidth / 2
    '设置圆心坐标 X 位置
    Y = Picture1.ScaleHeight / 2
    '设置圆心坐标 Y 位置
    For R = 0 To Picture1.ScaleWidth / 2
    '设置半径
      Picture1.Circle (X, Y), R, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
      '在控件上绘制圆
    Next R
    SavePicture Picture1.Image, "TEST.BMP"
    '将图片保存到文件
End Sub
```

程序说明：

在 Form_Click()事件中的代码 SavePicture Picture1.Image, "TEST.BMP"的作用是把控件中的图形存储到位图文件中。

3．运行程序

选择菜单 File/Save ..，存储文件，按键盘上的功能键 F5 运行程序，在窗体上单击鼠标，在 PictureBox 控件上就会以随机色绘制出一系列的同心圆，如图 2-9 所示。

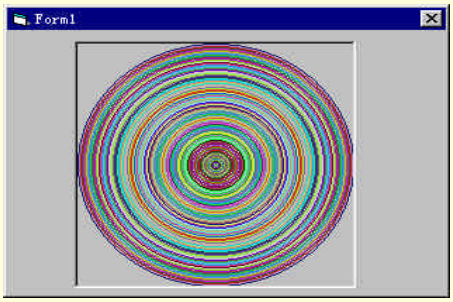


图 2-9 程序运行结果

同时我们也可以在 Viusal Basic 的缺省路径中找到 TEST.BMP 文件，可以用画笔或其他软件打开这个位图文件，如图 2-10 所示。

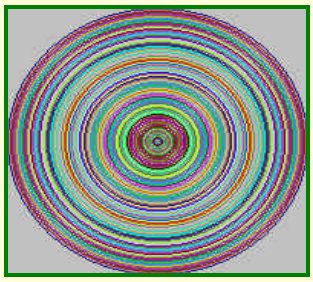


图 2-10 存储的位图文件

2.1.6 Paint 事件

在一个对象被移动或放大之后，或在一个覆盖该对象的窗体被移开之后，该对象部分或全部暴露时，此事件发生。

它的语法结构如下：

Private Sub Picture1_Paint()

如果在代码中需要图形方法的输出，则 Paint 事件过程就很有用，通过使用 Paint 事件，可以确保输出在必要的时候时能被重新绘制。

下面就是一个应用 Paint 事件的示例，示例应用程序的作用是在 PictureBox 控件上绘制一个填充圆，当窗体的大小和位置改变时，要在窗体上重新输出一个填充圆，实现此功能的具体步骤如下：

1. 开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加一个 PictureBox 控件，窗体及控件的属性设置如表 2-4 所示。

表 2-4 控件与窗体的属性设置

窗体或控件	属 性	设 置
窗体	(Name)	Form1
	BorderStyle	2 - Sizable
	Caption	Form1
	Height	3600
	Left	0
	Moveable	True
	StartUpPosition	3 - Windows Default
	Top	0
	Width	4800
PictureBox 控件	(Name)	Picture1
	Align	2 - Align Bottom
	AutoRedraw	False
	Height	2295
	Left	0
	Picture	(None)
	Top	900
	Width	4680

这样设置的窗体具有以下的特性：

- 窗体在程序的运行过程中可以随意的改变大小；
- 运行程序，窗体的位置可以随意的改变；
- 窗体在程序的运行过程中可以在屏幕上随意移动。

如表 2-3 设置的 PictureBox 控件具有如下的特性：

- 控件的名称为 Picture1；
- 控件显示在窗体的底部，其宽度等于窗体的 ScaleWidth 属性设置值；
- 控件的自动重绘处于无效的状态；
- 控件不加载图像文件。

添加控件后的窗体如图 2-11 所示。

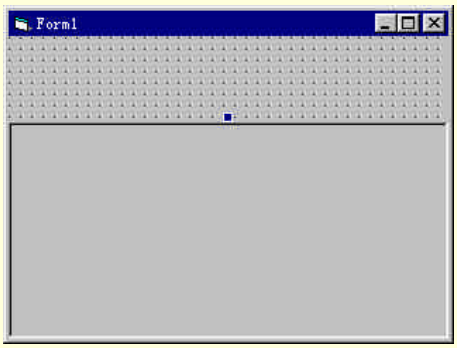


图 2-11 添加控件后的窗体

2．程序的初始化

在程序的设计阶段双击窗体，在弹出的代码窗口中找到 Form_Load()事件，并且在其中添加程序的初始化代码如下：

```
Private Sub Form_Load()
    Picture1.FillColor = &HFFFF00
    '设置填充色为兰色
    Picture1.FillStyle = 7
    '设置填充模式
End Sub
```

程序说明：

在程序运行的初始阶段，首先执行窗体 Form_Load()事件中的代码，通过 Form_Load()事件中的代码设置控件 Picture1 的填充色为兰色，然后通过语句 Picture1.FillStyle = 7 来设置控件的填充模式为交叉线。

3．添加绘图代码

返回到程序的设计画面，双击 PictureBox 控件，在弹出的代码窗口中找到控件的 Picture1_Paint()事件，在该事件中添加绘图的代码：

```
Private Sub Picture1_Paint()
    Dim X, Y
    '声明变量
    X = Picture1.ScaleLeft + Picture1.ScaleWidth / 2
    ' 设置圆心 X 坐标
    Y = Picture1.ScaleTop + Picture1.ScaleHeight / 2
    ' 设置圆心 Y 坐标
```

```
    Picture1.Circle (X, Y), 1000
    '绘制一个圆
End Sub
```

程序说明：

首先定义了两个坐标变量 X，Y，**用于存储圆心的坐标，然后设置圆心坐标位于在控件的中央，最后通过语句** Picture1.Circle (X, Y), 1000 **在控件上绘制一个圆。**

添加控件重绘的代码，双击 PictureBox **控件，在代码窗口找到控件的** Picture1_Resize()**事件，并且在其中添加更新画面的代码：**

```
Private Sub Picture1_Resize()
    Refresh
    '更新
End Sub
```

4. 运行程序

存储文件，按键盘上的功能键 F5**，运行程序，程序运行的初始画面如图** 2-12 **所示。**

在程序的运行过程中，可以随时的改变窗体的大小和位置，同时 PictureBox **控件会随着窗体的改变而改变，图形(填充圆)也会随之作相应的变化，如图** 2-13 **所示。**

注意：

- **如果控件的** AutoRedraw **属性被设置为** True**，重新绘图就会自动进行，于是就不需要** Paint **事件。**

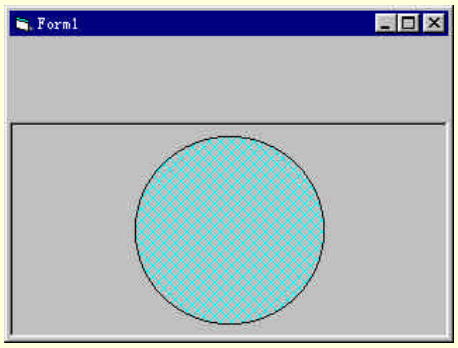


图 2-12 **程序运行的初始画面**

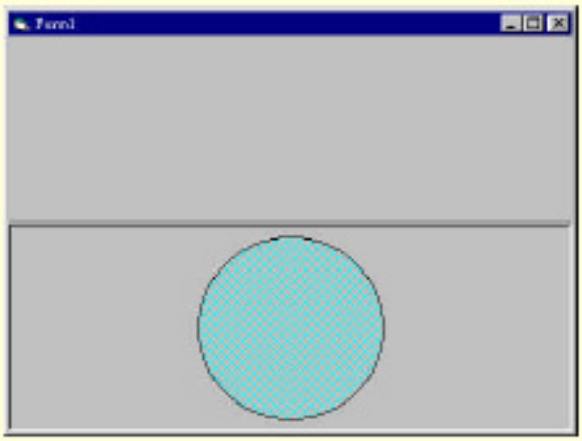


图 2-13 **重绘后的窗体**

2.1.7 PaintPicture **方法**

PaintPicture **方法用以在** PictureBox **控件上绘制图形文件（***.bmp, *.wmf, *.emf, *.ico **或*** .dib**）的内容。**

它的语法结构如下所示：

object.PaintPicture picture, x1, y1, width1, height1, x2, y2, width2, height2, opcode

其中 PaintPicture 方法的语法中各个部分的说明如表 2-5 所示。

表 2-5 语法说明

参 数	说 明
Object	一个对象表达式，如果省略 object，带有焦点的 Form 对象缺省为 object
Picture	要绘制到 object 上的图像来源
x1, y1	指定在控件上绘制 picture 的目标坐标（x-轴和 y-轴）
Width1	指示 picture 的目标宽度，如果该参数省略，则使用源宽度
Height1	指示 picture 的目标高度如果该参数省略，则使用源高度
x2, y2	指示 picture 内剪贴区的坐标（x-轴和 y-轴），如果该参数省略，则缺省为 0
Width2	指示 picture 内剪贴区的源宽度，如果该参数省略，则使用整个源宽度
Height2	指示 picture 内剪贴区的源高度，如果该参数省略，则使用整个源高度
Opcode	是长型数值或仅由位图使用的代码

在表 2-4 中所示的参数中，并不是所有的参数都是必选的参数，其中 Width1 参数、Height1 参数、x2，y2 参数、Width2 Height2 参数和 Opcode 参数都是可选的参数。

技巧：

通过使用负的目标高度值（height1）或目标宽度值（width1），可以以水平或垂直翻转位图。

下面通过一个示例程序来说明 PaintPicture 方法在 PictureBox 控件上的应用，本示例程序的作用是显示一幅逐渐缩小的位图文件，实现此功能的具体步骤如下：

1．开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，在窗体上放置两个 PictureBox 控件，添加控件后的窗体如图 2-14 所示。

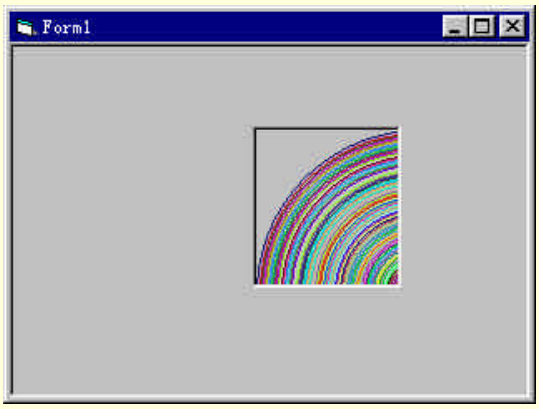


图 2-14 添加控件后的窗体

其中控件的属性设置如表 2-6 所示。

表 2-6 控件的属性设置

属 性	设 置
(Name)	Picture1
Align	3 - Align Left
Height	3195
Left	0
Picture	(None)
Top	0
Visible	True
Width	4680
(Name)	Picture2
Align	0 - None
Height	1455
Left	2160
Picture	(Bitmap)
Top	720
Visible	False
Width	1335

其中控件 Picture2 的 Visible 属性设置为 False，也即控件 Picture2 在程序运行期间处于不可见的状态，它的作用只是为控件 picture1 提供一个图像文件的容器。

2. 添加代码

在程序的设计阶段双击控件 picture1，在弹出的代码窗口中找到控件 picture1 的 Picture1_Click()事件，并且在其中添加下列代码：

```
Private Sub Picture1_Click()
Dim i As Double
'定义一个变量
For i = 1 To 10 Step 0.01
Picture1.PaintPicture Picture2.Picture, 0, 0, Picture1.ScaleWidth, Picture1.ScaleHeight, i, i, Picture2.ScaleWidth * i, Picture2.ScaleHeight * i
'复制图像
Cls
'清除图像
Next
End Sub
```

程序说明：

在程序的运行过程中，在控件 picture1 上单击鼠标时，就会激活控件 picture1 的 Picture1_Click()事件，程序首先定义一个变量 I 用来控制循环，然后通过语句 Picture1.PaintPicture Picture2.Picture，0，0，Picture1.ScaleWidth，Picture1.Scale_Height，I，I，Picture2.ScaleWidth * I，Picture2.ScaleHeight * i 来从控件 picture2 复制图像文件，动画效果的实现是通过不断的清除图像与重现来实现的。

3. 运行程序

存储文件，按键盘上的功能键 F5 运行程序，在控件 Picture1 上单击鼠标时，控件上就会有一幅图像不断的向左上角移动，并且会留下一个长长的“尾巴”，结果如图 2-15 所示。

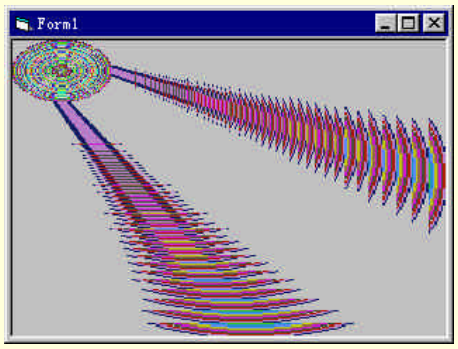


图 2-15 程序的运行结果

注意：

✍ 在 PaintPicture 方法中可以省略任何多个可选的参数，但是需要注意的是，如果想指定某个可选参数，则必须先指定语法中出现在该参数前面的全部参数。

2.2 使用命令按钮控件

命令按钮（CommandButton）控件可以开始、中断或者结束一个进程，当控件处于有效的状态时，单击 CommandButton 控件就会产生一种被按下的感觉，所以通常的情况下 CommandButton 控件都是作为一个按钮的形式出现在窗体上，但是有些时候还可以把它作为其他控件的容器。

下面首先介绍一下 CommandButton 控件的常用的属性、事件和方法。

2.2.1 DisabledPicture 属性

DisabledPicture 属性用于设置在控件无效时对一个图片的引用，该图片当控件处于无效的状态时显示在控件中。

它的语法如下所示：

object.DisabledPicture [= picture]

DisabledPicture 属性可以在设计阶段从属性窗口加载图像文件，同时也可以在运行时使用 LoadPicture 函数加载图像。

例如在程序的设计阶段双击窗体，在弹出的代码窗口中找到窗体的 Form_Load()事件，在其中添加下列代码：

```
Private Sub Form_Load()
Command1.Enabled = False
'设置控件处于无效的状态
Command1.DisabledPicture = LoadPicture("d:\test.bmp")
'设置控件的 DisabledPicture 属性
End Sub
```

添加代码后，返回到窗体的设计界面，设置控件 Command1 的 Style 属性值为 1，然后存储文件，运行程序，程序运行的初始画面如图 2-16 所示。

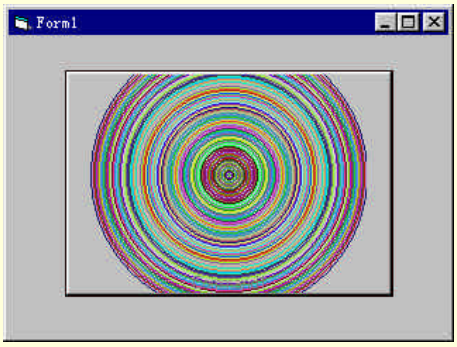


图 2-16 程序运行的结果

程序运行后，首先设置控件处于无效的状态，然后设置控件的 DisabledPicture 属性值，在无效的控件中显示一个图像文件。

注意：

当 CommandButton 控件的 Style 属性设置为 0 时，DisabledPicture 属性将被忽略，只当 CommandButton 控件的 Style 属性设置为 1 时，DisabledPicture 属性才会起作用。

2.2.2 DownPicture 属性

DownPicture 属性用于设置在控件被单击并处于压下状态时对一个对图像文件的引用，该图片在控件处于压下的状态时显示在控件中。

它的语法结构如下所示：

object.DownPicture [= picture]

其中 DownPicture 属性所引用的图像文件可以在设计阶段从属性窗口加载图像文件，同时也可以在运行时使用 LoadPicture 函数加载图像。

例如在程序的设计阶段双击窗体，在弹出的代码窗口中找到窗体的 Form_Load()事件，在其中添加下列代码：

```
Private Sub Form_Load()
Command1.Enabled = True
```

```
'设置控件处于有效的状态
Command1.Caption = "控件的 DownPicture 属性的设置"
'设置控件的 Caption 属性
Command1.DownPicture = LoadPicture("d:\Arrows.bmp")
'设置控件的 DisabledPicture 属性
End Sub
```

存储文件，运行程序，在 CommandButton 控件上按下鼠标的左键，控件上就会显示一幅图像文件——d:\Arrows.bmp，如图 2-17 所示。

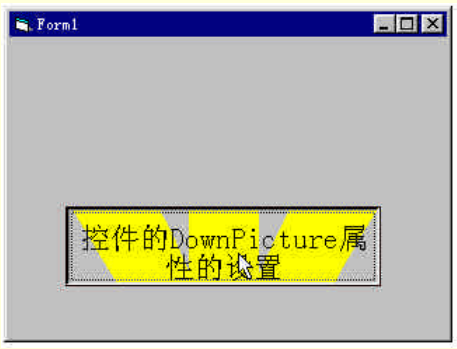


图 2-17 程序运行结果

注意：

> ✍ 当 CommandButton 控件的 Style 属性设置为 0 时，DownPicture 属性将被忽略，只当 CommandButton 控件的 Style 属性设置为 1 时，DownPicture 属性才会起作用。

2.2.3 UserMaskColor 属性、MaskColor 属性

MaskColor 属性用于返回或设置一个在 CommandButton 控件的图像文件中作为透明的颜色。

它的语法结构如下：

```
object.MaskColor [= color]
```

在 Viusal Basic 中使用 Microsoft Windows 操作环境的 RGB 配色方案，如&HFFFFFF 代表白色，&HFF&代表红色，&HFF00&代表绿色，&HFF0000 代表兰色等。

注意：

> ✍ 只有当 CommandButton 控件中的 UseMaskColor 属性设置为 True，同时 CommandButton 控件的 Picture 属性设置为一个位图文件时，MaskColor 属性才能够被使用。

UserMaskColor 属性用于设置控件是否具有设置透明色的能力，也即设置控件中是否可以使用 MaskColor 属性来设置透明色。

它的语法结构如下所示：

```
object.UseMaskColor [= boolean]
```

其中 UserMaskColor 属性的设置值及其说明如下：

- True：可以赋值给 MaskColor 属性的颜色一个颜色代码，在该颜色所在处创建透明区域；
- False：MaskColor 属性将被忽略。

注意：

> ✍ 只有 CommandButton 控件的 Style 属性设置为 1 时，也即控件处于 Graphical 状态时，才可以设置 UserMaskColor 属性。

如图 2-18 所示即为设置了 UserMaskColor 属性和 MaskColor 属性的 CommandButton 控件的显示情况。

图 2-18　设置控件的透明区域

其中 CommandButton 控件的属性设置如表 2-7 所示。

表 2-7　控件的属性设置

属　性	设　置
(Name)	Command1
Appearance	1 - 3D
Cancel	True
Caption	usermaskcolor 和 maskcolor 属性
Default	True
DownPicture	(Bitmap)
Enabled	True
Height	2655
Left	1080
MaskColor	&H00FFFFFF&
Picture	(Bitmap)
Style	1 - Graphical
Top	240
UseMaskColor	True
Visible	True
Width	2775

如表 2-6 的属性设置使得 CommandButton 控件具有如下所示的特性：

- 控件的名称为 Command1；
- 控件的外表为 3D 外观；
- 控件的标题显示为 Usermaskcolor 和 Maskcolor 属性；
- 控件的风格为 Graphical；
- 控件的 Picture 属性设置为一个 BMP 位图；
- 控件的 UseMaskColor 属性处于有效的状态；
- 控件的 MaskColor 属性设置为白色。

其中控件 Picture 属性设置位图如图 2-19 所示。

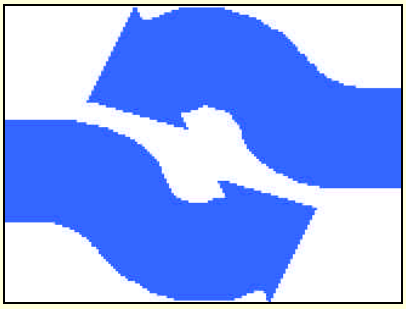


图 2-19　控件调用的位图

2.2.4　命令按钮实例

下面就是一个利用 CommandButton 控件来编制计算器应用程序的示例，在这个计算器的应用程序中可以进

行简单的加减乘除运算，实现此功能的具体步骤如下：

1．开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加 16 个 CommandButton 控件和一个 TextBox 控件，其中 TextBox 控件的作用是显示用户的输入及运算结果的输出，而 CommandButton 的作用是提供数字键和运算符，添加控件后的窗体如图 2-20 所示。

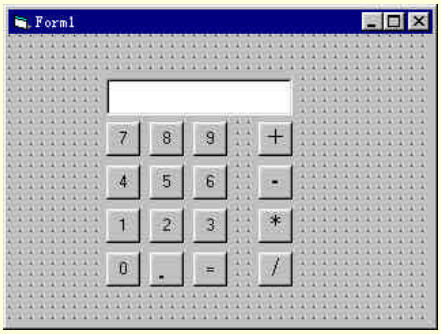


图 2-20　添加控件后的窗体

其中控件的属性设置如表 2-8 所示。

表 2-8　控件的属性设置

控　件	属　性	设　置
CommandButton	(Name)	Command1
	Caption	"/"
	Height	375
	Index	15
	Left	2760
	Top	2400
	Width	375
CommandButton	(Name)	Command1
	Caption	"*"
	Height	375
	Index	14
	Left	2760
	Top	1920
	Width	375
CommandButton	(Name)	Command1
	Caption	"-"
	Height	375
	Index	13
	Left	2760
	Top	1440
	Width	375
CommandButton	(Name)	Command1
	Caption	"+"
	Height	375
	Index	12
	Left	2760
	Top	960
	Width	375
CommandButton	(Name)	Command1
	Caption	"="
	Height	375
	Index	11
	Left	2040
	Top	2400
	Width	375
CommandButton	(Name)	Command1
	Caption	"."
	Height	375
	Index	10
	Left	1560

	Top	2400
	Width	375
CommandButton	(Name)	Command1
	Caption	"0"
	Height	375
	Index	0
	Left	1080
	Top	2400
	Width	375
TextBox	(Name)	Text1
	Height	405
	TabIndex	9
	Top	480
	Width	2055
CommandButton	(Name)	Command1
	Caption	"9"
	Height	375
	Index	9
	Left	2040
	Top	960
	Width	375
CommandButton	(Name)	Command1
	Caption	"8"
	Height	375
	Index	8
	Left	1560
	Top	960
	Width	375
CommandButton	(Name)	Command1
	Caption	"7"
	Height	375
	Index	7
	Left	1080
	Top	960
	Width	375
CommandButton	(Name)	Command1
	Caption	"6"
	Height	375
	Index	6
	Left	2040
	Top	1440
	Width	375
CommandButton	(Name)	Command1
	Caption	"5"
	Height	375
	Index	5
	Left	1560
	Top	1440
	Width	375
CommandButton	(Name)	Command1
	Caption	"4"
	Height	375
	Index	4
	Left	1080
	Top	1440
	Width	375
CommandButton	(Name)	Command1
	Caption	"3"
	Height	375
	Index	3
	Left	2040
	Top	1920
	Width	375
CommandButton	(Name)	Command1
	Caption	"2"
	Height	375
	Index	2

	Left	1560
	Top	1920
	Width	375
CommandButton	(Name)	Command1
	Caption	"1"
	Height	375
	Index	1
	Left	1080
	Top	1920
	Width	375

2．程序的初始化

在程序的设计阶段双击窗体，在弹出的代码窗口中找到程序的声明段，并在其中添加下列代码：

```
Dim flag As Boolean
'定义一个标志变量
Dim num1 As Double
Dim num2 As Double
'定义两个 Double 型变量存储用户的输入
```

程序说明：

开始执行程序时，调用程序声明段中的代码，即首先通过语句 Dimflag As Boolean 定义一个标志变量，用来判断用户在同一个数据的输入过程中是否按过“ . ”按钮，然后通过两条语句：Dim num1 As Double 和 Dim num2 As Double 来定义两个 Double 型变量存储用户的输入。

在代码窗口中找到窗体的 Form_Load()事件，并且在其中添加窗体的初始化代码如下所示：

```
Private Sub Form_Load()
flag = True
num1 = 0
num2 = 0
'程序的初始化
End Sub
```

程序说明：

在窗体的初始化代码中，首先通过一个赋值语句 flag = True 来设置“ . ”按钮的相应状态处于有效的状态，然后设置参与运算的两个 Double 型变量的初始值为 0。

3．响应按钮的按键动作

在代码窗口中选择控件 Command1 的 Command1_Click(Index As Integer)事件，如图 2-21 所示。

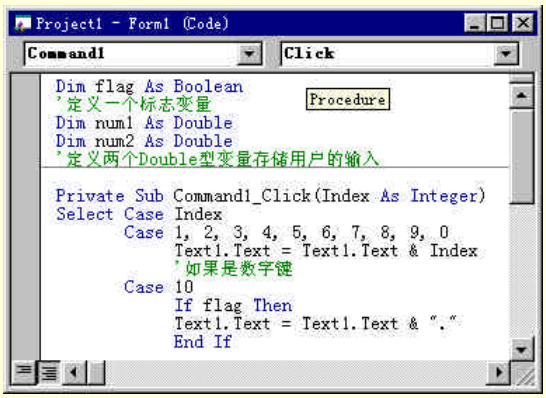


图 2-21 代码窗口

在 Command1_Click(Index As Integer)事件中添加下列代码：

```
Private Sub Command1_Click(Index As Integer)
Select Case Index
        Case 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
            Text1.Text = Text1.Text & Index
```

```
            '如果是数字键
        Case 10
            If flag Then
            Text1.Text = Text1.Text & "."
            End If
            flag = False
            '如果是小数点
        Case 12, 13, 14, 15
            num1 = Val(Text1.Text)
            Text1.Text = ""
            flag = True
            '如果是运算符
         Case 11
            Text1.Text = Val(Text1.Text) * num1
            flag = True
            '如果是符号“ = ”
    End Select
    End Sub
```

程序说明：

在程序的运行过程中单击 Command1 **控件，程序就会通过控件的** Index **属性来判断用户按下的是哪个按钮。**

如果用户按下的是数字键(Case 1, 2, 3, 4, 5, 6, 7, 8, 9, 0)**，那么就要在文本框中显示这个字符**(Text1.Text = Text1.Text & Index)**；**

如果用户按下的是小数点(Case 10)**，如果条件满足的话，就要在文本框中的用户输入数据之后显示一个小数点**(Text1.Text = Text1.Text & ".")**；**

如果用户按下的是运算符(Case 12, 13, 14, 15)**，那么程序首先通过语句** num1 = Val(Text1.Text)**来保存用户刚才的输入数据，然后清空文本框，等待用户的下一次输入；**

否则是符号“ = **”**(Case 11)**，那么就会通过语句**(Text1.Text = Val(Text1.Text) * num1)**来显示运算的输出结果。**

4. 存储文件

完成以上的工作后，选择菜单 File/Save**，就会弹出如图** 2-22 **所示的存储文件的对话框。**

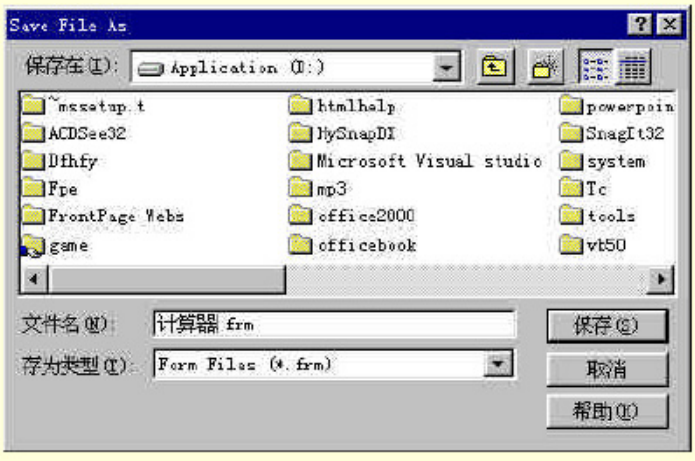


图 2-22 **存储文件对话框**

在“ 文件名 ”输入框中输入文件名，单击“ 保存 ”按钮，就可以存储窗体文件，用户也可以在存储窗体文件的同时存储工程文件。

5. 运行程序

选择菜单 File/Make Project1.EXE**，就会弹出如图** 2-23 **所示的对话框，用户可以在“ 文件名 ”输入框中选择**

应用程序的名称。

用户也可以在生成可执行文件对话框中选择Options...按钮来设置程序的属性，如程序的标题和程序的图标等。

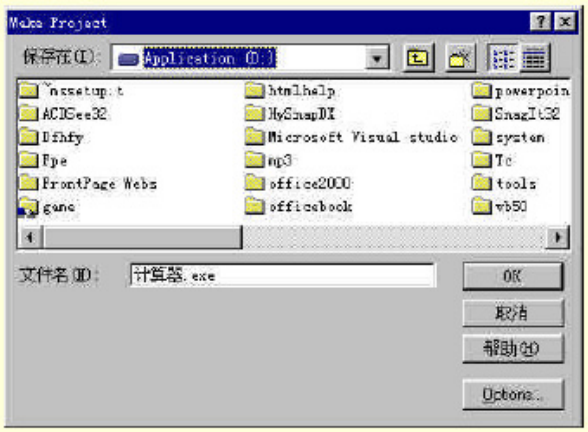


图 2-23 生成可执行文件对话框

提示：

- 1.如果把 CommandButton 控件的 Default 属性设置为 True，那么在按 ENTER 键时也选中命令按钮；
- 2.如果把 CommandButton 控件的 Cancel 属性设置成 True，那么在按 ESC 键时也选中 CommandButton。

按键盘上的功能键 F5，运行程序，结果如图 2-24 所示。

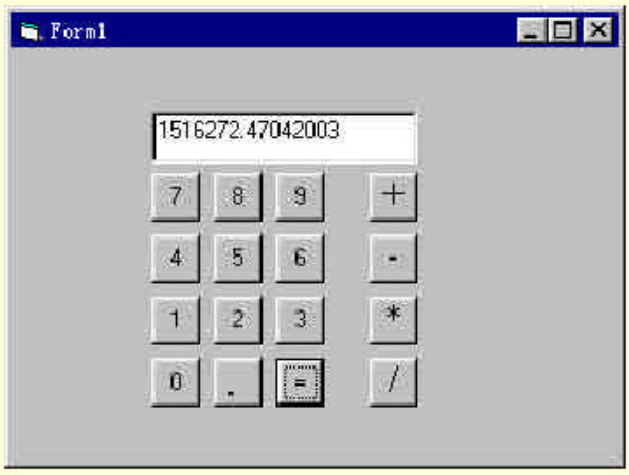


图 2-24 程序运行结果

2.3 使用复选按钮控件

复选按钮（OptionButton）控件用来显示一个可以打开或关闭的选项，而且在同一组中的 OptionButton 控件，用户只能选择其中的一项。

如图 2-25 所示为添加到窗体上的四个 OptionButton 控件。

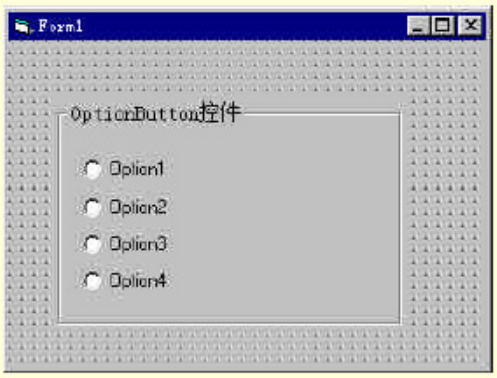


图 2-25 窗体上的 OptionButton 控件

OptionButton 控件和 CheckBox 控件功能相似，但是二者间也存在着重要差别，在选择一个 OptionButton 时，同组中的其他 OptionButton 控件自动无效。相反，可以选择任意数量的 CheckBox 控件。

在如图 2-25 所示的窗体中，由于四个 OptionButton 控件都放置在同一个 Frame 控件上，通过 Frame 控件组成了一个选项组，在程序的运行过程中用户只能选择其中的一项，如图 2-26 所示。

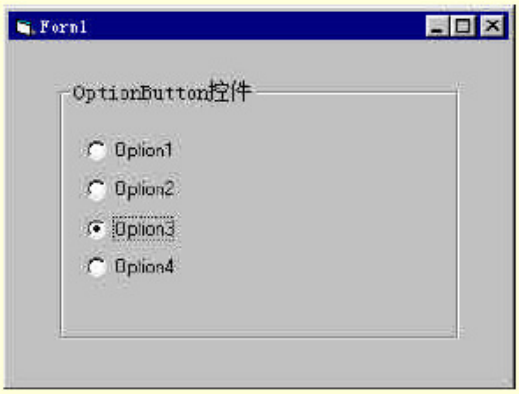


图 2-26 程序运行结果

OptionButton 控件常用的属性如表 2-9 所示。

表 2-9 OptionButton 控件常用的属性

Alignment 属性	Left 属性	Top 属性
ForeColor 属性	MouseIcon 属性	Caption 属性
MousePointer 属性	Container 属性	Name 属性
DisabledPicture 属性	OLEDropMode 属性	DownPicture 属性
Parent 属性	DragIcon 属性	Picture 属性
DragMode 属性	Style 属性	Enabled 属性
TabIndex 属性	Font 属性	TabStop 属性
FontBold 属性	FontItalic 属性	FontStrikethru
FontUnderline 属性	Tag 属性	FontName 属性
ToolTipText 属性	FontSize 属性	UseMaskColor 属性
Height 属性	Width 属性	Value 属性
HelpContextID 属性	Visible 属性	hWnd 属性
WhatsThisHelpID 属性	Index 属性	

在 OptionButton 控件中有一个常用的属性和一个常用的事件值得注意，那就是 Value 属性和 Click 事件，下面将分别的加以叙述。

2.3.1 Value 属性

Value 属性用于设置一个 OptionButton 控件所代表选项的选中状态，它的语法结构如下所示：

object.Value [= value]

其中 Value 属性的设置值及其说明如下：

- True：选中了控件所代表的选项
- False：没有选中控件所代表的选项

如下例所示，在程序的设计阶段双击窗体，在弹出的代码窗口中找到窗体的 Form_Load()事件，并且添加下列代码：

```
Private Sub Form_Load()
Option1.Value = True
'设置控件 Option1 处于选中的状态
Option2.Value = True
'设置控件 Option2 处于选中的状态
End Sub
```

按键盘上的功能键 F5，运行程序，结果如图 2-27 所示。

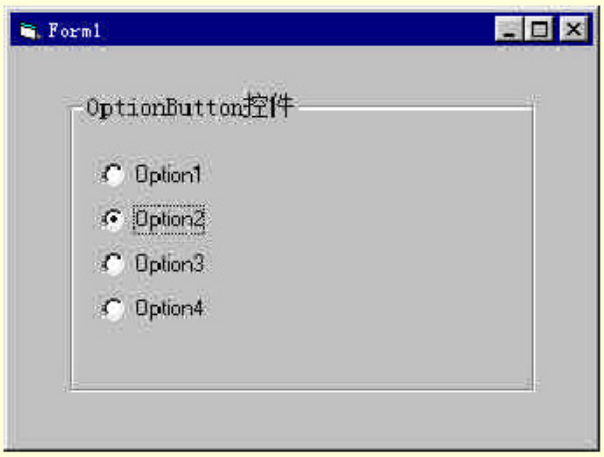


图 2-27 程序运行结果

程序说明：

虽然在程序中有两条语句 Option1.Value = True 和 Option2.Value = True 设置控件 Option1 和控件 Option2 都处于选中的状态，但是由于它们处于同一个组中，所以只能够选中一个，所以只有控件 Option2 处于选中的状态。

2.3.2 Click 事件

OptionButton 控件的 Click 事件是在将一个 OptionButton 控件的 Value 属性设置为 True 的时候发生的。

它的语法结构如下所示：

```
Private Sub object_Click([index As Integer])
```

对于 OptionButton 控件而言，Click 事件仅当单击鼠标左键时发生。

下面以一个示例程序来说明 OptionButton 控件的 Click 事件的用法，示例程序的功能是显示两个数的四则运算结果，它的具体步骤如下：

1. 开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加四个 OptionButton 控件和两个 Label 控件，其中 OptionButton 控件的作用是为用户提供四个选择加减乘除的选项，而 Label 控件用于显示文本和运算结果，添加控件后的窗体如图 2-28 所示。

图 2-28 添加控件后的窗体

其中控件的属性设置如表 2-10 所示。

表 2-10 控件的属性设置

控 件	属 性	设 置
OptionButton	(Name)	Option1
	Caption	"减 法"
	Height	375
	Index	3
	Left	360
	Top	2160
	Width	1575
OptionButton	(Name)	Option1
	Caption	"加 法"
	Height	375
	Index	2
	Left	360
	Top	1680
	Width	1575
OptionButton	(Name)	Option1
	Caption	"除 法"
	Height	375
OptionButton	Index	1
	Left	360
	Top	1200
	Width	1575
OptionButton	(Name)	Option1
	Caption	"乘 法"
	Height	375
	Index	0
	Left	360
	Top	720
	Value	-1 'True
	Width	1575
Label	(Name)	Label2
	Height	375
	Left	2040
	Top	2160
	Width	1695
Label	(Name)	Label1
	Caption	"数字 56 与数字 98 运算"
	Height	375
	Left	2040
	Top	1200
	Width	2295

2．程序初始化

在程序的设计阶段双击窗体，在弹出的代码窗口中找到 Form_Load()事件，并且在其中添加程序的初始化代码如下：

```
Private Sub Form_Load()
Option1(0).Value = True
'选中第一个选项
Label2.Caption = 56 * 89
'程序的初始化
End Sub
```

程序说明：

首先设置控件 Option1(0)**处于选中的状态，即** Option1(0).Value = True**，然后通过语句** Label2.Caption = 56 * 89 **来显示运算的结果。**

3．**响应** Click **事件**

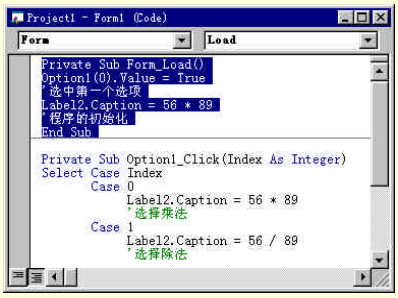


图 2-29　**代码窗口**

在代码窗口中找到控件 Option1 **的** Option1_Click(Index As Integer)**事件，并且在其中添加对** Click **事件的响应代码，代码窗口如图** 2-29 **所示。**

```
Private Sub Option1_Click(Index As Integer)
Select Case Index
    Case 0
        Label2.Caption = 56 * 89
        '选择乘法
    Case 1
        Label2.Caption = 56 / 89
        '选择除法
    Case 2
        Label2.Caption = 56 + 89
        '选择加法
    Case 3
        Label2.Caption = 56 - 89
        '选择减法
End Select
End Sub
```

程序说明：

当程序运行过程中在控件 Option1 **上单击鼠标的左键时，就会激活控件的** Option1_Click(Index As Integer)**事件，然后程序根据** Index **属性来判断用户选中的是那一个运算规则，然后通过语句** Label2.Caption = 56 + 89(**如果选择的是加法)来显示运算结果。**

4．运行程序

选择菜单 File/Save ..**存储文件，按键盘上的功能键** F5 **运行程序，在程序的运行画面上选择“除法”选项，结果如图** 2-30 **所示。**

图 2-30 程序运行结果

2.4 使用滚动条控件

滚动条控件包括 HscrollBar 控件和 VscrollBar 控件，这些控件用于在显的示信息量很大时，为用户提供一个定位的基准。

如图 2-31 所示即为 HscrollBar 控件和 VscrollBar 控件在工具箱上的显示情况。

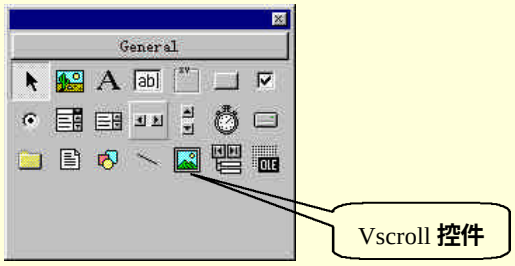


图 2-31 工具箱上的滑动条控件

由于 HscrollBar 控件和 VscrollBar 控件经常在一起使用，而且它们又有很多相同之处，所以在这里就把 HscrollBar 控件和 VscrollBar 控件放在一起进行讲述。

HscrollBar 控件和 VscrollBar 控件常用的属性如表 2-11 所示。

表 2-11 常用的属性

Container 属性	MouseIcon 属性	DragIcon 属性
MousePointer 属性	DragMode 属性	Name 属性
Enabled 属性	Parent 属性	Height 属性
Width 属性	TabIndex 属性	HelpContextID 属性
TabStop 属性	hWnd 属性	Tag 属性
Iandex 属性	Value 属性	LargeChange 属性
SmallChange 属性	Visible 属性	Left 属性
Top 属性	WhatsThisHelpID 属性	Max 属性
Min 属性		

下面对 HscrollBar 控件和 VscrollBar 控件特有的属性作一个简单的介绍，它们分别是 LargeChange 属性、SmallChange 属性、Max 属性和 Min 属性

2.4.1 LargeChange 属性、SmallChange 属性

HscrollBar 控件和 VscrollBar 控件中的 LargeChange 属性用于设置当用户单击滚动条和滚动箭头之间的区域时，HScrollBar 或 VscrollBar 控件的 Value 属性值的改变量。

与 LargeChange 属性相对，SmallChange 用于设置当用户单击滚动箭头时，滚动条控件的 Value 属性值的改变量。

它们的语法结构如下所示：

object.LargeChange [= number]

object.SmallChange [= number]

对于 LargeChange 属性、SmallChange 属性，有效的设置值位于 1 和 32，767 之间，缺省时，它们都被设置为 1。

例如首先向窗体上添加一个 HscrollBar 控件和一个 VscrollBar 控件，然后执行下面的程序：

```
Private Sub Form_Load()
HScroll1.Max = 50
'设置控件 HScroll1 的最大值
VScroll1.Max = 50
'设置控件 VScroll1 的最大值
HScroll1.LargeChange = 10
'设置控件 HScroll1 的最大变化量
VScroll1.LargeChange = 5
'设置控件 VScroll1 的最大变化量
HScroll1.SmallChange = 1
'设置控件 HScroll1 的最小变化量
VScroll1.SmallChange = 1
'设置控件 VScroll1 的最小变化量
End Sub
```

程序的运行结果如图 2-32 所示。

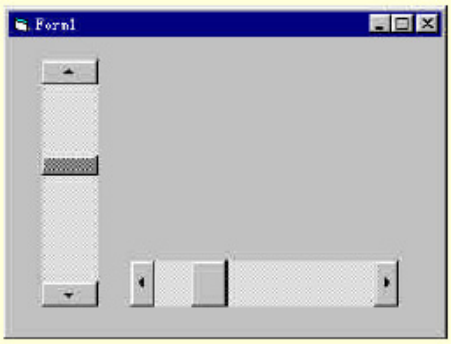


图 2-32 程序的运行结果

单击水平滚动条和滚动箭头之间的区域时，水平滚动条每次改变 10 个单位，而单击垂直滚动条和滚动箭头之间的区域时，垂直滚动条每次改变 5 个单位，当用户单击水平滚动条或者垂直滚动条的箭头时，每次只是移动一个单位。

2.4.2 Max 属性和 Min 属性

HscrollBar 控件和 VscrollBar 控件中的 Max 属性用于设置当滚动框处于底部（对于垂直滚动条）或最右位置（对于水平滚动条）时，一个滚动条位置的 Value 属性最大设置值。

与 Max 属性向对应，Min 属性则用于设置当滚动框处于顶部（对于垂直滚动条）或最右位置（对于水平滚动条）时，一个滚动条位置的 Value 属性最小设置值。

它们的语法结构如下所示：

object.Max [= value]

object.Min [= value]

对于 Max 属性和 Min 属性，它们的有效设置值范围是在 -32,768 和 32,767 之间的一个整数。如果 Max

属性设置值比 Min 属性设置值小，那么最大值将被分别设为水平滚动条或垂直滚动条的最左面或最上面的位置处。

下面就是一个综合的利用 HscrollBar 控件和 VscrollBar 控件的各种属性来设计的一个应用程序，它的作用是在窗体的范围内跟踪鼠标的位置，并且用 HscrollBar 控件和 VscrollBar 控件表示出来，实现它的具体步骤如下：

1．开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，在窗体上放置一个 HscrollBar 控件、VscrollBar 控件和一个 PictureBox 控件，其中 HscrollBar 控件的作用是跟踪鼠标的水平位置，VscrollBar 控件的作用是跟踪鼠标的垂直位置，利用这两个控件的综合作用就可以跟踪窗体中鼠标的位置了，而 PictureBox 控件的作用是为鼠标的移动提供一个容器，添加控件后的窗体如图 2-33 所示。

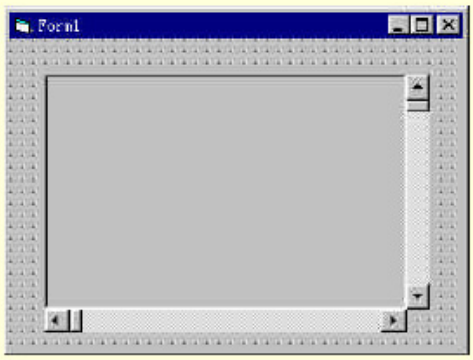


图 2-33　添加控件后的窗体

其中控件的属性设置如表 2-12 示。

表 2-12　控件的属性设置

控　件	属　性	设　置
PictureBox	(Name)	Picture1
	Height	2415
	Left	360
	Top	360
	Width	3735
VscrollBar	(Name)	VScroll1
	Height	2415
	Left	4080
	Top	360
	Width	255
HScrollBar	(Name)	HScroll1
	Height	255
	Left	360
	Top	2760
	Width	3735

2．添加初始化代码

在程序的设计阶段双击窗体，就会弹出一个代码窗口，在窗体 Form_Load()事件中添加下列程序的初始化代码：

```
Private Sub Form_Load()
VScroll1.Min = 0
HScroll1.Min = 0
'设置控件的最小设置值
VScroll1.Max = 100
HScroll1.Max = 100
'设置控件的最大设置值
VScroll1.Value = 50
HScroll1.Value = 50
```

'初始化滚动块的位置

End Sub

程序说明：

程序首先通过四条语句来设置垂直滚动条和水平滚动条的最大设置值和最小设置值。

然后通过 VScroll1.Value = 50、 HScroll1.Value = 50 **来初始化滚动块的初始位置在滚动条的中间。**

3. 响应鼠标的移动

双击窗体中的 PictureBox **控件，在弹出的代码窗口中移动到** Picture1_MouseMove()**事件中，并且添加下列响应鼠标动作的代码：**

```
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Y > Picture1.Top And Y < Picture1.Top + Picture1.Height Then
VScroll1.Value = (Y - Picture1.Top) / (Picture1.Height) * 100
'跟踪垂直位置
End If
If X > Picture1.Left And X < Picture1.Left + Picture1.Width Then
HScroll1.Value = (X - Picture1.Left) / (Picture1.Width) * 100
'跟踪水平位置
End If
End Sub
```

程序说明：

在程序的运行过程中，当鼠标在控件 Picture1 **中时，就会激活** Picture1_MouseMove()**事件。**

然后程序通过两个赋值语句把鼠标的当前位置转换成水平滚动条和垂直滚动条的 Value **属性值。**

这样就实现了滚动条对鼠标位置的跟踪，但是当鼠标移出控件 Picture1**，滚动条的位置就不在变化了。**

程序的运行结果如图 2-34 **所示。**

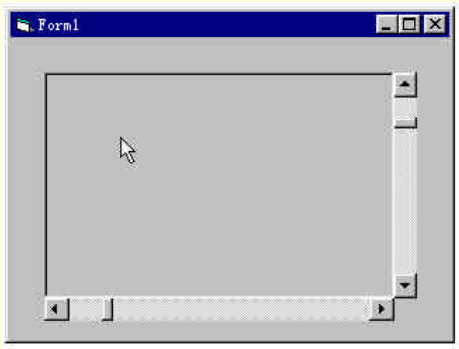
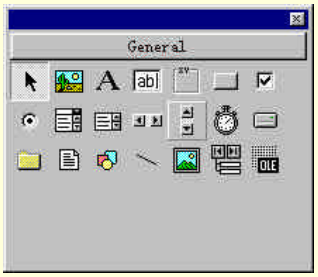


图 2-34 **滚动条对鼠标的跟踪**

2.5 使用列表框控件

列表框（ListBox**）控件的作用时显示一个项目列表，用户可以从中可以选择一个或者多个选项。**

如图 2-35 **所示即为在工具箱上的** ListBox **控件。**



ListBox 控件

图 2-35 工具箱上的 ListBox 控件

ListBox 控件有很多独特的属性，所以要想真正的掌握 ListBox 控件，首先就要熟悉这些属性。

2.5.1 Columns 属性

Columns 属性用于设置 ListBox 控件是水平还是垂直滚动、以及显示选项中列的排列方式。

它的语法结构如下所示：

object.Columns [= number]

其中 Columns 属性设置值及其说明如下：

- ListBox 竖直滚动，并且所有的选项都排列在一列中；
- 1 到 *n* ListBox 水平滚动，同时选项会自动的安排在多个列中。

下面就是一个 Columns 属性应用的例子，首先启动一个新的项目，在空白的窗体上放置两个 ListBox 控件，控件的属性设置如表 2-13 所示。

表 2-13 控件的属性设置

控 件	属 性	设 置
ListBox	(Name)	List2
	Columns	2
	Height	2595
	Left	2400
	Top	240
	Width	1935
ListBox	(Name)	List1
	Columns	0
	Height	2595
	Left	240
	Top	240
	Width	1935

其中控件 List1 中的所有选项都处于同一列中，而控件 List2 的选项会自动的排列在两列中。

添加控件后的窗体如图 2-36 所示。

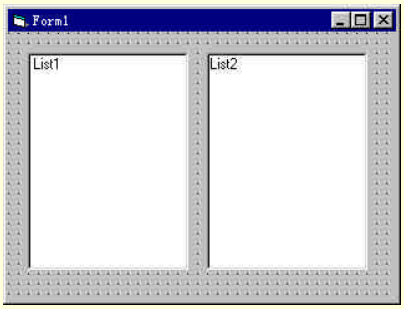


图 2-36 添加控件后的窗体

然后双击窗体，在弹出的代码窗口中找到窗体 Form_Load()事件，并且在其中添加下列程序的初始化代码如下：

```
Private Sub Form_Load()
    Dim I As Integer
```

```
    '声明一个整型变量.
    For I = 0 To Screen.FontCount - 1
        List1.AddItem Screen.Fonts(I)
        ' 填充第一个列表框
        List2.AddItem Screen.Fonts(I)
        '填充第二个列表框
    Next I
End Sub
```

程序说明：

当程序开始运行时，激活了窗体的 Form_Load()**事件，首先定义了一个整型的变量** I**，用于控制程序中的循环，然后通过一个循环语句来读取屏幕的字体样式，最后通过两个赋值的语句来填充两个列表框。**

运行程序，程序的运行结果如图 2-37 **所示。**

图 2-37 **程序的运行结果**

注意：

✍ **在程序的运行过程中** Columns **属性是只读的。**

2.5.2 MultiSelect 属性

MultiSelect **属性用于设置** ListBox **控件中进行复选的方式。**

它的语法结构如下所示：

object.MultiSelect=[Number]

其中 MultiSelect **属性的设置值及其说明如下：**

- 0**：不允许复选**
- 1**：在列表中单击鼠标就可以进行选择的动作**
- 2**：许用** SHIFT **键、方向键、**CTRL **键和鼠标键来进行复选的动作**

下面以一个示例来说明 MultiSelect **属性的应用，示例程序的作用是从一个列表框向另外的列表框中添加选项，它的具体步骤如下所示：**

1. **开始工作**

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加两个 ListBox **控件和一个** CommandButton **控件，添加控件后的窗体如图** 2-38 **所示。**

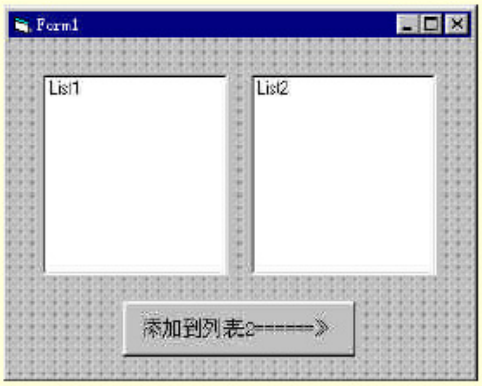


图 2-38 添加控件后的窗体

其中控件的属性设置如表 2-14 所示。

表 2-14 控件的属性设置

控 件	属 性	设 置
CommandButton	(Name)	Command1
	Caption	"添加到列表 2======》"
	Height	555
	Left	1200
	Top	2640
	Width	2415
ListBox	(Name)	List2
	Height	2010
	Left	2520
	TabIndex	1
	Top	360
	Width	1935
ListBox	(Name)	List1
	Height	2010
	Left	360
	MultiSelect	2 'Extended
	Top	360
	Width	1935

2．程序的初始化

在程序的设计阶段，在窗体上双击鼠标，在弹出的代码窗口中找到窗体的 Form_Load()事件。

其中添加程序的初始化代码如下：

```
Private Sub Form_Load()
List1.AddItem "北京"
List1.AddItem "上海"
List1.AddItem "天津"
List1.AddItem "重庆"
List1.AddItem "哈尔滨"
List1.AddItem "深圳"
List1.AddItem "广东"
List1.AddItem "珠海"
List1.AddItem "汕头"
List1.AddItem "海南"
'初始化控件 List1
List2.Clear
'初始化控件 List2
End Sub
```

在程序的初始化代码中，首先向控件 List1 中添加了 10 个选项，然后通过一条语句 List2.Clear 把控件 List2

清空。

添加代码后的代码窗口如图 2-39 所示。

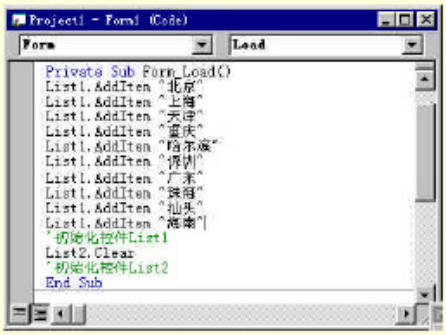


图 2-39 代码窗口

3. 响应按钮的按下动作

双击控件 Command1，在代码窗口中的光标就会自动的跳转到控件 Command1 的 Command1_Click()事件处，在该事件中添加下列代码：

```
Private Sub Command1_Click()
    For I = 0 To List1.ListCount - 1
        If List1.Selected(I) Then
            '判断选项的选中状态
            List2.AddItem List1.List(I)
            '向控件 List2 添加选项
        End If
    Next I
End Sub
```

4. 运行程序

存储文件，按键盘上的功能键 F5 运行程序，程序运行的初始画面如图 2-40 所示。

图 2-40 程序运行的初始画面

在程序运行的初始画面中，由于执行了 List2.Clear 语句，所以控件 List2 被清空了，在列表框 List1 中用 SHIFT 键、CTRL 键和鼠标键选择选项，单击“添加到列表 2=====》”，结果如图 2-41 所示。

图 2-41 程序的运行结果

2.6 使用组合框控件

组合框（ComboBox）控件既可以在控件的文本框部分输入信息，也可以在控件的列表框部分选择一个给定的选项。如图 2-42 所示即为工具箱上的 ComboBox 控件。

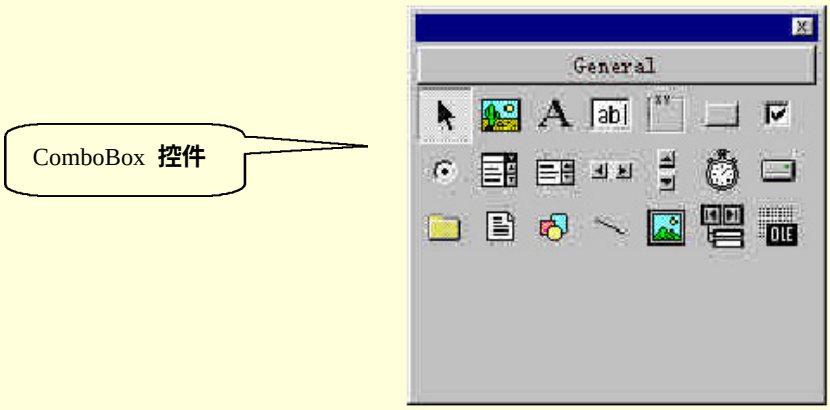


图 2-42 工具箱上的 ComboBox 控件

ComboBox 控件的大多数属性、事件和方法都与 ListBox 控件相差不多，所以在这里就不多加叙述了，只是介绍一下 ComboBox 控件特有的属性。

2.6.1 Style 属性

ComboBox 控件的 Style 属性用于设置 ComboBox 控件的显示类型。

其中 Style 属性的设置值及其说明如下所示：

- 0：下拉式组合框，ComboBox 控件包括一个下拉式列表和一个文本框
- 1：简单组合框，ComboBox 控件包括一个文本框和一个不能下拉的列表
- 3：下拉式列表，用户只能够从下拉式列表中选择选项，不能够接受输入

例如在窗体上添加三个 ComboBox 控件，控件的 Style 属性分别设置为 0，1，2，并且在窗体的 Form_Load() 事件中添加下列代码：

```
Private Sub Form_Load()
Combo1.AddItem "北京"
Combo1.AddItem "上海"
'设置控件 Combo1 的选项
Combo2.AddItem "北京"
```

```
Combo2.AddItem "上海"
'设置控件 Combo2 的选项
Combo3.AddItem "北京"
Combo3.AddItem "上海"
'设置控件 Combo3 的选项
End Sub
```

存储文件，运行程序，结果如图 2-43 所示，虽然对控件 Combo1、Combo2 和 Combo3 都执行了相同的代码，但是结果由于 Style 属性设置的不同，所以三个控件的显示形式是不同的。

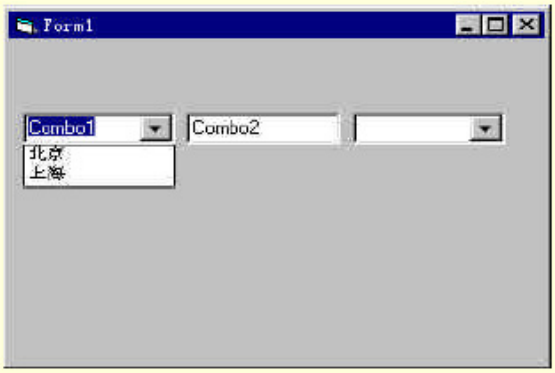


图 2-43 Style 属性的设置

2.6.2 Click 事件

Click 事件在 Vsiaul Basic 的应用程序中用的十分的普遍，一般的情况下 Click 事件都是依靠单击鼠标来激活的，但是在有些时候也可以通过控件状态的改变来激活，如下面的程序所示。

1．开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加一个 ComboBox 控件，控件的属性设置如表 2-15 所示。

表 2-15 控件的属性设置

属 性	设 置
(Name)	Combo1
Height	315
Left	840
Style	0 - Dropdown Combo
Text	Combo1
Top	480
Width	2895

2．添加代码

首先添加程序的初始化代码，在程序的设计阶段双击窗体，在弹出的代码窗口中找到窗体的 Form_Load() 事件。

其中添加下列代码：

```
Private Sub Form_Load()
Combo1.AddItem "北京"
Combo1.AddItem "上海"
Combo1.AddItem "哈尔滨"
Combo1.AddItem "重庆"
'控件的初始化
End Sub
```

程序说明：

在程序的初始化代码中，通过四次反复的调用控件 Combo1 的 AddItem 方法向控件的列表框中添加了四个

选项——北京、上海、哈尔滨和重庆。

接着添加响应在控件上单击鼠标引起控件状态变化的代码，在窗体上双击控件 Combo1，在弹出的代码窗口中找到控件的 Combo1_Click()事件。

其中添加下列代码：

```
Private Sub Combo1_Click()
Select Case Combo1.Text
        Case "北京"
        MsgBox "北京—中国的首都"
        Case "上海"
        MsgBox "上海—中国的经济中心"
        Case "哈尔滨"
        MsgBox "哈尔滨——冰城"
        Case "重庆"
        MsgBox "重庆——山城"
End Select
End Sub
```

程序说明：

在程序的运行过程中，打开控件 Combo1，选中控件的一个选项，比如选中了“哈尔滨”，那么就会激活语句 MsgBox "哈尔滨——冰城"，显示一个输出对话框。

3. 运行程序

存储文件，按键盘上的功能键 F5 运行程序，程序运行的结果如图 2-44 所示，在列表框中用户不但可以选择一个选项，而且可以自己输入一个新的选项。

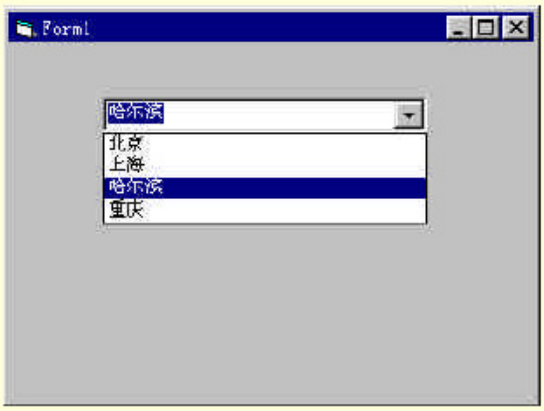


图 2-44 程序的运行结果

2.7 使用文本框控件

文本框（TextBox ）控件有些时候用于接受用户的输入，有些时候用于显示程序的信息，另外有一些时候还用来显示程序的输出。

如图 2-45 所示即为 TextBox 控件在工具箱上的显示情况。

图 2-45 工具箱上的 TextBox 控件

如表 2-16 所示为 TextBox 控件常用的属性。

表 2-16 TextBox 控件常用的属性

Alignment 属性	MouseIcon 属性	Appearance 属性	MousePointer 属性
BackColor 属性	ForeColor 属性	MultiLine 属性	BorderStyle 属性
Name 属性	Container 属性	OLEDragMode 属性	DataChanged 属性
OLEDropMode 属性	DataField 属性	OLEDropMode 属性	DataSource 属性
Parent 属性	DragIcon 属性	PasswordChar 属性	DragMode 属性
ScrollBars 属性	Enabled 属性	SelLength 属性	SelStart 属性
SelText 属性	Font 属性	FontBold 属性	FontItalic 属性
FontStrikethru 属性	FontUnderline 属性	TabIndex 属性	FontName 属性
TabStop 属性	FontSize 属性	Tag 属性	Height 属性
Width 属性	Text 属性	HelpContextID 属性	ToolTipText 属性
HideSelection 属性	Visible 属性	hWnd 属性	WhatsThisHelpID 属性
Index 属性	Left 属性	Top 属性	LinkItem 属性
LinkMode 属性	Locked 属性	LinkTimeout 属性	LinkTopic 属性
MaxLength 属性			

TextBox 控件的用途很广泛，但是由于它的语法结构非常的简单，所以在这里只是简单的介绍一下 TextBox 控件特有的属性和事件。

2.7.1 MultiLine 属性

MultiLine 属性用于设置 TextBox 控件能否输入和显示多行文本，它的语法结构如下所示：

object. MultiLine [= boolean]

其中 MultiLine 属性的设置及其说明如下：

- True ：允许输入和显示多行文本；
- False：所有的文本都限制在一行之内。

将 TextBox 控件的 MultiLine 属性设置为 True，就可以在控件中输入和显示多行文本，如果同时设置了控件的 ScrollBars 属性，那么就可以在 TextBox 控件上定制垂直和水平的滚动条。

下面就是一个利用 MultiLine 属性的设置来实现在 TextBox 控件中输入竖行文本的示例，实现此功能的具体步骤如下：

1. 开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，向窗体上添加一个 TextBox 控件，其中控件的属性设置如表 2-17 所示。

表 2-17 控件的属性设置

属 性	设 置
(Name)	Text1
Font	@宋体
Height	2175
Left	600
Locked	False
MultiLine	True
ScrollBars	3 - Both
Text	(Text)
Top	480
Width	3135

这样设置的控件具有如下所示的特性：

- 在程序的运行过程中可以编辑控件中的文本；
- 在控件中可以输入多行文本；
- 当文本长度或宽度大于控件的相应宽度和高度后，就会显示水平或垂直的滚动条；
- 字体设置为@宋体。

2. 字体的设置

在实现竖行文本的显示过程中，字体的设置是一个相当重要的过程，在控件的属性设置对话框中单击 Font 属性设置框右侧的...按钮，就会弹出如图 2-46 所示的字体设置对话框。

图 2-46 字体设置对话框

在字体设置的对话框中，如下设置字体：

- 字体选项框中选择@宋体；
- 在“字体样式”输入框中选择 Bold；
- 在“大小”输入框中选择“小三”；
- 在“语系”选择框中选择 CHINESE_GB2312。

3. 运行程序

存储文件，运行程序，程序的运行结果如图 2-47 所示。

在窗体的文本框中按如下的方法输入文字：输入“出报”，按 Enter 键，输入“付回”，按 Enter 键，输入“份份”，按 Enter 键，输入“——”，按 Enter 键。

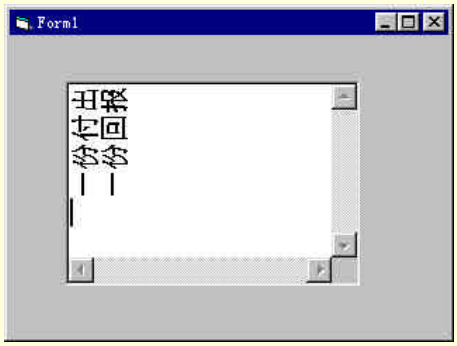


图 2-47 程序的运行结果

2.7.2 SelStart 属性

SelStart 属性用于设置所选择的文本的起始点。

它的语法结构如下所示：

object.SelStart [= index]

下面就是一个利用SelStart 属性来实现字符串查找功能的示例程序，实现此功能的具体步骤如下：

1. 开始工作

首先启动一个新的项目，向空白的窗体上添加一个 TextBox 控件和一个 CommandButton 控件，其中 TextBox 控件的作用是为字符串的查找提供一个容器，而 CommandButton 控件的作用是为字符串的查找提供一个控制。

两个控件的属性设置如表 2-18 所示。

表 2-18 控件的属性设置

控 件	属 性	设 置
CommandButton	(Name)	Command1
	Caption	"查 找"
	Height	495
	Left	3000
	Top	2400
	Width	1335
TextBox	(Name)	Text1
	Height	2415
	Left	600
	MultiLine	-1 'True
	Text	"find.frx":0000
	Top	480
	Width	2055

添加控件后的窗体如图 2-48 所示。

图 2-48 添加控件后的窗体

2. 添加代码

在程序的设计阶段，双击窗体中的 CommandButton 控件，在弹出的对话框中找到控件的 Command1_Click()

事件，并且在其中添加下列代码：

```
Private Sub Command1_Click()
Dim find As String
Dim address As String
    ' 声明两个字符串变量
    find = InputBox("请输入一个字符串：")
    '从用户的输入中得到查找的字符串
    address = InStr(Text1.Text, find)
    ' 在文本中查找字符串
    If address Then
        '找到了字符串
        Text1.SelStart = address - 1
        ' 设置选定的起始位置
        Text1.SelLength = Len(find)
        '设置字符串长度
    Else
        MsgBox "没有找到字符串"
        '没有找到
    End If
End Sub
```

程序说明：

在程序运行的过程中，单击 CommandButton 控件，就会激活控件的 Command1_Click()事件，程序首先定义了两个字符串变量，用于存储用户输入要查找的字符串；

然后程序通过语句 address = InStr(Text1.Text, find)来实现字符串的查找操作，如果在给定的字符串中能够找到 Find 字符串，那么字符串 address 就非空。

3．运行程序

存储文件，按键盘上的功能键 F5，运行程序，在运行的窗体上单击“查找”按钮，就会弹出如图 2-49 所示的对话框。

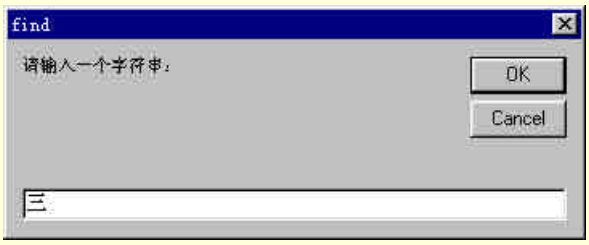


图 2-49　弹出对话框

在弹出的对话框中可以输入要查找的字符串，如在本示例中输入“三”，单击 OK 按钮，由于在文本框中有“三”这个字符串，所以程序运行的结果如图 2-50 所示。

图 2-50　程序的运行结果

2.8 小 结

在本章中，主要介绍 Visual Basic 中常见的控件和它们的使用方法。

通过本章的学习，用户应该可以明确的掌握各个控件的功能和作用，特别是各个控件的属性值，读者要熟练的应用。当然，读者在学习使用属性对控件进行设置外，还学习了如何利用程序中的代码动态控制控件，这在某些应用程序中是非常有用的。

最后笔者要说的是，利用 Visual Basic 提供的控件进行程序界面设计，不仅仅是简单的编程问题，还是个人欣赏观念和修养的问题，这个能力可以在以后的编程学习中体会到。在后面的各章中，您就可以慢慢理解和掌握了。

第三章　常用对话框的设计和使用

用任何的编程语言编制应用程序时，都是一个程序员与编程环境的交流过程，而任何一个应用程序的运行的目的都是为了用户与程序之间的成功交流，从而达到更好的为用户服务的目的。在本章主要讲述的是在编制应用程序的过程中，程序员如何与 Visual Basic 6.0 编程环境之间交流，如何能够更加高效的工作。

提示：

- 在 Visual Basic 6.0 中的交流手段有两种：一是在程序运行过程中的对话框，另外就是利用程序调试工具。

本章将通过几个典型的示例介绍对话框设计，而程序调试的内容在本书的其他内容中穿插介绍。

3.1　输入对话框

在 Visual Basic 6.0 中显示一个输入对话框所调用的函数是 InputBox()，它的作用是显示一个对话框，等待用户输入正文或按下按钮，并返回包含文本框内容的一个字符串。

它的语法结构如下所示：

```
InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])
```

如下面的几条语句就可以实现显示一个输入对话框的功能。

```
Private Sub Form_Load()
InputBox ""
End Sub
```

运行程序，结果如图 3-1 所示。

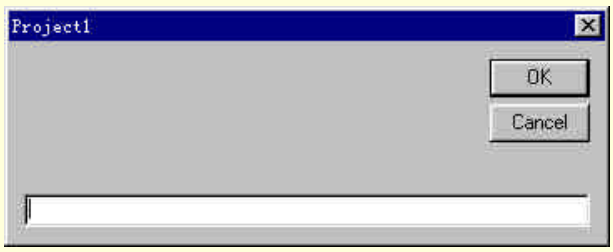


图 3-1　不带任何参数的输入对话框

其中 InputBox 函数的参数说明如表 3-1 所示。

表 3-1　参数说明

参　数	说　明
prompt	作为对话框消息出现的字符串表达式
title	显示对话框标题栏中的字符串表达式
default	显示文本框中的字符串表达式，在没有其他输入时作为缺省值
xpos	指定对话框的左边与屏幕左边的水平距离
ypos	指定对话框的上边与屏幕上边的距离
Context	由帮助文件的作者指定给某个帮助主题的帮助上下文编号

下面以一个示例来说明 InputBox 函数的用法，它的具体设计步骤如下：

1．开始工作

首先启动一个新的项目，在空白的窗体上放置一个 CommandButton 控件、三个 TextBox 控件和两个 Label 控件，窗体及控件的属性设置如表 3-2 所示。

表 3-2 窗体和控件的属性设置

名 称	属 性	设 置
窗体	(Name)	Form1
	Caption	输入对话框示例程序
TextBox 控件	(Name)	Text1
	Text	3.141592653589
TextBox 控件	(Name)	Text2
	Text	待输入数据
TextBox 控件	(Name)	Text3
	Text	计算结果
CommandButton 控件	(Name)	Command1
	Caption	输入数据
Label 控件	(Name)	Label2
	Caption	*
Label 控件	(Name)	Label1
	Caption	=

添加控件后的窗体如图 3-2 所示。

图 3-2 添加控件后的窗体

2．添加代码

首先进行程序的初始化工作，在窗体的设计阶段双击窗体，在窗体的声明模块中定义一个全局变量。

```
Dim number1 As Double
'定义一个全局变量
```

添加窗体的初始化代码，在窗体的代码窗口中找到窗体的 Form_Load()事件，并且在其中添加初始化变量的代码如下：

```
Private Sub Form_Load()
Number = 0
'变量初始化
End Sub
```

最后添加相应 CommandButton 控件动作的代码，在控件“ 输入数据 ”按钮的 Command1_Click()事件中添加下列代码：

```
Private Sub Command1_Click()
Number = InputBox("请输入一个有效的数字变量值：", "输入数据", 0)
'显示一个输入对话框
Text2.Text = Number
Text3.Text = Number * Text1.Text
'显示运算结果
End Sub
```

3. 运行程序

存储文件，按键盘上的功能键 F5 运行程序，程序运行的初始画面如图 3-3 所示。

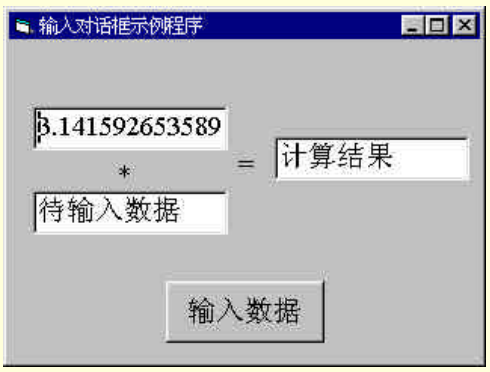


图 3-3 程序运行的初始画面

单击“输入数据”按钮，就会弹出一个等待用户输入一个有效的数字的对话框，如图 3-4 所示。

图 3-4 输入对话框

在“输入数据”对话框中输入一个数字后，单击 OK 按钮，就会返回程序运行的窗体，如图 3-5 所示。

图 3-5 程序的运行结果

3.2 输出对话框

在 Visual Basic6.0 中调用 MsgBox()函数显示一个输出对话框，等待用户单击按钮，并返回一个整型数，告诉用户单击哪一个按钮。

它的语法如下所示：

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

其中的参数及其说明如下所示。

Prompt 作为显示在对话框中的消息

Buttons 指定显示按钮的数目及形式，使用的图标样式，缺省按钮是什么以及消息框的强制回应等

Title 在对话框标题栏中显示的字符串表达式

Helpfile 识别用来向对话框提供上下文相关帮助的帮助文件
Context 由帮助文件的作者指定给适当的帮助主题的帮助上下文编号
其中 Button 参数的设置略微有一些复杂，它包括四个方面的设置，如下所示：

- 第一组值 描述对话框中显示的按钮的类型与数目(0-5)；
- 第二组值 描述图标的样式(16，32，48，64)；
- 第三组值 说明哪一个按钮是缺省值(0，256，512)；
- 第四组值 决定消息框的强制返回性(0, 4096)。

它的具体设置值及其说明如表 3-3 所示。

表 3-3 Button 参数的设置

设置值	说 明
0	只显示 OK 按钮。
1	显示 OK 及 Cancel 按钮。
2	显示 Abort、Retry 及 Ignore 按钮。
3	显示 Yes、No 及 Cancel 按钮。
4	显示 Yes 及 No 按钮。
5	显示 Retry 及 Cancel 按钮。
16	显示 Critical Message 图标。
32	显示 Warning Query 图标。
48	显示 Warning Message 图标。
64	显示 Information Message 图标。
0	第一个按钮是缺省值。
256	第二个按钮 是缺省值。
512	第三个按钮是缺省值。
768	第四个按钮是缺省值。
0	应用程序被挂起，直到用户对消息框作出响应才继续工作。
4096	全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。

注意：

- 将这些数字相加以生成 buttons 参数值的时候，只能由每组值取用一个数字。

MsgBox 函数的功能不仅能够显示一个输出对话框，而且还能够返回一个整型的数值，把用户的反应通知给应用程序，如表 3-4 所示。

表 3-4 MsgBox 函数的返回值

返回值	用户的反应
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

下面以一个示例来说明 MsgBox 函数的用法：

1. 开始工作

首先启动一个新的项目，向空白的窗体上添加一个 Shape 控件，添加控件后的窗体如图 3-6 所示。

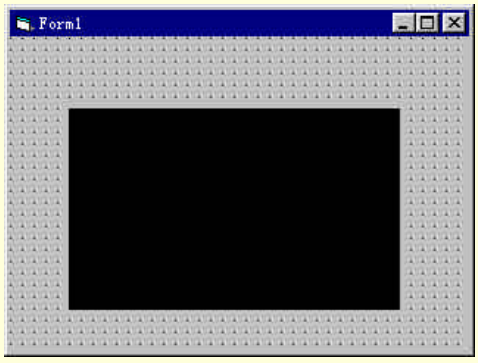


图 3-6 添加控件后的窗体

其中 Shape 控件的属性设置如表 3-5 所示。

表 3-5 Shape 控件的属性设置

属 性	设 置
(Name)	Shape1
BackStyle	0 - Transparent
BorderStyle	1 - Solid
FillColor	&H00000000&
FillStyle	0 - Solid
Height	2055
Left	600
Shape	0 - Rectangle
Top	720
Width	3375

这样设置的控件有以下的特性：

- 以实心的方式填充控件；
- 以黑色(&H00000000&)填充控件；
- 控件显示的是矩形。

2. 添加代码

在窗体的设计阶段，双击窗体，在窗体的 Form_Load()事件中添加下列代码：

```
Private Sub Form_Load()
Dim str As Integer
'定义一个整型变量
str = MsgBox("MsgBox 函数的用法：", 67, "MsgBox 函数")
'输出一个对话框
If str < 7 Then
Shape1.Shape = str - 1
End If '改变控件的形状
End Sub
```

提示：

运行程序，就会激活一个 Form_Load()事件，程序首先定义一个整型变量用来存储用户对输出对话框的响应，然后通过语句 Shape1.Shape = str - 1 把用户的响应转化到控件形状的改变上。

3. 运行程序

存储文件，运行程序，就会弹出一个如图 3-7 所示的对话框。

图 3-7 输出对话框

单击“确定”按钮，返回到运行的窗体，结果如图 3-8 所示，控件 Shape 的形状已经变成了圆角矩形。

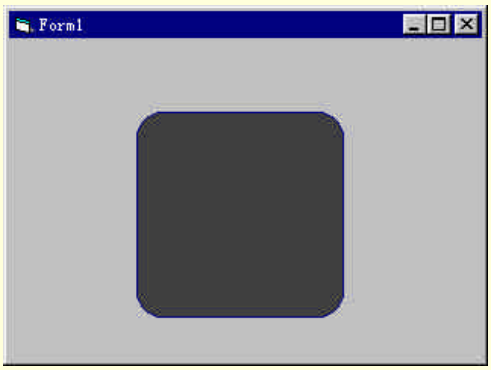


图 3-8 程序的运行结果

3.3 公共对话框控件

在 Visual Basic 6.0 中的公共对话框为用户提供了一组标准的操作对话框 Common Dialog，进行诸如打开和保存文件、设置打印选项以及选择颜色和字体等操作。

技巧：

如果想在程序中显示一个公共对话框，首先要将 CommonDialog 控件添加到缺省的工具箱中，然后才能够把它放置到窗体上设置其属性。

首先，我们需要设置对话框控件的属性。CommonDialog 控件所显示的对话框由控件的相应方法或者有 Action 属性的设置值来确定，如表 3-6 所示所示：

表 3-6 方法及其说明

方 法	说 明
ShowOpen	方法用来显示“打开”对话框
ShowSave	方法用来显示“另存为”对话框
ShowColor	方法用来显示“颜色”对话框
ShowFont	方法用来显示“字体”对话框
ShowPrinter	方法用来显示“打印”对话框
ShowHelp 方法用来调用 Windows	帮助

或者设置控件的 Action 属性值如表 3-7 所示：

表 3-7 属性说明

数 值	说 明
0	没有操作
1	显示“打开”对话框
2	显示“另存为”对话框
3	显示“颜色”对话框
4	显示“字体”对话框
5	显示“打印”对话框
6	运行 WINHLP32.EXE

CommonDialog 控件常用的属性和方法如表 3-8 所示。

表 3-8 控件常用的属性和方法

AboutBox 方法	ShowOpen 方法	ShowColor 方法
ShowPrinter 方法	ShowFont 方法	ShowSave 方法
ShowHelp 方法		
Action 属性	DialogTitle 属性	HelpFile 属性
FileName 属性	HelpKey 属性	FileTitle 属性
Filter 属性	InitDir 属性	Flags 属性
Max 属性	Min 属性	PrinterDefault 属性

选择菜单“工程”/“部件”，就会弹出一个如图 3-9 所示的添加控件的对话框，在其中选择 Microsoft Common Dialog Control 6.0，单击“确定”按钮，就向缺省的工具箱中添加了 CommonDialog 控件。

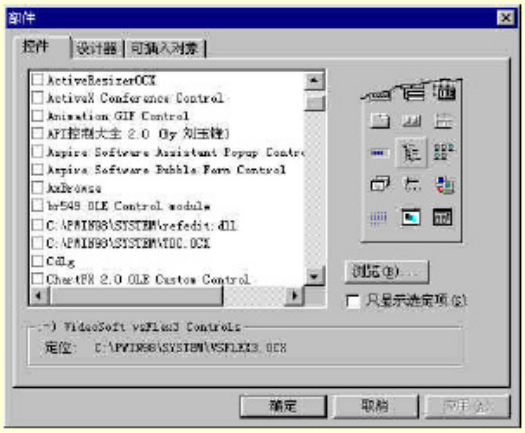


图 3-9 添加控件对话框

添加到窗体上的 CommonDiaolog 控件如图 3-10 所示。

图 3-10 CommonDiaolog 控件

提示：

> 标准对话框控件放置在工具箱上后，用户就可以象使用一般的控件那样使用它了，并且不用编写任何代码，请读者自己实践。

3.4 自定义对话框

在 Visual Basic 6.0 中对话框和普通的窗体没有什么区别，就其实质而言，对话框就是一个放置了控件的窗体，只不过它是在特定的条件下才显示，如图 3-11 所示即为一个用作对话框的窗体。

图 3-11 用作对话框的窗体

所以我们可以不但可以利用系统提供的现成的对话框，也可以自己定制对话框，下面就以一个示例来说明如何定制一个对话框。

1. 开始工作

首先启动一个新的项目，在窗体上放置一个 Shape 控件和两个 CommandButton 控件，它们的属性设置如表 3-9 所示。

表 3-9 控件与窗体的属性设置

控件或窗体	属 性	设 置
窗体 Form1	(Name)	Form1
	BorderStyle	3 - Fixed Dialog
	Caption	不同剖面线形式的 Shape 控件

	Height	3570
	Left	0
	Moveable	False
	StartUpPosition	2 - CenterScreen
	Top	0
	Width	4770
Shape 控件	(Name)	SProperties - Shape1
	BorderStyle	1 - Solid
	FillColor	&H00000000&
	FillStyle	7 - Diagonal Cross
	Height	1695
	Left	480
	Shape	4 - Rounded Rectangle
	Top	240
	Width	3495
CommandButton 控件	(Name)	Command1
	Caption	**选择剖面线**
	Height	495
	Left	360
	Top	2400
	Width	1215
CommandButton 控件	(Name)	Command2
	Caption	**结束运行**
	Height	495
	Left	2640
	Top	2400
	Width	1215

这样设置的窗体有如下的特性：

- 在窗体的标题栏中显示字符串“不同剖面线形式的 Shape 控件”；
- 在程序的运行过程中窗体不能够移动；
- 窗体不能够改变大小。

如表 3-6 设置 Shape 控件有如下的特性：

- 控件的填充模式为实心方式；
- 控件的显示形状为圆角矩形；
- 控件的剖面线形式为交叉线。

2．添加窗体

由于要定制一个对话框，而对话框又是特殊的窗体，所以要在现有的项目中添加一个标准的窗体，首先选择菜单“工程”/“添加窗体”，就会弹出如图 3-12 所示的添加窗体的对话框。

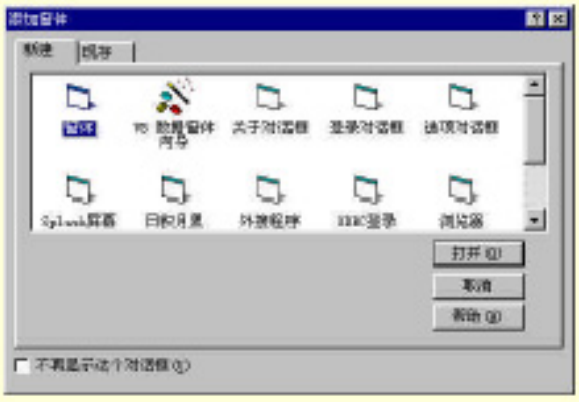


图 3-12 添加窗体的对话框

在“添加窗体”对话框中选择“窗体”项，单击“打开”按钮，就向当前的项目中添加了一个标准的窗体 Form2。添加窗体后的工程资源窗口如图 3-13 所示。

图 3-13　添加窗体后的工程资源窗口

3．为新窗体添加控件

在新添加的窗体 Form2 上放置一个 Frame 控件、两个 CommandButton 控件和七个 OptionButton 控件，其中 Frame 控件的作用是为七个 OptionButton 控件提供一个容器，而 OptionButton 控件用来设置窗体 Form1 中 Shape 控件的剖面线形式，而两个 CommandButton 控件分别用来完成剩余的“确定”和“取消”的功能。

添加控件后的窗体如图 3-14 所示。

图 3-14　添加控件后的窗体 Form2

其中窗体及控件的属性设置如表 3-10 所示。

表 3-10　窗体及控件的属性设置

窗体或控件	属　性	设　置
窗体 Form2	(Name)	Form2
	BorderStyle	3 - Fixed Dialog
	Caption	选择剖面线的形式
	Height	3255
	Left	0
	Moveable	False
	StartUpPosition	2 - CenterScreen
	Top	0
	Width	4770
Frame 控件	(Name)	Frame1
	Caption	剖面线形式
	Height	2175
	Left	360
	Top	360
	Width	3015
CommandButton 控件	(Name)	Command1
	Caption	确 定
	Height	495
	Left	3600
	Top	1200
	Width	975
CommandButton 控件	(Name)	Command2
	Caption	取 消
	Height	495
	Left	3600
	Top	2040
	Width	975

注意：

- 其余的七个 OptionButton 控件组成一个控件数组，名称从上到下依次为 Option(0)、Option(1)、Option(2)、Option(3)、Option(4)、Option(5)和 Option(6)。

4. 添加代码

首先给窗体 Form1 添加代码，在程序的设计阶段双击窗体，在窗体 Form1 的 Form_Load()事件中添加程序的初始化代码：

```
Private Sub Form_Load()
Form2.Hide
'隐藏窗体 Form2
End Sub
```

程序说明：

当程序开始运行时，首先装入窗体 Form1，与此同时通过 Form2.Hide 语句来隐藏窗体 Form2。

在程序的运行过程中，单击按钮“选择剖面线”，就要显示窗体 Form2，具体的代码如下：

```
Private Sub Command1_Click()
Form1.Hide
'隐藏窗体 Form1
Form2.Show
'显示窗体 Form2
End Sub
```

最后给窗体 Form2 添加代码，双击窗体 Form2 上的按钮“确定”，在它的 Command1_Click()事件中添加下列代码：

```
Private Sub Command1_Click()
Dim i As Integer
For i = 0 To 6 Step 1
If Option1(i).Value Then
Form1.Shape1.FillStyle = i + 1
End If
'改变剖面线的形式
Next
Form2.Hide
'隐藏窗体 Form2
Form1.Show
'显示窗体 Form1
End Sub
```

程序说明：

在程序的运行过程中，单击窗体 Form2 上的“确定”按钮时，程序就会根据用户的选择来改变窗体 Form1 中控件 Shape1 的剖面线的形式。

具体的语句如下：

```
If Option1(i).Value Then
Form1.Shape1.FillStyle = i + 1
End If
```

5. 运行程序

参照附后的源程序代码添加剩余的代码，然后选择菜单“工程”/“属性...”，就会弹出如图 3-15 所示的设置工程属性的对话框。

图 3-15 设置工程属性对话框

在“启动对象”选项框中选择启动的窗体为 Form1，在“工程类型”选项框中设置为“标准 EXE”，在“工程名称”选项框中设置工程的名字为“自定义对话框”，单击“确定”按钮进入下一步。

然后存储文件，按键盘上的功能键 F5 运行程序，程序运行的初始画面如图 3-16 所示。

图 3-16 程序运行的初始画面

单击“选择剖面线”按钮，就会弹出如图 3-17 所示的“选择剖面线的形式”对话框，在其中可以选择一种剖面线形式。

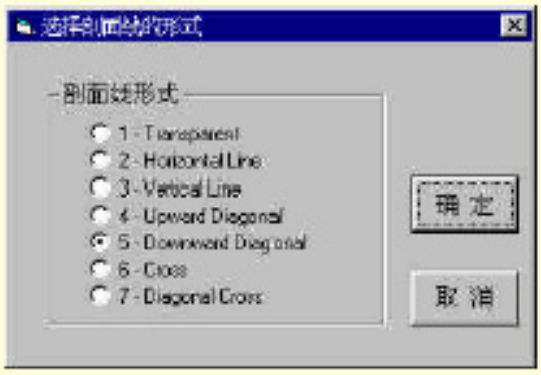


图 3-17 弹出对话框

在弹出的对话框中选择第五项“5 - Downward Diagonal”，单击“确定”按钮返回到程序运行的主窗体如图 3-18 所示。

图 3-18 程序运行结果

提示：

请读者仔细观察，矩形框内的剖面线形式由网格变为斜线。

附程序源代码如下：

程序清单

```
Begin Visual Basic.Form Form2
    Caption             =    "Form2"
    ClientHeight        =    2880
    ClientLeft          =    60
    ClientTop           =    345
    ClientWidth         =    4680
    LinkTopic           =    "Form2"
    ScaleHeight         =    2880
    ScaleWidth          =    4680
    StartUpPosition =    3   'Windows Default
    Begin Visual Basic.CommandButton Command2
        Caption             =    "取 消"
        BeginProperty Font
            Name                =    "宋体"
            Size                =    10.5
            Charset             =    134
            Weight              =    400
            Underline           =    0    'False
            Italic              =    0    'False
            Strikethrough       =    0    'False
        EndProperty
        Height              =    495
        Left                =    3600
        TabIndex            =    9
        Top                 =    2040
        Width               =    975
    End
    Begin Visual Basic.CommandButton Command1
        Caption             =    "确 定"
```

```
    BeginProperty Font
        Name            =    "宋体"
        Size            =    10.5
        Charset         =    134
        Weight          =    400
        Underline       =    0    'False
        Italic          =    0    'False
        Strikethrough   =    0    'False
    EndProperty
    Height          =    495
    Left            =    3600
    TabIndex        =    8
    Top             =    1200
    Width           =    975
End
Begin Visual Basic.OptionButton Option1
    Caption         =    "1 - Transparent"
    Height          =    195
    Index           =    0
    Left            =    720
    TabIndex        =    7
    Top             =    720
    Width           =    2175
End
Begin Visual Basic.OptionButton Option1
    Caption         =    "7 - Diagonal Cross"
    Height          =    195
    Index           =    6
    Left            =    720
    TabIndex        =    6
    Top             =    2160
    Width           =    2175
End
Begin Visual Basic.OptionButton Option1
    Caption         =    "6 - Cross"
    Height          =    195
    Index           =    5
    Left            =    720
    TabIndex        =    5
    Top             =    1920
    Width           =    2175
End
Begin Visual Basic.OptionButton Option1
    Caption         =    "5 - Downward Diagonal"
    Height          =    195
    Index           =    4
```

```
    Left            =   720
    TabIndex        =   4
    Top             =   1680
    Width           =   2175
End
Begin Visual Basic.OptionButton Option1
    Caption         =   "4 - Upward Diagonal"
    Height          =   195
    Index           =   3
    Left            =   720
    TabIndex        =   3
    Top             =   1440
    Width           =   2175
End
Begin Visual Basic.OptionButton Option1
    Caption         =   "3 - Vertical Line"
    Height          =   195
    Index           =   2
    Left            =   720
    TabIndex        =   2
    Top             =   1200
    Width           =   2175
End
Begin Visual Basic.OptionButton Option1
    Caption         =   "2 - Horizontal Line"
    Height          =   195
    Index           =   1
    Left            =   720
    TabIndex        =   1
    Top             =   960
    Width           =   2175
End
Begin Visual Basic.Frame Frame1
    Caption         =   "剖面线形式"
    BeginProperty Font
        Name            =   "宋体"
        Size            =   10.5
        Charset         =   134
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   2175
    Left            =   360
    TabIndex        =   0
```

```
        Top             =     360
        Width           =     3015
    End
End
Attribute Visual Basic_Name = "Form2"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False
Private Sub Command1_Click()
Dim i As Integer
For i = 0 To 6 Step 1
If Option1(i).Value Then
Form1.Shape1.FillStyle = i
End If
'改变剖面线的形式
Next
Form2.Hide
'隐藏窗体 Form2
Form1.Show
'显示窗体 Form1
End Sub

Private Sub Command2_Click()
Form2.Hide
'隐藏窗体 Form2
Form1.Show
'显示窗体 Form1
End Sub
```

窗体 Form2 的代码如下：

```
Begin Visual Basic.Form Form1
    Caption             =     "Form1"
    ClientHeight        =     3195
    ClientLeft          =     60
    ClientTop           =     345
    ClientWidth         =     4680
    LinkTopic           =     "Form1"
    ScaleHeight         =     3195
    ScaleWidth          =     4680
    StartUpPosition =     3    'Windows Default
    Begin Visual Basic.CommandButton Command2
        Caption             =     "结束运行"
        BeginProperty Font
            Name                =     "宋体"
            Size                =     10.5
            Charset             =     134
```

```
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
        EndProperty
        Height          =   495
        Left            =   2640
        TabIndex        =   1
        Top             =   2400
        Width           =   1215
    End
    Begin Visual Basic.CommandButton Command1
        Caption         =   "选择剖面线"
        BeginProperty Font
            Name            =   "宋体"
            Size            =   10.5
            Charset         =   134
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
        EndProperty
        Height          =   495
        Left            =   360
        TabIndex        =   0
        Top             =   2400
        Width           =   1215
    End
    Begin Visual Basic.Shape Shape1
        FillStyle       =   6   'Cross
        Height          =   1695
        Left            =   480
        Shape           =   4   'Rounded Rectangle
        Top             =   360
        Width           =   3495
    End
End
Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False
Private Sub Command1_Click()
Form1.Hide
Form2.Show
End Sub
```

```
Private Sub Command2_Click()
End
'结束程序的运行
End Sub

Private Sub Form_Load()
Form2.Hide
'隐藏窗体 Form2
End Sub
```

3.5 小 结

在应用程序的设计和运行过程中，很重要的一个方面就是程序如何能够更好的与程序员和用户进行交流，所以笔者特意把程序与用户和程序员的交流单独的列出一章来。

程序与用户或者程序员的交流的手段是很多的，在本章中介绍了一种常用的方法，那就式用户自己设计的设计输入输出对话框以及自定义对话框和公共对话框。对话框的设计技术并不是特别难，关键是如何合理的实现对话框在整个程序中的功能，并且对话框的风格和工程项目的特色一致。

第四章 Windows 用户窗体的设计

用户窗体的界面设计可以分为三个方面的内容：第一个是利用 VCL 控件（可视化控件）的窗体界面设计，另外一个是多重窗体的设计（Multiple Forms），还有一个就是经常提到的多文档界面的设计，即 MDI。

注意：

✍ 我们重点需要了解多重窗体和多文档界面窗体的区别。

多重窗体（也叫复合窗体）是指在一个应用程序中有多个窗口界面，一个个分别的显示在屏幕上，而多文档界面是指在一个父窗口中包含有多个子窗口，在程序运行过程中可以同时打开多个窗体，这些窗体之间的关系由程序指定。

下面将分别对三种窗体的界面设计方法加以介绍。

4.1 VCL 控件窗体设计

一个应用程序的好与坏，除了算法的设计，代码的编写方面的因素外，用户界面的设计也是一个非常重要的方面，在可视化的编程环境中，虽然不用编写大量的代码就可以设计出丰富的界面，但是要想设计出有个性有特色的用户界面，还是需要程序的开发人员下一番功夫的。

下面就是一个利用 VCL 控件来进行程序设计的示例，在这个示例程序中所用到的一些控件在程序的初始阶段没有被加载，还需要读者自己把它们添加到工具箱中。

示例程序的具体步骤如下：

1．启动一个新项目

选择菜单“工程”/“部件”，在弹出的对话框中选择所需要的，添加控件后的工具箱如图 4-1 所示。

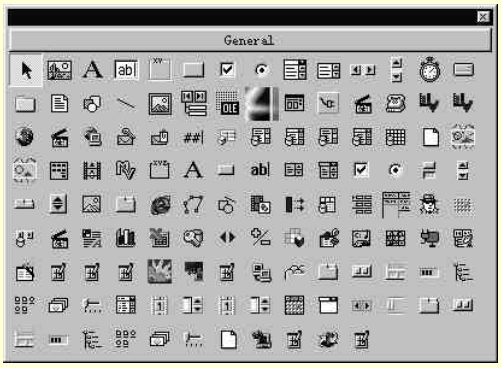


图 4-1 添加控件后的工具箱

注意：

✍ 选择菜单“工程”/“部件”的意思是选择“工程”菜单上的“部件”命令，以下同。

2．SSTab 控件

在 Visual Basic 6.0 所提供的控件中，有一些控件可以作为容器控件，在其上可以放置一些 VCL 控件来进行控件的窗体界面设计，如 Frame 控件和 SSTab 控件等，下面就来介绍一下 SSTab 控件。

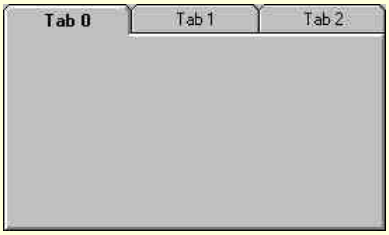


图 4-2 在窗体上的 SSTab 控件

它的属性设置如表 4-1 所示。

表 4-1 SSTab 控件的属性设置

属 性	设 置
(Name)	SSTab1
Caption	Tab 1
Enabled	True
Height	2415
Left	240
Picture	(None)
Tab	1
TabHeight	295
Tabs	3
TabsPerRow	3
Top	120
Width	4095
WordWrap	True

说明：

控件的名称为 SSTab1，它有三个选项，即选项 0、选项 1 和选项 2，现在选中的是选项 1，现在处于有效的状态，它的显示文字为“ Tab 1 ”，图片为空。

SSTab 控件的状态可以在程序中动态的修改，比如在窗体的设计阶段双击窗体，在窗体的 Form_Load()事件中添加下列代码：

```
Private Sub Form_Load()
SSTab1.TabCaption(1) = "选项 1"
'设置选项 1 上的文本
SSTab1.TabVisible(2) = False
'设置选项 2 上的可见状态
SSTab1.TabEnabled(0) = False
'设置选项 0 的有效状态
SSTab1.Tab = 1
'设置选项 1 为选中的状态
SSTab1.Tabs = 4
'添加一个选项
End Sub
```

保存文件，运行程序，结果如图 4-3 所示。

图 4-3 程序运行结果

3. 向窗体上添加控件

向窗体上添加一个 CommandButton 控件、两个 TextBox 控件、一个 SSTab 控件、四个 ToggleButton 控件、两个 UpDown 控件和四个 Label 控件。

添加控件后的窗体如图 4-4 所示。

图 4-4 添加控件后的窗体

控件说明：

- CommandButton 控件用来结束程序的运行，为程序的正常结束提供一个出口；
- TextBox 控件用来显示程序中的两个变量 X 和 Y 的值；
- UpDown 控件用来实现变量的动态更新；
- ToggleButton 控件用来实现不同的加减乘除运算；
- Label 控件用来显示文本和运算结果；
- SSTab 控件用来显示不同的选项，同时作为 ToggleButton 控件的容器。

4. 添加代码

下面以乘法操作为例来添加代码，在程序的设计阶段双击 ToggleButton3 控件，在它的 ToggleButton3_Click() 事件中添加下列代码：

```
Private Sub ToggleButton3_Click()
Label4.Caption = Val(Text1.Text) * Val(Text2.Text)
'乘法操作
End Sub
```

其余控件的代码添加过程在这里就不加赘述，详细的代码请参看附后的完整源程序代码。

5. 运行程序

添加完代码后，按功能键 F5 运行程序，程序运行的初始画面如图 4-5 所示。

图 4-5 程序运行初始画面

在文本框中输入变量 X 和 Y 的值，选择一种算法，如除法操作，单击相应的按钮，结果如图 4-6 所示。

图 4-6 程序的运行结果

附程序的完整代码：

程序清单

```
Version 6.00
Private Sub Form_Load()
SSTab1.TabCaption(0) = "加法"
SSTab1.TabCaption(1) = "减法"
SSTab1.TabCaption(2) = "乘法"
SSTab1.TabCaption(3) = "除法"
'设置控件的显示文本
If Text1.Text = "" Then
Text1.Text = "0.00000"
End If
'初始化文本框
If Text2.Text = "" Then
Text2.Text = "0.00000"
End If
'初始化文本框
End Sub

Private Sub Text2_Change()
If Text2.Text = 0 Then
ToggleButton4.Enabled = False
Else
ToggleButton4.Enabled = True
End If
'如果文本框 2 中的数字为 0
'那么除法无效
End Sub

Private Sub ToggleButton1_Click()
Label4.Caption = Val(Text1.Text) + Val(Text2.Text)
'加法操作
End Sub
```

```
Private Sub ToggleButton2_Click()
Label4.Caption = Val(Text1.Text) - Val(Text2.Text)
'减法操作
End Sub

Private Sub ToggleButton3_Click()
Label4.Caption = Val(Text1.Text) * Val(Text2.Text)
'乘法操作
End Sub

Private Sub ToggleButton4_Click()
Label4.Caption = Val(Text1.Text) / Val(Text2.Text)
'除法操作
End Sub

Private Sub UpDown1_DownClick()
If Text1.Text > 0 Then
Text1.Text = Text1.Text - 1
End If
'单击向下的箭头，数字减 1
End Sub

Private Sub UpDown1_UpClick()
If Text1.Text = "" Then
Text1.Text = 0
End If
Text1.Text = Text1.Text + 1
'单击向上的箭头，数字加 1
End Sub

Private Sub UpDown2_DownClick()
If Text2.Text > 0 Then
Text2.Text = Text2.Text - 1
End If
'单击向下的箭头，数字减 1
End Sub

Private Sub UpDown2_UpClick()
If Text2.Text = "" Then
Text2.Text = 0
End If
Text2.Text = Text2.Text + 1
'单击向上的箭头，数字加 1
End Sub
```

4.2 多重窗体设计

一个大的应用程序往往不会只包括一个窗体，否则程序就显得有一些单薄，而多窗体程序设计可以使程序变的丰富多彩，在多重窗体中每个单独的窗体都有自己的特点与功能，所以程序的功能会更加强大。

4.2.1 多重窗体设计常用方法

在多窗体的程序设计中经常要用到下面四种方法：Load 方法、Show 方法、Hide 方法和 Unload 方法。

1．Load 方法

它的语法结构如下：

Load [窗体名称]

使用 Load 方法调用的窗体只是被装入内存，并不会显示出来，与此同时，会产生一个 Form_Load()事件。如：

```
Load Form2
```

2．Show 方法

它的语法结构如下：

[窗体名称].Show

使用 Show 方法会显示被调用的窗体，如果在调用 Show 方法之前没有把窗体调入内存，那么调用 Show 方法会自动的把窗体调入内存。如：

```
Load Form2
Form2.Show
```

3．Hide 方法

它的语法结构如下：

[窗体名称].Hide

使用 Hide 方法会隐藏被调用的窗体，但是在调用 Hide 方法之后不会把窗体移出内存，但是被调用的窗体中的属性等已经处于无效的状态。如：

```
Form1.Hide
Form2.Show
```

4．Unload 方法

它的语法结构如下：

Unload [窗体名称]

使用 Unload 方法会从内存中移去被调用的窗体，与此同时，窗体中的变量和属性等都会处于无效的状态，在移去窗体的同时会产生一个 Form_QueryUnload()事件。如：

```
Form1.Show
Unload Form2
```

4.2.2 多重窗体设计实例

下面就以一个示例来说明如何进行多重窗体的程序设计，具体的设计步骤如下：

1．启动新项目

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，窗体的属性设置如表 4-2 所示。

表 4-2 窗体的属性设置

属 性	设 置
(Name)	Form1
BorderStyle	3 - Fixed Dialog
Caption	Form1
Enabled	True
Height	2940
Left	0
MDIChild	False

Moveable	False
StartUpPosition	2 – CenterScreen
Top	0
Visible	True
Width	4065

这样设置的窗体有以下特性：

- 窗体在程序的运行过程中始终位于屏幕的中央；
- 窗体的大小不能够在程序的运行过程中改变；
- 在程序的运行过程中不能够移动窗体。

2. 添加控件

在空白的窗体上添加窗体如下控件：五个 Label 控件、四个 TextBox 控件和两个 CommandButton 控件。添加控件后的窗体如图 4-6 所示。

图 4-7 添加控件后的窗体

技巧：

添加上述几类常用控件的方法非常简单，只需要在“工具箱”上双击需要的控件即可。

其中控件的属性设置如表 4-3 所示。

表 4-3 控件的属性设置

属 性	设 置
(Name)	Label1
Caption	企业季度销售情况
(Name)	Label2
Caption	第一季度
(Name)	Label3
Caption	第二季度
(Name)	Label4
Caption	第三季度
(Name)	Label5
Caption	第四季度
(Name)	Text1
Text	
(Name)	Text1
Text	
(Name)	Text1
Text	
(Name)	Text1
Text	
(Name)	Command1
Caption	图 形
(Name)	Command2
Caption	退 出

由于篇幅有限，所以在这里代码的添加过程就不加赘述了，详细的代码请参看附后的完整程序代码。

3. 添加窗体

选择菜单“工程”/“添加窗体”，就会弹出如图 4-8 所示的对话框。

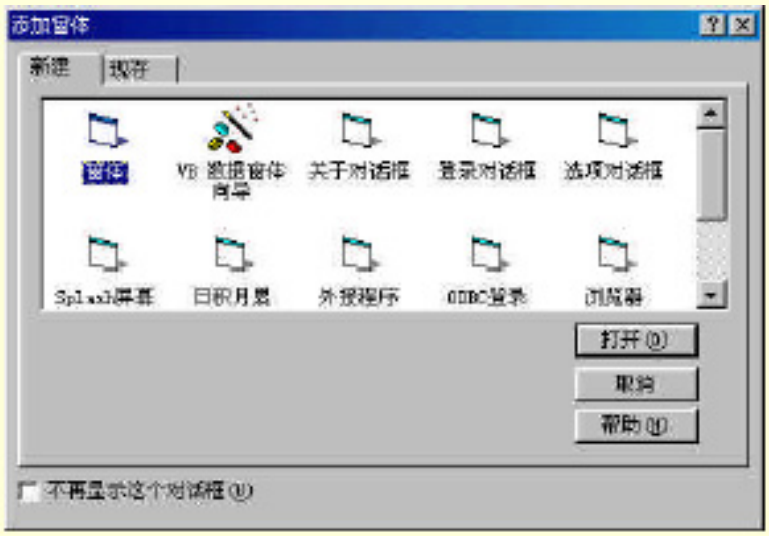


图 4-8　添加窗体对话框

在“添加窗体”对话框中选择“窗体”项，单击“打开”按钮，系统就会自动的向原有的项目中添加一个窗体。

4．添加控件

在新增的窗体上放置一个 CommandButton 控件，它的作用是在隐藏新增的窗体的同时显示原有的窗体。添加控件后的新增窗体如图 4-9 所示。

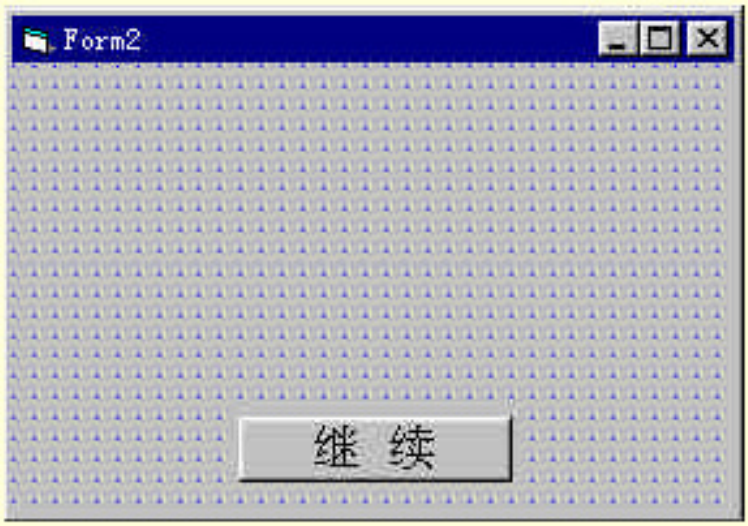


图 4-9　添加控件后的新增窗体

窗体及控件的属性设置如表 4-4 所示。

表 4-4　窗体的属性设置

项　目	内　容
(Name)	Form2
BorderStyle	3 - Fixed Dialog
Caption	Form2
Height	2970
Left	0
Moveable	False
StartUpPosition	2 – CenterScreen
Top	0
Width	4275
(Name)	Command1
Caption	继续

在“继续”按钮的 Command1_Click()**事件中添加下列代码：**

```
Private Sub Command1_Click()
Form2.Hide
'隐藏窗体 2
Form1.Show
'显示窗体 1
End Sub
```

在窗体 Form2 **被显示的过程中，单击“继续”按钮，就会激活** Command1_Click()**事件，然后通过** Form2.Hide **和** Form1.Show **两条语句实现隐藏窗体** 2 **和显示窗体的功能。**

5. **添加说明模块**

选择菜单“工程”/“添加模块”，就会弹出如图 4-10 **所示的对话框。**

图 4-10 **添加模块对话框**

在模块的声明段中添加下列代码：

```
Global a1 As Single
Global a2 As Single
Global a3 As Single
Global a4 As Single
'定义四个全局变量
```

注意：

- **定义的四个全局变量分别用来存储第一季度、第二季度、第三季度和第四季度的销售情况。**

添加模块和新窗体的工程资源窗口如图 4-11 **所示。**

图 4-11 **工程资源窗口**

6. 运行程序

依附后的完整程序代码，添加剩余的代码，存储文件，运行程序，初始画面如图 4-12 所示。

图 4-12 程序运行初始画面

单击“图形”按钮，结果如图 4-13 所示。

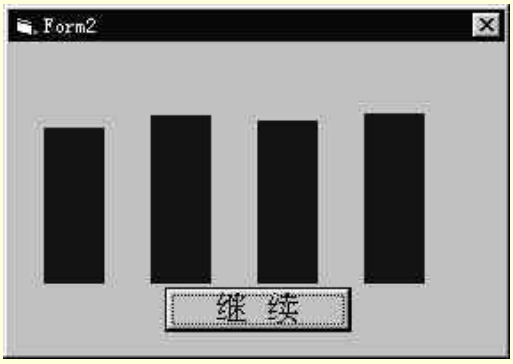


图 4-13 程序的运行结果——显示窗体 Form2

附程序的完整源代码：

程序清单（多窗体.bas 文件内容）

```
Attribute VB_Name = "Module1"
Global a1 As Single
Global a2 As Single
Global a3 As Single
Global a4 As Single
'定义四个全局变量
```

程序清单（多窗体 1.frm 内容）

```
VERSION 6.00
Begin VB.Form Form1
   Caption         =   "Form1"
   ClientHeight    =   2565
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   3975
   LinkTopic       =   "Form1"
   ScaleHeight     =   2565
   ScaleWidth      =   3975
```

```
StartUpPosition =     3   'Windows Default
Begin VB.CommandButton Command2
   Caption             =     "退 出"
   BeginProperty Font
      Name               =     "宋体"
      Size               =     14.25
      Charset            =     134
      Weight             =     400
      Underline          =     0     'False
      Italic             =     0     'False
      Strikethrough      =     0     'False
   EndProperty
   Height              =     375
   Left                =     2280
   TabIndex            =     10
   Top                 =     2040
   Width               =     1215
End
Begin VB.CommandButton Command1
   Caption             =     "图 形"
   BeginProperty Font
      Name               =     "宋体"
      Size               =     14.25
      Charset            =     134
      Weight             =     400
      Underline          =     0     'False
      Italic             =     0     'False
      Strikethrough      =     0     'False
   EndProperty
   Height              =     375
   Left                =     480
   TabIndex            =     9
   Top                 =     2040
   Width               =     1215
End
Begin VB.TextBox Text4
   BeginProperty Font
      Name               =     "宋体"
      Size               =     10.5
      Charset            =     134
      Weight             =     400
      Underline          =     0     'False
      Italic             =     0     'False
      Strikethrough      =     0     'False
   EndProperty
   Height              =     495
```

```
    Left            =   3000
    TabIndex        =   8
    Top             =   1440
    Width           =   855
End
Begin VB.TextBox Text3
    BeginProperty Font
        Name            =   "宋体"
        Size            =   10.5
        Charset         =   134
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   495
    Left            =   2040
    TabIndex        =   7
    Top             =   1440
    Width           =   855
End
Begin VB.TextBox Text2
    BeginProperty Font
        Name            =   "宋体"
        Size            =   10.5
        Charset         =   134
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   495
    Left            =   1080
    TabIndex        =   6
    Top             =   1440
    Width           =   855
End
Begin VB.TextBox Text1
    BeginProperty Font
        Name            =   "宋体"
        Size            =   10.5
        Charset         =   134
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
```

```
    EndProperty
    Height          =   495
    Left            =   120
    TabIndex        =   5
    Top             =   1440
    Width           =   855
End
Begin VB.Label Label5
    Caption         =   "第一季度"
    BeginProperty Font
        Name            =   "宋体"
        Size            =   12
        Charset         =   134
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   615
    Left            =   3120
    TabIndex        =   4
    Top             =   840
    Width           =   615
End
Begin VB.Label Label4
    Caption         =   "第一季度"
    BeginProperty Font
        Name            =   "宋体"
        Size            =   12
        Charset         =   134
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   615
    Left            =   2160
    TabIndex        =   3
    Top             =   840
    Width           =   615
End
Begin VB.Label Label3
    Caption         =   "第一季度"
    BeginProperty Font
        Name            =   "宋体"
        Size            =   12
```

```
        Charset         =   134
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   615
    Left            =   1200
    TabIndex        =   2
    Top             =   840
    Width           =   615
End
Begin VB.Label Label2
    Caption         =   "第一季度"
    BeginProperty Font
        Name            =   "宋体"
        Size            =   12
        Charset         =   134
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   495
    Left            =   240
    TabIndex        =   1
    Top             =   840
    Width           =   615
End
Begin VB.Label Label1
    Caption         =   "企业季度销售情况"
    BeginProperty Font
        Name            =   "宋体"
        Size            =   15
        Charset         =   134
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height          =   375
    Left            =   720
    TabIndex        =   0
    Top             =   240
    Width           =   2535
End
```

```
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Command1_Click()
Form1.Hide
'隐藏窗体 1
Form2.Show
'显示窗体 2
End Sub

Private Sub Command2_Click()
End
'结束运行
End Sub

Private Sub Text1_Change()
a1 = Val(Text1.Text)
'转换变量
End Sub

Private Sub Text2_Change()
a2 = Val(Text2.Text)
'转换变量
End Sub

Private Sub Text3_Change()
a3 = Val(Text3.Text)
'转换变量
End Sub

Private Sub Text4_Change()
a4 = Val(Text4.Text)
'转换变量
End Sub
```

程序清单（多窗体 2.frm 文件内容）

```
Version 6.00
Begin VB.Form Form2
   AutoRedraw      =   -1  'True
   Caption         =   "Form2"
   ClientHeight    =   2595
   ClientLeft      =   60
```

```
    ClientTop          =     345
    ClientWidth        =     4185
    ForeColor          =     &H00FF0000&
    LinkTopic          =     "Form2"
    ScaleHeight        =     2595
    ScaleWidth         =     4185
    StartUpPosition =     3   'Windows Default
    Begin VB.CommandButton Command1
        Caption             =     "继 续"
        BeginProperty Font
            Name                =     "宋体"
            Size                =     14.25
            Charset             =     134
            Weight              =     400
            Underline           =     0     'False
            Italic              =     0     'False
            Strikethrough       =     0     'False
        EndProperty
        Height              =     375
        Left                =     1320
        TabIndex            =     0
        Top                 =     2040
        Width               =     1575
    End
End
Attribute VB_Name = "Form2"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Command1_Click()
Form2.Hide
'隐藏窗体 2
Form1.Show
'显示窗体 1
End Sub

Private Sub Form_Load()
Form2.AutoRedraw = True
'自动重绘处于有效的状态
Form2.ForeColor = &HFF0000
'设置前景色
Line (300, 1000 - a1)-(800, 2000), , BF
Line (1200, 1000 - a2)-(1700, 2000), , BF
Line (2100, 1000 - a3)-(2600, 2000), , BF
Line (3000, 1000 - a4)-(3500, 2000), , BF
```

'绘制季度销售方框

End Sub

4.3 多文档界面窗体设计

多文档界面（Multiple Document Interface）是指在一个父窗口下面可以同时打开多个子窗口。

注意：

✍ **为了更清楚的了解多文档界面窗体的含义，用户在学习的时候请和上节所讲的内容做比较。**

下面以一个示例来说明如何利用 Visual Basic 6.0 中的 MDI 进行程序的设计，示例程序的具体步骤如下：

1. **开始工作**

首先启动一个新的项目,在屏幕上会出现一个空白的窗体，窗体的属性设置如表 4-5 所示。

表 4-5 窗体的属性设置

属 性	设 置
(Name)	Form1
AutoRedraw	True
BorderStyle	2 – Sizable
Caption	Form1
Height	3600
Left	0
MDIChild	True
Moveable	True
Top	0
Width	4800

这样设置的窗体有以下的特性：

- **窗体的自动重绘处于有效的状态；**
- **在程序的运行过程中窗体可以改变大小；**
- **窗体可以作为 MDI 窗体的子窗体；**
- **在程序的运行过程窗体可以移动。**

2. **添加 MDI 窗体**

选择菜单“工程”/“添加 MDI 窗体”，就会弹出如图 4-14 所示的对话框。

图 4-14 添加 MDI 窗体对话框

单击“打开“按钮，添加一个 MDI 窗体，窗体的属性设置如表 4-6 所示。

表 4-6 MDI 窗体的属性设置

属 性	设 置
(Name)	MDIForm1
AutoShowChildren	False
Caption	MDIForm1
Height	3885

Left	105
Moveable	False
StartUpPosition	2 - CenterScreen
Top	-180
Width	4800

这样设置的 MDI 窗体有如下特性：

- **MDI 窗体不能够自动的显示子窗体；**
- **窗体在程序的运行过程中不能移动；**
- **窗体始终位于屏幕的中央。**

3. **添加子窗体**

选择菜单“工程”/“添加窗体”，就会弹出如图 4-15 所示的对话框。

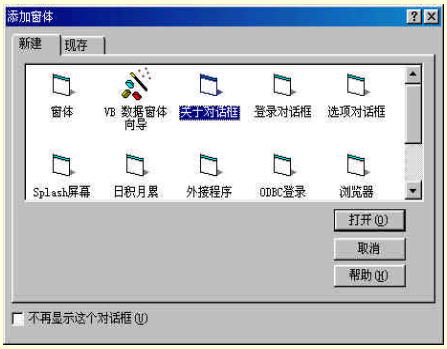


图 4-15 添加子窗体对话框

在“添加窗体”对话框中选择“关于对话框”，单击“打开“按钮，就在 MDI 窗体上添加了一个子窗体，子窗体 FrmAbout 的属性设置如表 4-7 所示。

表 4-7 子窗体 FrmAbout 的属性设置

属 性	设 置
(Name)	FrmAbout
BorderStyle	3 - Fixed Dialog
Caption	About MyApp
Enabled	True
Height	3930
Left	2295
MDIChild	True
Moveable	True
Top	1605
Visible	True
Width	5820

这样设置的子窗体有如下的特性：

- **窗体在程序的运行过程中不能改变大小；**
- **程序运行时可以移动窗体；**
- **窗体为 MDI 窗体的一个子窗体。**

4. **为子窗体 Form1 添加代码**

在程序的设计阶段双击子窗体 Form1，在它的 Form_Load()过程中添加下列代码：

```
Private Sub Form_Load()
Dim x As Integer
For x = 0 To 3000 Step 75
    Line (0, 1000)-(x, 0)
    Line (0, 1000)-(x, 2000)
    Line (3000, 1000)-(3000 - x, 2000)
```

```
    Line (3000, 1000)-(3000 - x, 0)
Next
End Sub
```

提示：

程序首先定义了一个整型变量 x，然后进入了一个循环，在循环中通过四个画直线的语句来实现程序中绘制图形的功能。

5. 为 MDI 窗体添加菜单

选择菜单“工具”菜单上的“菜单编辑器”，就会弹出如图 4-16 所示的对话框。

在菜单编辑器中，建立两个菜单项——“文件”和“退出”，在“文件”项下还有两个子菜单——“绘图”和“关于”，它们的属性设置如表 4-8 所示。

添加菜单后的 MDI 窗体如图 4-17 所示。

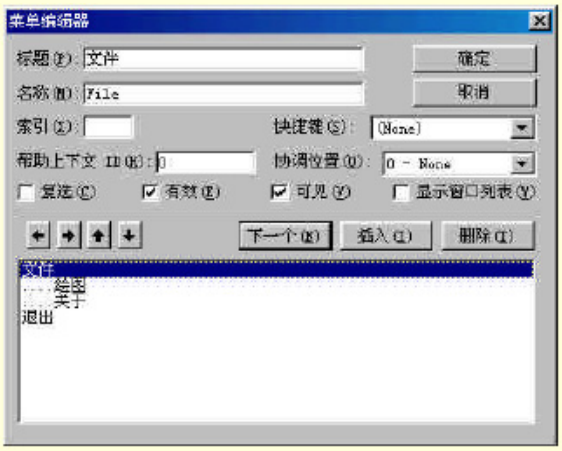


图 4-16 菜单编辑器对话框

表 4-8 菜单属性

属 性	设 置
Caption	**文件**
Name	File
Caption	**绘图**
Name	Draw
Caption	**关于**
Name	About
Caption	**退出**
Name	Exit

图 4-17 添加菜单后的 MDI 窗体

6. 添加代码

由于在本节的后面附了程序的完整源代码，所以在这里就不对代码的添加过程做更多的阐述，读者如果有不清楚的地方，请参见附后的源代码。

存储文件，运行程序，初始画面如图 4-18 所示。

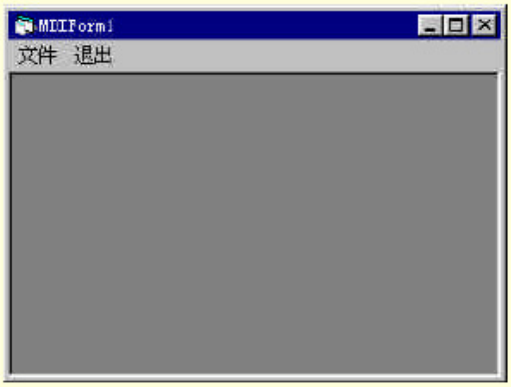


图 4-18 程序运行初始画面

单击菜单“文件”/“绘图”，就会弹出如图 4-19 所示的绘图子窗体。

图 4-19 绘图子窗体

单击菜单“文件”/“关于”，就会弹出如图 4-20 所示的 About 子窗体。

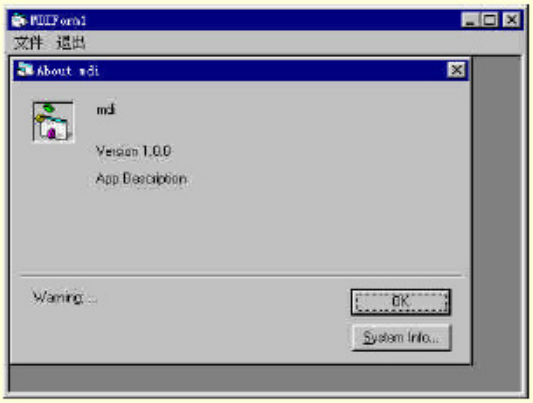


图 4-20 About 子窗体

附程序完整源代码：

程序清单（MDIForm1.frm 文件内容）

```
VERSION 6.00
```

```
Begin VB.MDIForm MDIForm1
   AutoShowChildren=   0   'False
   BackColor       =   &H8000000C&
   Caption         =   "MDIForm1"
   ClientHeight    =   3195
   ClientLeft      =   165
   ClientTop       =   735
   ClientWidth     =   4680
   LinkTopic       =   "MDIForm1"
   StartUpPosition =   3   'Windows Default
   Begin VB.Menu File
      Caption         =   "文件"
      Begin VB.Menu Draw
         Caption         =   "绘图"
      End
      Begin VB.Menu About
         Caption         =   "关于"
      End
   End
   Begin VB.Menu Exit
      Caption         =   "退出"
   End
End
Attribute VB_Name = "MDIForm1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub About_Click()
Form1.Hide
'隐藏窗体 Form1
frmAbout.Show
'显示窗体 frmAbout
End Sub

Private Sub Draw_Click()
frmAbout.Hide
'隐藏窗体 frmAbout
Form1.Show
'显示窗体 Form1
End Sub

Private Sub Exit_Click()
End
'结束运行
End Sub
```

程序清单（Form1.frm 文件内容）

```
VERSION 6.00
Begin VB.Form Form1
   AutoRedraw      =   -1  'True
   Caption         =   "Form1"
   ClientHeight    =   3195
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   4680
   LinkTopic       =   "Form1"
   MDIChild        =   -1  'True
   ScaleHeight     =   3195
   ScaleWidth      =   4680
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Form_Load()
Dim x As Integer
For x = 0 To 3000 Step 75
    Line (0, 1000)-(x, 0)
    Line (0, 1000)-(x, 2000)
    Line (3000, 1000)-(3000 - x, 2000)
    Line (3000, 1000)-(3000 - x, 0)
Next
End Sub
```

程序清单（frmAbout.frm 文件内容）

```
Version 6.00
Begin VB.Form frmAbout
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "About MyApp"
   ClientHeight    =   3555
   ClientLeft      =   2340
   ClientTop       =   1935
   ClientWidth     =   5730
   ClipControls    =   0   'False
   LinkTopic       =   "Form2"
   MaxButton       =   0   'False
   MDIChild        =   -1  'True
   MinButton       =   0   'False
   ScaleHeight     =   2453.724
   ScaleMode       =   0  'User
```

```
ScaleWidth      =   5380.766
ShowInTaskbar   =   0   'False
Begin VB.PictureBox picIcon
   AutoSize        =   -1  'True
   ClipControls    =   0   'False
   Height          =   540
   Left            =   240
   Picture         =   "frmAbout.frx":0000
   ScaleHeight     =   337.12
   ScaleMode       =   0   'User
   ScaleWidth      =   337.12
   TabIndex        =   1
   Top             =   240
   Width           =   540
End
Begin VB.CommandButton cmdOK
   Cancel          =   -1  'True
   Caption         =   "OK"
   Default         =   -1  'True
   Height          =   345
   Left            =   4245
   TabIndex        =   0
   Top             =   2625
   Width           =   1260
End
Begin VB.CommandButton cmdSysInfo
   Caption         =   "&System Info..."
   Height          =   345
   Left            =   4260
   TabIndex        =   2
   Top             =   3075
   Width           =   1245
End
Begin VB.Line Line1
   BorderColor     =   &H00808080&
   BorderStyle     =   6   'Inside Solid
   Index           =   1
   X1              =   84.515
   X2              =   5309.398
   Y1              =   1687.583
   Y2              =   1687.583
End
Begin VB.Label lblDescription
   Caption         =   "App Description"
   ForeColor       =   &H00000000&
   Height          =   1170
```

```
        Left            =   1050
        TabIndex        =   3
        Top             =   1125
        Width           =   3885
    End
    Begin VB.Label lblTitle
        Caption         =   "Application Title"
        ForeColor       =   &H00000000&
        Height          =   480
        Left            =   1050
        TabIndex        =   5
        Top             =   240
        Width           =   3885
    End
    Begin VB.Line Line1
        BorderColor     =   &H00FFFFFF&
        BorderWidth     =   2
        Index           =   0
        X1              =   98.6
        X2              =   5309.398
        Y1              =   1697.936
        Y2              =   1697.936
    End
    Begin VB.Label lblVersion
        Caption         =   "Version"
        Height          =   225
        Left            =   1050
        TabIndex        =   6
        Top             =   780
        Width           =   3885
    End
    Begin VB.Label lblDisclaimer
        Caption         =   "Warning: ..."
        ForeColor       =   &H00000000&
        Height          =   825
        Left            =   255
        TabIndex        =   4
        Top             =   2625
        Width           =   3870
    End
End
Attribute VB_Name = "frmAbout"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

```
Option Explicit

' Reg Key Security Options...
Const READ_CONTROL = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + _
                       KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
                       KEY_NOTIFY + KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE = &H80000002
Const ERROR_SUCCESS = 0
Const REG_SZ = 1                        ' Unicode nul terminated string
Const REG_DWORD = 4                     ' 32-bit number

Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC = "MSINFO"
Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

Private Sub cmdSysInfo_Click()
   Call StartSysInfo
End Sub

Private Sub cmdOK_Click()
   Unload Me
End Sub

Private Sub Form_Load()
      Me.Caption = "About " & App.Title
      lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
      lblTitle.Caption = App.Title
End Sub

Public Sub StartSysInfo()
```

```
    On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String

    ' Try To Get System Info Program Path\Name From Registry...
    If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO, SysInfoPath) Then
    ' Try To Get System Info Program Path Only From Registry...
    ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC, gREGVALSYSINFOLOC, SysInfoPath) Then
        ' Validate Existance Of Known 32 Bit File Version
        If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
            SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

        ' Error - File Can Not Be Found...
        Else
            GoTo SysInfoErr
        End If
    ' Error - Registry Entry Can Not Be Found...
    Else
        GoTo SysInfoErr
    End If

    Call Shell(SysInfoPath, vbNormalFocus)

    Exit Sub
    SysInfoErr:
    MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub
```

4.4 小 结

通过本章的学习，读者可以对 Visual Basic 6.0 **程序设计中的界面设计有了深入的认识，尤其是对可视化编程有了更加直观的学习。**

本章集中的介绍了界面设计的三种方法：第一个是利用 VCL **控件进行的窗体界面设计，另外一个是多重窗体的设计（**Multiple Forms**），还有一个就是经常提到的多文档界面的设计，即** MDI（Multiple Document Interface）。

提示：

> **学习了本章的几个英文缩写，用户在以后再看到** MDI、SDI（**单文档界面**）**和** MF **这些术语，就不会感到陌生了。**

另外，在介绍界面设计概念的同时，本章还特地为读者设计了示例程序，希望读者在实践的基础上理解，并设计出满意的 Windows **风格程序界面。**

第五章 格式文件操作和处理

文件是计算机的基本概念，也是计算机进行运算和存储的基础。在这里，我们不对文件处理做太多的原理阐述，本章通过几个示例程序向读者介绍有关 Visual Basic 6.0 中处理文件的基本技术和常用技巧。

在 Visual Basic 6.0 中处理文件可以通过以下几个途径，如通过文件处理控件、调用系统函数和利用 Visual Basic 6.0 自定义文件函数等方式，下面将分别加以介绍。

5.1 文件的格式处理

在 Visual Basic 6.0 中有三个文件类控件——DriveListBox 控件、DirListBox 控件和 FileListBox 控件，它们分别用来处理驱动器、目录和文件的信息，但是在程序运行的过程中，只有将以上三个控件结合起来实用，才能够进行真正的文件处理操作。

下面就以一个程序设计示例来说明在 Visual Basic 6.0 中利用控件如何处理文件，具体的程序设计步骤如下所示：

1. 开始工作

首先激活 Visual Basic 6.0 应用程序，在 Visual Basic 6.0 的集成开发环境中用鼠标选择菜单“文件”/“新建工程”选项，在屏幕上就会弹出一个如图 5-1 所示的“新建工程”对话框。

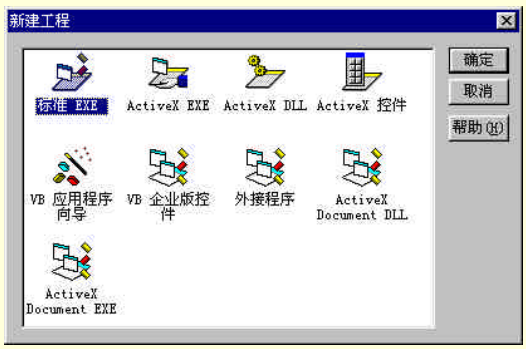


图 5-1 “新建工程”对话框

在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 6.0 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form File
BorderStyle      =     3
Caption     =     "利用控件处理文件"
MaxButton=     0
MinButton=     0
ScaleHeight      =     4590
ScaleWidth       =     6885
Moveable  =     0
StartUpPosition =      2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- 窗体的名字为 File，在程序设计的过程中可以通过 File 这个名字来调用窗体；
- 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；
- 窗体的标题栏中显示文本“利用控件处理文件”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

设置后的程序运行窗体效果如图 5-2 所示。

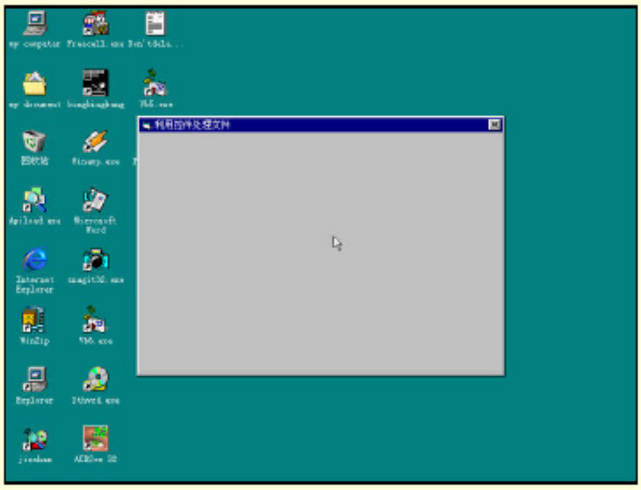


图 5-2 设置属性后的窗体特性

2．添加文件类控件

在本示例程序中，由于要利用对文件进综合处理，所以要向当前空白的窗体上添加一个 DriveListBox 控件、一个 DirListBox 控件和一个 FileListBox 控件。

它们的属性设置如下所示：

```
    Begin Visual Basic.DriveListBox Drive1
Height      =     360
Left  =     360
Top   =     360
Width       =     2895
    End
    Begin Visual Basic.DirListBox Dir1
Height      =     2385
Left  =     360
Top   =     960
Width       =     2895
    End
    Begin Visual Basic.FileListBox File1
Height      =     3015
Left  =     3480
Pattern     =     "*.txt;*.bmp;*.exe;*.hlp"
Top   =     360
Width       =     3015
    End
```

添加文件类控件后的窗体如图 5-3 所示。

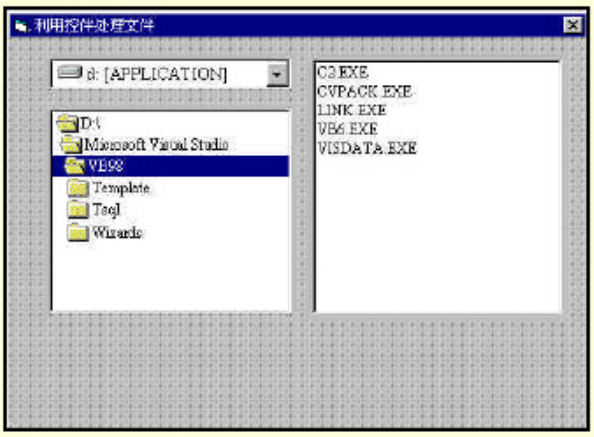


图 5-3 添加文件类控件后的窗体

添加到窗体上的三个文件类控件，它们的作用是用来分别处理驱动器、目录和文件的信息。

注意：

- FileListBox 控件的 Pattern 属性设置为"*.txt;*.bmp;*.exe;*.hlp"，也就意味着，在程序运行的过程中，文件类控件中只能够显示以*.txt;*.bmp;*.exe;*.hlp 为文件后缀的文件。

3. 添加文件信息显示控件

以上所添加的三个文件类控件在程序运行的过程中，只能够完成显示驱动器、目录和文件名称等简单的信息，为了能够完整的显示选中文件和选中目录的信息，在本示例程序中还要向窗体添加两个 TextBox 控件和两个 Label 控件。

添加了文件信息显示控件后的窗体如图 5-4 所示。

图 5-4 添加了文件信息显示控件后的窗体

所添加的文件信息显示控件的作用如下所示：

- Label1 控件：显示固定的文本“目录路径”；
- Text1 控件：显示当前选中目录的路径；
- Label2 控件：显示固定的文本“文件路径”；
- Text2 控件：显示当前选中文件的路径。

其中添加到窗体上文件信息显示控件的属性设置如下所示：

```
Begin Visual Basic.TextBox Text2
Height      =      615
Left  =      3480
MultiLine =      -1
ScrollBars =      1
Top   =      3840
```

```
Width      =     2895
    End
    Begin Visual Basic.TextBox Text1
Height     =     615
Left  =     360
MultiLine  =     -1
ScrollBars =     1
Top   =     3840
Width      =     2895
    End
    Begin Visual Basic.Label Label2
AutoSize   =     -1
Caption    =     "文件路径："
Height     =     285
Left  =     3480
Top   =     3480
Width      =     1200
    End
    Begin Visual Basic.Label Label1
AutoSize   =     -1
Caption    =     "目录路径："
Height     =     285
Left  =     360
Top   =     3480
Width      =     1200
    End
```

经过以上属性设置后的信息显示控件具有如下所示的特性：

- Text1 **控件：在文本框** Text1 **控件中可以接收用户的多行文本输入，如果用户输入信息大于文本的显示区域，就会自动的显示滚动条；**
- Label1 **控件：在程序运行过程中，可以根据控件中的文本自动的调节大小；**
- Text2 **控件：特性与** Text1 **控件相同；**
- Label2 **控件：特性与** Label1 **控件相同。**

4. **程序初始化**

注意：

✍ **在本示例程序中，所谓程序的初始化，指的就是为窗体事件** Private Sub Form_Load()**添加响应代码。**

在程序设计的过程中，首先在项目管理器中用鼠标单击窗体 File**，激活窗体后，用鼠标双击窗体上的空白处，在屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择窗体** Form**，在对应的事件列表中选择窗体的初始化事件** Private Sub Form_Load()**，在代码编辑窗口把光标移动到** Private Sub Form_Load()**事件的处理过程中，并且添加如下所示的程序初始化代码：**

```
Private Sub Form_Load()
Text1.Text = ""
Text2.Text = ""
'清空文本框
Drive1.Drive = "c:"
'设置缺省驱动器
Dir1.Path = "c:\windows"
```

'设置缺省工作路径

End Sub

窗体 Private Sub Form_Load()事件中的代码在对话框窗体被激活的初期就被执行，程序首先会通过语句 Text1.Text = ""、Text2.Text = ""来清空窗体上用于显示文件信息的两个文本框，然后设置程序的缺省工作驱动器和缺省工作路径为“ c: ”和“ c:\windows ”，经过程序初始化后的运行窗体如图 5-5 所示。

图 5-5 程序初始化后的运行窗体

5. 协调文件类控件

前面虽然添加了三个文件类控件，但是到目前为止，在程序运行的过程中它们只能够完成分别显示驱动器、目录和文件名称的工作，为了能够使以上三个控件协调的工作，在本示例程序中还要另外添加代码。

在程序设计的过程中，用鼠标左键双击窗体上的文件类控件，屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中分别选择控件 Dir1 和 Drive1，在对应的事件列表中选择 Private Sub Dir1_Change()事件和 Private Sub Drive1_Change()事件，并且添加如下所示的响应代码：

```
Private Sub Dir1_Change()
File1.Path = Dir1.Path
'设置文件显示路径
Text1.Text = Dir1.Path
'显示当前路径
Text2.Text = ""
'清空文本框
End Sub

Private Sub Drive1_Change()
    On Error GoTo DriveErrs
    '错误处理
    Dir1.Path = Drive1.Drive
    '设置目录显示路径
……
Text2.Text = ""
'清空文本框
End Sub
```

这样在程序运行的过程中，当用户在驱动器列表框中改变驱动器时，就会激活控件的 Private Sub Drive1_Change()事件，程序通过语句 Dir1.Path = Drive1.Drive 设置了目录显示的驱动器，当用户在目录列表框中改变目录时，就会激活控件的 Private Sub Dir1_Change()事件，程序通过 File1.Path = Dir1.Path 设置了文件的显示路径，并且通过代码 Text1.Text = Dir1.Path 显示系统当前的路径。

6. 打开文件的操作

在本示例程序中，可以显示以*.txt;*.bmp;*.exe;*.hlp 为文件后缀的文件，而且还可以对这四种类型的文件进行处理。

技巧：

✍ 为了编程的方便，在程序中可以通过分别调用“记事本”、“画笔”、“帮助”等应用程序来执行以*.txt;*.bmp;*.exe;*.hlp 为文件后缀的文件。

为此，在程序设计的过程中，用鼠标左键双击窗体上的 File1 控件，屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择控件 File1，在对应的事件列表中选择 Private Sub File1_DblClick()事件，并且添加如下所示的响应代码：

```
Private Sub File1_DblClick()
    temp = LCase(Right$(File1.FileName, 3))
    '获取被单击的文件名的最后 3 个字母。
    If Mid(File1.Path, Len(File1.Path)) = "\" Then
       dbclickfile = File1.Path & File1.FileName
       '如果被单击的文件在根目录，就追加文件名。
    Else
       dbclickfile = File1.Path & "\" & File1.FileName
       '如果被单击的文件不在根目录，就追加 "\" 和文件名。
    End If
    Select Case UCase$(Trim$(temp))
    Case "TXT"
         X = Shell("Notepad " + dbclickfile, 1)
         '打开文本文件
    Case "EXE"
         X = Shell(dbclick, 1)
         '打开可执行文件
         ……
    End Select
End Sub
```

在程序运行的过程中，当用户在 File1 上双击鼠标左键时，就会激活控件的 Private Sub File1_DblClick()事件，程序首先通过语句 temp = LCase(Right$(File1.FileName, 3))获取被单击的文件名的最后 3 个字母，然后通过一个条件判断语句来为当前选中的文件添加路径，最后通过选择语句来判断选中文件的后缀，执行相应的应用程序。

例如，用户在 File1 控件的以*.bmp 为结尾的图像文件上双击鼠标左键时，就会激活控件的 Private Sub File1_DblClick()事件，程序通过判断就可以通过 X = Shell("Pbrush " + dbclickfile，1)语句调用“画笔”应用程序来打开指定的图像文件，如图 5-6 所示。

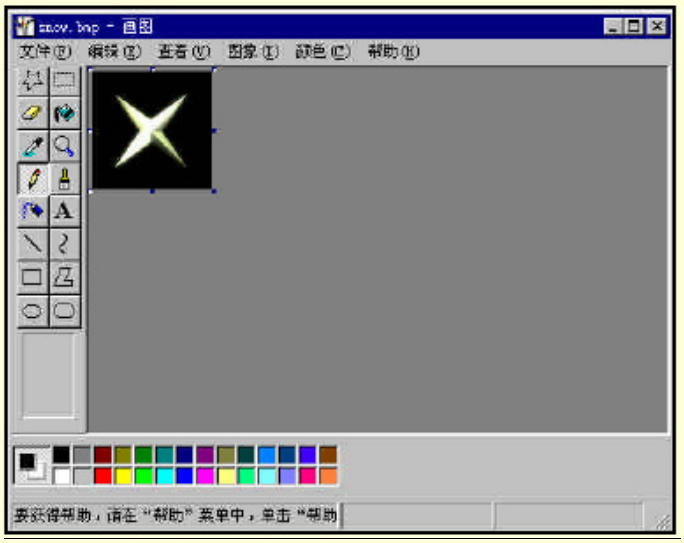


图 5-6 调用“画笔”应用程序来打开指定的图像文件

又如，用户在 File1 控件的以*.hlp 为结尾的图像文件上双击鼠标左键时，就会激活控件的 Private Sub File1_DblClick()事件，程序通过判断就可以通过 X = Shell("WinHelp " + dbclick，1)语句调用“帮助”应用程序来打开指定的帮助文件。如图 5-7 所示。

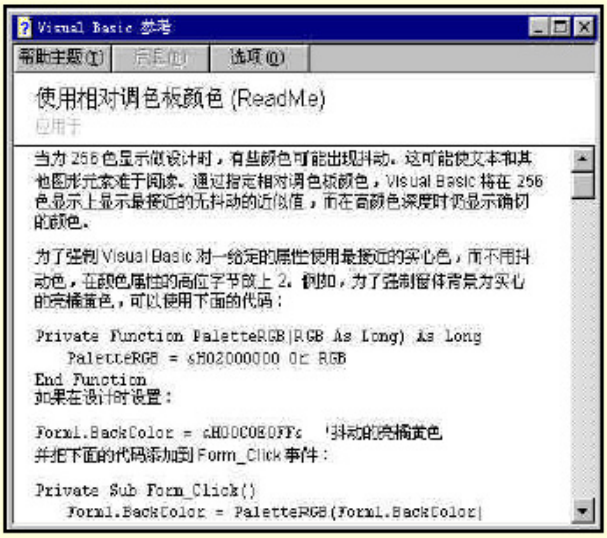


图 5-7 调用“帮助”应用程序来打开指定的帮助文件

7. 运行程序

按照程序清单所示添加剩余的程序代码，用鼠标选择菜单“文件”中的“保存”来存储文件，然后在键盘上按下功能键 F5 运行程序，程序运行初始画面如图 5-8 所示。

图 5-8 程序运行初始画面

在程序运行的过程中，用户可以随时通过用鼠标双击 FileListBox **控件中的文件来执行，例如，用鼠标左键双击一个文本文件，结果如图** 5-9 **所示。**

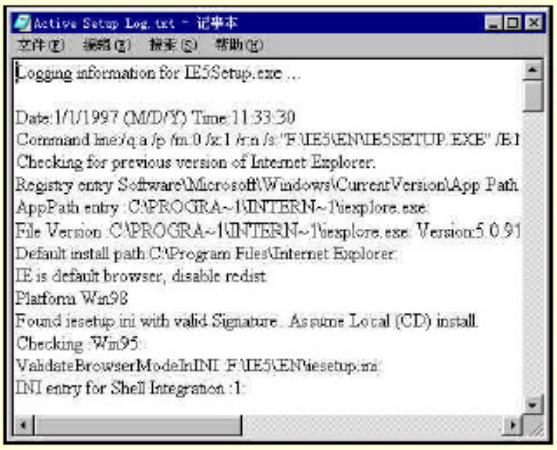


图 5-9 **程序运行结果**

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Begin Visual Basic.Form File
BorderStyle      =      3
Caption     =      "利用控件处理文件"
MaxButton=      0
MinButton =      0
Moveable  =      0
StartUpPosition  =      2
End
Attribute Visual Basic_Name = "File"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False

Private Sub Dir1_Change()
File1.Path = Dir1.Path
'设置文件显示路径
Text1.Text = Dir1.Path
'显示当前路径
Text2.Text = ""
'清空文本框
End Sub

Private Sub Drive1_Change()
    On Error GoTo DriveErrs
    '错误处理
    Dir1.Path = Drive1.Drive
```

```
    '设置目录显示路径
Exit Sub
DriveErrs:
'错误处理
    Select Case Err
        Case 68
            MsgBox prompt:="驱动器未准备好。请在驱动器内插入磁盘。", _
            buttons:=vbExclamation
            '将路径重置为先前使用的驱动器。
            Drive1.Drive = Dir1.Path
            Exit Sub
        Case Else
            MsgBox prompt:="应用程序错误。", buttons:=vbExclamation
    End Select
Text2.Text = ""
'清空文本框
End Sub

Private Sub File1_Click()
If Mid(File1.Path, Len(File1.Path)) = "\" Then
    Text2.Text = File1.Path & File1.FileName
    '如果被单击的文件在根目录，就追加文件名。
Else
    Text2.Text = File1.Path & "\" & File1.FileName
    '如果被单击的文件不在根目录，就追加 "\" 和文件名。
End If
End Sub

Private Sub File1_DblClick()
    temp = LCase(Right$(File1.FileName, 3))
    '获取被单击的文件名的最后 3 个字母。
    If Mid(File1.Path, Len(File1.Path)) = "\" Then
       dbclickfile = File1.Path & File1.FileName
       '如果被单击的文件在根目录，就追加文件名。
    Else
       dbclickfile = File1.Path & "\" & File1.FileName
       '如果被单击的文件不在根目录，就追加 "\" 和文件名。
    End If
    Select Case UCase$(Trim$(temp))
    Case "TXT"
        X = Shell("Notepad " + dbclickfile, 1)
        '打开文本文件
    Case "BMP"
        X = Shell("Pbrush " + dbclickfile, 1)
        '打开图像文件
    Case "EXE"
```

```
            X = Shell(dbclick, 1)
            '打开可执行文件
        Case "HLP"
            X = Shell("WinHelp " + dbclick, 1)
            '打开帮助文件
        Case Else
            nl = Chr$(10) + Chr$(13)
            msg = "试用下面的文件类型之一："
            msg = vbCrLf & msg & vbCrLf & vbCrLf & "        .txt, .bmp, .exe, .hlp"
            MsgBox msg
            '处理其他类型文件
        End Select
    End Sub
    Private Sub Form_Load()
    Text1.Text = ""
    Text2.Text = ""
    '清空文本框
    Drive1.Drive = "c:"
    '设置缺省驱动器
    Dir1.Path = "c:\windows"
    '设置缺省工作路径
    End Sub
```

5.2 创建查询文件程序

在 Windows 操作系统中包括一个文件查询程序，读者可以通过单击 Windows“开始”菜单中的“查找”/“文件或文件夹”选项来激活这个应用程序，运行结果如图 5-10 所示。

图 5-10 查询文件应用程序

提示：

在 Visual Basic 6.0 中，我们也可以制作一个类似的文件查找程序，它是文件处理的关键技术之一。

下面就通过一个示例程序向读者说明，在 Visual Basic 6.0 中如何通过文件类控件和函数来制作一个文件查找程序，它可以在由用户指定的路径下查找指定的文件，其中还支持通配符，具体的程序设计步骤如下所示：

1．开始工作

首先激活 Visual Basic 6.0 应用程序，在 Visual Basic 6.0 的集成开发环境中用鼠标选择菜单“文件”中的“新建工程”选项，在屏幕上就会弹出一个如图 5-11 所示的“新建工程”对话框。

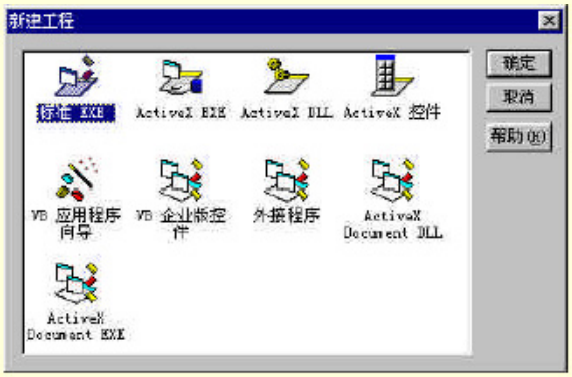


图 5-11 “新建工程”对话框

在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 6.0 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form WinSeek
BorderStyle      =     3
Caption     =     "文件查找应用程序"
Moveable  =     0
MaxButton=     0
MinButton =     0
ScaleHeight      =     4620
ScaleWidth=     6915
StartUpPosition  =     2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；
- 窗体的标题栏中显示文本“文件查找应用程”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

2．添加文件类控件

在本示例程序中，为了实现查找文件的功能，首先要能够显示文件，所以在程序设计的过程中要向当前空白的窗体上添加一个 DriveListBox 控件、一个 DirListBox 控件和一个 FileListBox 控件。

它们的作用如下所示：

- DriveListBox 控件：在程序运行的过程中显示用户计算机上的所有有效的驱动器，并且负责用户选择驱动器时的错误处理；
- DirListBox 控件：在程序运行的过程中显示当前驱动器下的各个目录项；
- FileListBox 控件：显示当前路径下的文件列表，文件显示的类型需要由用户在程序运行的过程中指定。

添加控件后的窗体如图 5-12 所示。

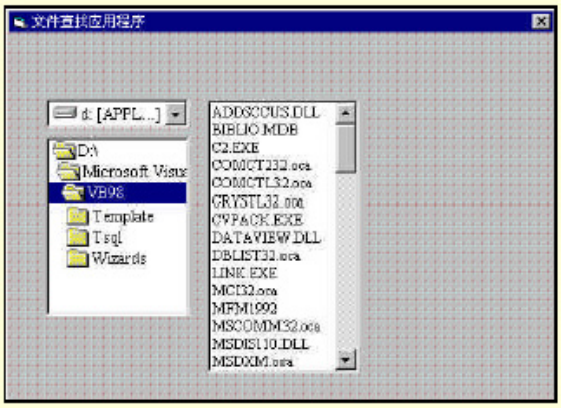


图 5-12　添加文件类控件后的窗体

添加到窗体上的三个文件类控件的属性设置如下所示：

```
    Begin Visual Basic.FileListBox filList
Archive     =     0
Height      =     3465
Left  =     2520
Hidden      =     0
System      =     0
ReadOnly    =     0
Top   =     840
Width       =     1935
    End
    Begin Visual Basic.DirListBox dirList
Height      =     2250
Left  =     480
Top   =     1320
Width       =     1815
    End
    Begin Visual Basic.DriveListBox drvList
Height      =     360
Left  =     480
Top   =     840
Width       =     1815
    End
```

提示：

✍ **经过以上属性设置的文件类控件 filList 在程序运行的初期不能够显示系统文件、只读文件、隐藏文件和存档文件。**

3．添加其他控件

为了在程序的过程中能够实现查找文件和控制文件显示类型的作用，在程序设计的过程中要向窗体上 11 个 ActiveX 控件，过程如下所示：

首先向窗体上添加两个 CommandButton 控件，这两个 ActiveX 控件的作用是——完成查找文件的具体动作、结束程序的运行，它们的属性设置如下所示：

```
    Begin Visual Basic.CommandButton cmdSearch
Caption     =     "查找"
Height      =     480
Left  =     480
```

```
Top  =    3840
Width     =    1800
   End
   Begin Visual Basic.CommandButton cmdExit
Caption   =    "退出"
Left  =    4800
Top  =    3840
   End
```

添加两个按钮控件后，接着在窗体上放置一个 Label 控件和一个 TextBox 控件，它们的作用如下所示：

- Label 控件：显示固定的提示文本“查找条件”；
- TextBox 控件：在其中用户可以输入查找文件的查询条件，在其中还支持统配符“*”和“？”。

其中 Label 控件和 TextBox 控件的属性设置如下所示：

```
   Begin Visual Basic.TextBox txtSearchSpec
Height    =    285
Left  =    2520
Top  =    360
Width     =    1935
   End
   Begin Visual Basic.Label lblCriteria
Caption   =    "查找条件："
Height    =    255
Left  =    480
   End
```

设置完 Label 控件和 TextBox 控件的属性后，接着向窗体上放置一个 Frame 控件和四个 CheckBox 控件，添加控件后的窗体如图 5-13 所示。

添加到窗体上的 Frame 控件充当四个 CheckBox 控件的容器，而四个 CheckBox 控件的作用如下所示：

- Check1（0）控件：设置在程序运行的过程中，文件类控件中是否显示系统文件；
- Check1（1）控件：设置在程序运行的过程中，文件类控件中是否显示隐藏文件；
- Check1（2）控件：设置在程序运行的过程中，文件类控件中是否显示只读文件；
- Check1（3）控件：设置在程序运行的过程中，文件类控件中是否显示存档文件。

图 5-13 添加 ActiveX 控件后的窗体

Frame 控件和四个 CheckBox 控件的属性设置如下所示：

```
   Begin Visual Basic.Frame Frame1
      Caption        =    "显示类型："
      Begin Visual Basic.CheckBox Check1
         Caption     =    "存档"
```

```
        Index          =     3
    End
    Begin Visual Basic.CheckBox Check1
        Caption        =     "只读"
        Index          =     2
    End
    Begin Visual Basic.CheckBox Check1
        Caption        =     "隐藏"
        Index          =     1
    End
    Begin Visual Basic.CheckBox Check1
        Caption        =     "系统"
        Index          =     0
    End
End
```

4. 程序初始化

在程序设计的过程中，首先在项目管理器中用鼠标单击窗体 File，**激活窗体后，用鼠标双击窗体上的空白处，在屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择窗体** Form，**在对应的事件列表中选择窗体的初始化事件** Private Sub Form_Load()，**在代码编辑窗口把光标移动到** Private Sub Form_Load()**事件的处理过程中，并且添加如下所示的程序初始化代码：**

```
Private Sub Form_Load()
drvList.Drive = "c:"
'设置缺省驱动器
dirList.Path = "c:\windows"
'设置缺省工作路径
txtSearchSpec.Text = "*.*"
'设置文本框中显示文本
filList.Pattern = "*.*"
'设置过滤器
Check1(0).Value = 0
……
'初始化 Check1 控件数组状态
End Sub
```

程序说明：

窗体 Private Sub Form_Load()**事件中的代码在对话框窗体被激活的初期就被执行，程序首先会通过语句设置缺省工作驱动器和缺省工作路径为“** c: **”和“** c:\windows **”，然后设置文件类控件的过滤器为“** *.* **”，即在程序运行的初期将显示所有类型的文件，最后通过** Check1(0).Value = 0、Check1(1).Value = 0、Check1(2).Value = 0 **和** Check1(3).Value = 0 **来初始化** Check1 **控件数组状态。**

经过程序初始化后的运行窗体如图 5-14 **所示。**

图 5-14 经过程序初始化后的运行窗体

5. 查找指定类型的文件

示例程序在运行的过程中，可以在用户指定的目录下查找指定类型的文件，其中还支持通配符操作。

在程序设计的过程中，在代码窗口的对象列表中选择控件 cmdSearch，在对应的事件列表中选择 Private Sub cmdSearch_Click()事件，把鼠标移动到事件处理过程中，并且添加如下所示的响应代码：

```
Private Sub cmdSearch_Click()
filList.Pattern = txtSearchSpec.Text
filList.Path = dirList.Path
End Sub
```

在程序运行的过程中，如果用户单击窗体上的 cmdSearch 按钮控件，就会激活控件的 Private Sub cmdSearch_Click()事件，程序首先通过语句 filList.Pattern = txtSearchSpec.Text 把当前文本框中的文本内容设置为文件查找的过滤器，然后通过语句 filList.Path = dirList.Path 来显示文件查找结果。如图 5-15 所示即为在 c:\windows 目录下查找所有的以字母 a 开头，并且以*.ini 为后缀文件的查询结果。

图 5-15 在 c:\windows 目录下查询指定文件

6. 更改显示文件的类型

提示：

- 示例程序在运行的过程中，可以在用户指定的目录下显示指定属性的文件，如系统文件、只读文件、隐藏文件和存档文件等。

为此，在程序设计的过程中，用鼠标左键双击窗体上的 Check1 控件数组，屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择控件数组 Check1，在对应的事件列表中选择 Private Sub Check1_Click(Index As Integer)事件，把鼠标移动到事件处理过程中，并且添加如下所示的响应代码：

```
Private Sub Check1_Click(Index As Integer)
```

```
Select Case Index
        Case 0
            filList.System = Not filList.System
        Case 1
            filList.Hidden = Not filList.Hidden
        Case 2
            filList.ReadOnly = Not filList.ReadOnly
        Case 3
            filList.Archive = Not filList.Archive
End Select
End Sub
```

在程序运行的过程中，当用户改变控件数组 Check1 **中的选项时，就会激活控件数组的** Private Sub Check1_Click(Index As Integer)**事件，程序通过一个选择语句判断用户的选择，然后执行相应的分支语句来决定文件类控件中显示文件的类型。**

例如，当用户选中“系统”选项，程序就会通过 Private Sub Check1_Click(Index As Integer)**事件中的** filList.System = Not filList.System **语句在当前的文件类控件中显示系统文件，其他几种显示类型的转换过程与系统文件类似，在这里就不多加赘述了。**

7．运行程序

按照程序清单所示添加剩余的程序代码，用鼠标选择菜单“文件”中的“保存”来存储文件，然后在键盘上按下功能键 F5 **运行程序，程序运行画面如图** 5-16 **所示。**

图 5-16 **程序运行画面**

在程序运行的过程中，用户可以通过文件类控件来显示当前用户计算机上的所有有效的驱动器、目录和文件，并且可以选择显示文件的类型（系统、隐藏、只读和存档等），还可以在文本框中输入要查找的文件名来实现查找的功能。

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Begin Visual Basic.Form WinSeek
BorderStyle     =     3
Caption     =     "文件查找应用程序"
ClientHeight     =     4620
ClientLeft  =     1920
ClientTop  =     1890
```

```
ClientWidth      =      6915
Moveable  =      0
MaxButton=       0
MinButton=       0
ScaleHeight      =      4620
ScaleWidth=      6915
StartUpPosition  =      2
End

Attribute Visual Basic_Name = "WinSeek"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False

Private Sub Check1_Click(Index As Integer)
Select Case Index
        Case 0
                filList.System = Not filList.System
                '是否显示系统文件
        Case 1
                filList.Hidden = Not filList.Hidden
                '是否显示隐藏文件
        Case 2
                filList.ReadOnly = Not filList.ReadOnly
                '是否显示只读文件
        Case 3
                filList.Archive = Not filList.Archive
                '是否显示存档文件
End Select
End Sub

Private Sub cmdExit_Click()
End
'结束程序运行
End Sub
Private Sub cmdSearch_Click()
filList.Pattern = txtSearchSpec.Text
'设置过滤器
filList.Path = dirList.Path
'显示查找结果
End Sub

Private Sub DirList_Change()
filList.Path = dirList.Path
'设置文件显示路径
```

```
End Sub

Private Sub DrvList_Change()
On Error GoTo DriveErrs
dirList.Path = drvList.Drive
'设置目录显示路径
Exit Sub
DriveErrs:
    Select Case Err
        Case 68
            MsgBox prompt:="驱动器未准备好。请在驱动器内插入磁盘。", _
            buttons:=vbExclamation
            ' 将路径重置为先前使用的驱动器。
            drvList.Drive = dirList.Path
            Exit Sub
        Case Else
            MsgBox prompt:="应用程序错误。", buttons:=vbExclamation
    End Select
End Sub

Private Sub Form_Load()
drvList.Drive = "c:"
'设置缺省驱动器
dirList.Path = "c:\windows"
'设置缺省工作路径
txtSearchSpec.Text = "*.*"
'设置文本框中显示文本
filList.Pattern = "*.*"
'设置过滤器
Check1(0).Value = 0
Check1(1).Value = 0
Check1(2).Value = 0
Check1(3).Value = 0
'初始化 Check1 控件数组状态
End Sub

Private Sub Form_Unload(Cancel As Integer)
End
'结束程序的运行
End Sub
```

5.3 利用文件函数

前面的两个示例程序大多是依靠着三个文件类控件来完成对文件的操作，其实在 Visual Basic 6.0 **系统中有很多的处理文件的系统语句和函数（以下统称文件函数），利用这些文件函数可以实现文件的复制、粘贴和删除**

等操作，下面就利用 Kill、GetAttr 和 FileDateTime 等文件函数来进行文件的处理操作。

注意：

> 示例程序可以显示选中的大小、修改时间和属性等信息，同时还可以通过简单的鼠标拖动就能够实现删除文件等操作。

具体的程序设计步骤如下所示：

1．开始工作

新建标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form Form1
BorderStyle=3
Caption="Form1"
ScaleHeight=4590
ScaleWidth =6885
ShowInTaskbar=0
StartUpPosition=2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；
- 窗体的标题栏中显示文本“ Form1 ”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

2．添加文件类控件

在本示例程序中，虽然主要的程序功能是通过调用 Visual Basic 6.0 的系统文件函数来完成的，但是首先要能够显示文件，所以在程序设计的过程中要向当前空白的窗体上添加一个 DriveListBox 控件、一个 DirListBox 控件、一个 FileListBox 控件和一个 ComboBox 控件，添加控件后的窗体如图 5-17 所示。

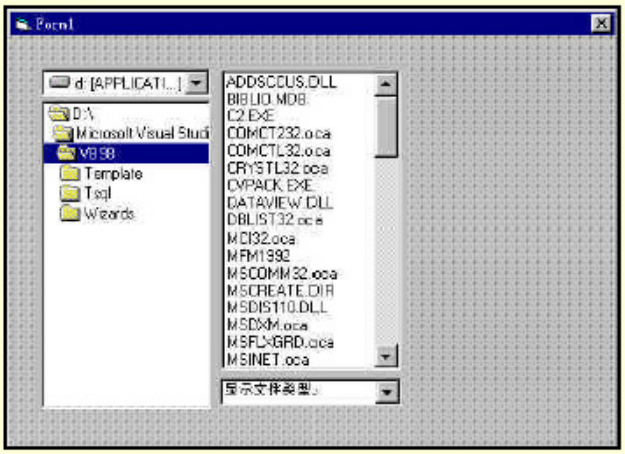


图 5-17　添加文件类控件后的窗体

其中四个控件的作用如下所示：

- DriveListBox 控件：在程序运行的过程中显示用户计算机上的所有有效的驱动器，并且负责用户选择驱动器时的错误处理；
- DirListBox 控件：在程序运行的过程中显示当前驱动器下的各个目录项；
- FileListBox 控件：显示当前路径下的文件列表，文件显示的类型需要由用户在程序运行的过程中指定；
- ComboBox 控件：用于在程序运行的过程中选择文件类控件的过滤器。

以上添加的四个 ActiveX 控件属性设置如下所示：

```
    Begin Visual Basic.ComboBox Combo1
Height     =     315
```

```
Left   =     2400
Text   =     "显示文件类型："
Top    =     3840
Width      =     2055
   End
   Begin Visual Basic.FileListBox File1
Height     =     3405
Archive    =     -1
Hidden     =     -1
Left   =     2400
ReadOnly   =     -1
System     =     -1
Top    =     360
Width      =     2055
   End
   Begin Visual Basic.DirListBox Dir1
DragIcon   =     "form2.frx":0054
Height     =     3465
Left   =     360
Top    =     720
Width      =     1935
   End
   Begin Visual Basic.DriveListBox Drive1
DragIcon   =     "form2.frx":035E
Height     =     315
Left   =     360
Top    =     360
Width      =     1935
   End
```

3. 添加其他 ActiveX 控件

在程序运行的过程中，还要显示选中文件的大小、修改时间和各种属性，这些都需要通过一定的显示容器才能够实现。

为此，在程序设计的过程中，向窗体上添加三个 Label **控件和三个** TextBox **控件，各个控件的功能将会在以后加以说明，添加控件后的窗体如图** 5-18 **所示。**

图 5-18　**添加控件后的窗体**

其中 ActiveX 控件的属性设置如下所示：

```
    Begin Visual Basic.TextBox Text3
Height      =      675
Left  =      4680
Top  =      3480
Width       =      2055
    End
    Begin Visual Basic.TextBox Text2
Top  =      2160
    End
    Begin Visual Basic.TextBox Text1
Top  =      840
    End
    Begin Visual Basic.Label Label3
AutoSize    =      -1
Caption     =      "修改时间："
Height      =      315
Left  =      4680
Top  =      3000
Width       =      1425
    End
    Begin Visual Basic.Label Label2
            Caption         =      "文件属性："
    End
    Begin Visual Basic.Label Label1
            Caption         =      "文件大小："
    End
```

它们的作用如下所示：

- Text1 控件：在程序运行的过程中显示选中文件的大小，以 Byte 为单位；
- Text2 控件：在程序运行的过程中显示选中文件的属性，如系统、存档和只读等；
- Text3 控件：在程序运行的过程中显示选中文件的最近修改时间和日期；
- Label 控件：显示固定的提示文本“文件大小”、“文件属性”和“修改时间”。

4. 程序初始化

在程序设计的过程中，首先在项目管理器中用鼠标单击窗体 File，激活窗体后，用鼠标双击窗体上的空白处，在屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择窗体 Form，在对应的事件列表中选择窗体的初始化事件 Private Sub Form_Load()，在代码编辑窗口把光标移动到 Private Sub Form_Load()事件的处理过程中，并且添加如下所示的程序初始化代码：

```
Private Sub Form_Load()
Drive1.Drive = "c:"
Dir1.Path = "c:\windows"
File1.Pattern = "*.*"
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
End Sub
```

窗体 Private Sub Form_Load()事件中的代码在对话框窗体被激活的初期就被执行，程序首先会通过语句设置

缺省工作驱动器和缺省工作路径为“c:”和“c:\windows”，然后设置文件类控件的过滤器为“*.*”，即在程序运行的初期将显示所有类型的文件，最后通过 Text1.Text = ""、Text2.Text = ""和 Text3.Text = ""清空窗体上的三个文本框。经过程序初始化后的运行窗体如图 5-19 所示。

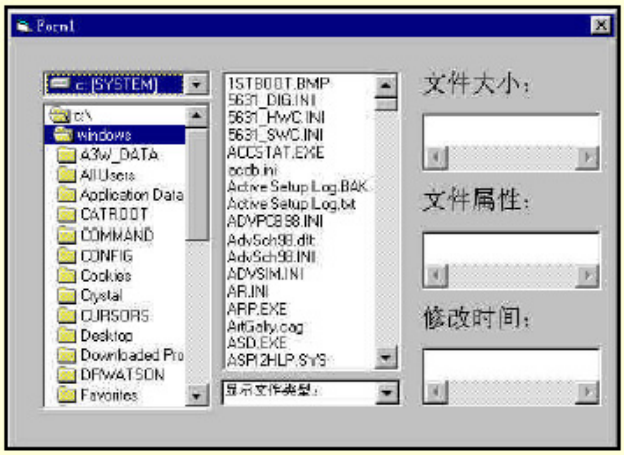


图 5-19　经过初始化后的程序运行窗体

5. 显示文件信息

示例程序在运行的过程中，用户可以在文件显示控件中选择一个有效的文件，同时在三个文本框中就会显示出选中文件的大小、修改时间和属性。为此，在程序设计的过程中，用鼠标左键双击窗体上的 File1 控件，屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择控件 File1，在对应的事件列表中选择 Private Sub File1_Click()事件，把鼠标移动到事件处理过程中，并且添加如下所示的响应代码：

```
Private Sub File1_Click()
Form1.Text2.Text = ""
If Mid(File1.Path, Len(File1.Path)) = "\" Then
    Fname = File1.Path & File1.FileName
    '如果被单击的文件在根目录，就追加文件名。
Else
    Fname = File1.Path & "\" & File1.FileName
    '如果被单击的文件不在根目录，就追加 "\" 和文件名。
End If
Form1.Text1.Text = FileLen(Fname) & "Byte"
If (GetAttr(Fname) And 32) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 存档"
End If
If (GetAttr(Fname) And 2) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 隐藏"
End If
If (GetAttr(Fname) And 1) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 只读"
End If
If (GetAttr(Fname) And 4) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 系统"
End If
Form1.Text3.Text = FileDateTime(Fname)
End Sub
```

程序说明：

在程序运行的过程中，当用户在文件显示控件 File1 上选中一个有效的文件后，程序就会自动的激活控件的 Private Sub File1_Click()事件：

首先通过语句 Form1.Text2.Text = ""清空文本框控件 Text2 中的文本，通过语句 Form1.Text1.Text = FileLen(Fname) & "Byte"显示用户选中文件的大小，然后通过四个条件判断语句来显示选中文件的属性（如存档、隐藏、只读和系统等），最后通过代码 Form1.Text3.Text = FileDateTime(Fname)显示选中文件的修改时间。

显示选中文件基本信息后的窗体如图 5-20 所示。

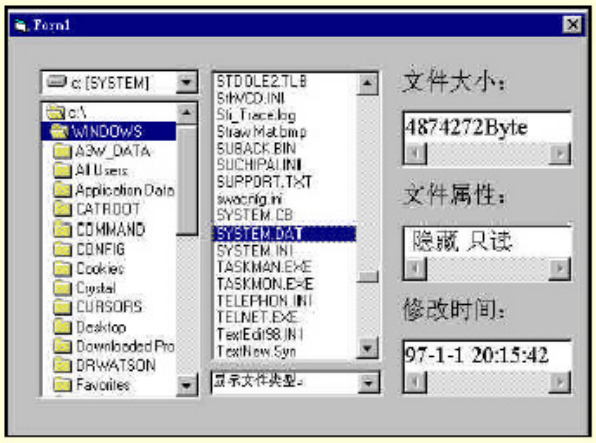


图 5-20 显示选中文件基本信息后的窗体

6. 拖动文件

注意：

- 在程序运行的过程中，当用户在文件控件 File1 中按下鼠标左键时，鼠标的形状就会改变，当把选中文件拖动到目录显示控件 Dir1 中时，程序就会自动的删除指定的文件。

为此，在程序设计的过程中，在窗体上双击左键，在屏幕上就会弹出一个代码窗口，在代码窗口对象列表中分别选择控件 File1 和 Dir1，在对应的事件列表中选择 Private Sub File1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)事件和 Private Sub Dir1_DragDrop(Source As Control, X As Single, Y As Single)事件，把光标移动到事件处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub File1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
File1.DragIcon = Drive1.DragIcon
File1.Drag
'开始拖动
End Sub
Private Sub Dir1_DragDrop(Source As Control, X As Single, Y As Single)
……
Kill Fname
'删除拖动文件
Form1.File1.Refresh
End Sub
```

在程序运行的过程中，当用户在文件显示控件 File1 上按下鼠标左键时，就会激活控件的 Private Sub File1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)事件，程序通过语句 File1.Drag 开始了选中文件的拖动操作，当把选中文件拖动到目录显示控件 Dir1 中时，就会激活控件的 Private Sub Dir1_DragDrop(Source As Control, X As Single, Y As Single)事件，通过语句 Kill Fname 删除指定的文件，最后通过代码 Form1.File1.Refresh 刷新删除文件后的文件显示控件。拖动文件时的窗体如图 5-21 所示。

图 5-21 拖动文件时的窗体

7. 运行程序

按照程序清单所示添加剩余的程序代码，用鼠标选择菜单“文件”中的“保存”来存储文件，然后在键盘上按下功能键 F5 运行程序，程序运行初始画面如图 5-22 所示。

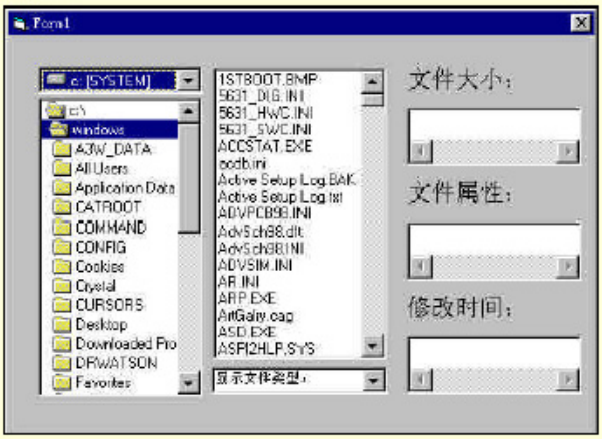


图 5-22 程序运行初始画面

在程序运行的过程中，用户可以选择计算机上的任何一个有效的驱动器和其中的目录，同时还可以显示选中文件的大小和修改事件等信息，还可以对选中文件进行拖动操作，程序运行结果如图 5-23 所示。

图 5-23 程序运行结果

提示：

- 在程序运行的过程中，用户还可以通过选择下拉式列表框中的内容来决定在文件控件中显示的类型，如所有文件、以*.txt 为文件后缀的文本文件、以*.bmp 为文件后缀的图像文件等。

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Begin Visual Basic.Form Form1
BorderStyle       =      3
Caption     =      "Form1"
MaxButton=      0
MinButton =      0
Moveable  =      0
ScaleHeight       =      4590
ScaleWidth=      6885
StartUpPosition  =      2
End
Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False

Private Sub Combo1_Click()
Form1.File1.Pattern = Form1.Combo1.Text
End Sub

Private Sub Dir1_Change()
Form1.File1.Path = Form1.Dir1.Path
End Sub

Private Sub Dir1_Click()
Fname = Form1.Dir1.Path
Result = GetAttr(Fname) And vbDirectory
If Result <> 0 Then
    Form1.Text2.Text = "目录"
End If
Form1.Text1.Text = ""
Form1.Text3.Text = ""
End Sub

Private Sub Dir1_DragDrop(Source As Control, X As Single, Y As Single)
If Mid(File1.Path, Len(File1.Path)) = "\" Then
    Fname = File1.Path & File1.FileName
    如果被单击的文件在根目录，就追加文件名
Else
    Fname = File1.Path & "\" & File1.FileName
    如果被单击的文件不在根目录，就追加 "\" 和文件名
End If
```

```
Kill Fname
Form1.File1.Refresh
End Sub

Private Sub Dir1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
Select Case State
    Case 0
        ' 当"源"进入"放"区域内时，显示一个新图标。
        File1.DragIcon = Dir1.DragIcon
    Case 1
        ' 当"源"离开"放"区域时，显示原来的 DragIcon 图标。
        File1.DragIcon = Drive1.DragIcon
    End Select
End Sub

Private Sub Drive1_Change()
    On Error GoTo DriveErrs
    '错误处理
    Dir1.Path = Drive1.Drive
    '设置目录显示路径
Exit Sub
DriveErrs:
'错误处理
    Select Case Err
        Case 68
            MsgBox prompt:="驱动器未准备好。请在驱动器内插入磁盘。", _
            buttons:=vbExclamation
            ' 将路径重置为先前使用的驱动器。
            Drive1.Drive = Dir1.Path
            Exit Sub
        Case Else
            MsgBox prompt:="应用程序错误。", buttons:=vbExclamation
    End Select
End Sub

Private Sub File1_Click()
Form1.Text2.Text = ""
If Mid(File1.Path, Len(File1.Path)) = "\" Then
    Fname = File1.Path & File1.FileName
    '如果被单击的文件在根目录，就追加文件名。
Else
    Fname = File1.Path & "\" & File1.FileName
    '如果被单击的文件不在根目录，就追加 "\" 和文件名。
End If
Form1.Text1.Text = FileLen(Fname) & "Byte"
If (GetAttr(Fname) And 32) <> 0 Then
```

```
    Form1.Text2.Text = Form1.Text2.Text & " 存档"
End If
If (GetAttr(Fname) And 2) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 隐藏"
End If
If (GetAttr(Fname) And 1) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 只读"
End If
If (GetAttr(Fname) And 4) <> 0 Then
    Form1.Text2.Text = Form1.Text2.Text & " 系统"
End If
Form1.Text3.Text = FileDateTime(Fname)
End Sub

Private Sub File1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
File1.DragIcon = Drive1.DragIcon
File1.Drag
End Sub
```

5.4 小 结

文件是计算机的基本概念，也是计算机进行运算和存储的基础，在 Visual Basic 6.0 中处理文件可以通过以下几个途径，如通过文件处理控件、调用系统函数和利用 Visual Basic 6.0 自定义文件函数等方式。

在本章中将会通过几个示例程序向读者介绍有关 Visual Basic 中处理文件的基本技术和常用技巧，如利用文件类控件处理文件、制作文件查询程序和利用文件函数处理文件等。

第六章　多媒体程序开发和应用

在当今计算机领域中，网络与多媒体技术已经成为人们生活的一部分，同时也是计算机领域中发展最为迅速的领域，多媒体编程的常用方法，已经成为程序员的必备技巧之一，所以笔者特地用一章的篇幅向读者介绍多媒体程序设计，其中包括动画程序制作，如何制作动画播放器等。

6.1　多媒体动画

图文并茂是多媒体程序的特点之一，在应用程序中适当的插入一些动画，会使应用程序变得生动、活泼，所以多媒体应用程序设计中动画程序设计是很重要的一部分内容。下面就通过一个示例程序向读者介绍 Visual Basic 6.0 中动画应用程序设计得一般方法和常用技巧，具体的程序设计步骤如下所示。

1．开始工作

首先用鼠标选择 Windows 操作系统“开始”菜单中的“程序”/Microsoft Visual Studio 中的 Microsoft Visual Basic 6.0 选项来激活 Visual Basic 应用程序，在 Visual Basic 6.0 的集成开发环境中用鼠标选择“文件”菜单中的“新建工程”选项，在屏幕上就会弹出一个如图 6-1 所示的“新建工程”对话框。

图 6-1　“新建工程”对话框

在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 6 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form Form1
BorderStyle =   3
Caption =   "播放动画"
Icon    =   "Redtop.frx":0000
MaxButton   =   0
MinButton   =   0
Moveable    =   0
ScaleHeight =   306
ScaleMode   =   3
ScaleWidth  =   459
StartUpPosition =   2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；
- 窗体的标题栏中显示文本“播放动画”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

2. 添加控件

在程序运行的过程中，为了能够显示动画，需要有显示容器，为此要向当前空白的窗体上添加两个 PictureBox 控件，同时向窗体上添加四个按钮控件、一个 Timer 控件、18 个 Image 控件、两个 Label 控件和两个 TextBox 控件，添加控件后的窗体如图 6-2 所示。

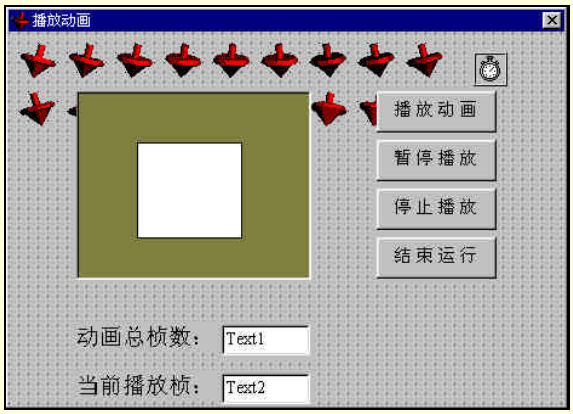


图 6-2 添加控件后的窗体

各个控件的作用如下所示：

- Image 控件：18 个 Image 控件组成一个控件数组，在其中存储了 18 幅图标文件，在程序运行的过程中，窗体的图标将会随着动画进程而改变；
- 两个 Label 控件：显示固定的文本“动画总桢数：”和“当前播放桢”；
- 四个 CommandButton 控件：在程序运行的过程中用于控制动画播放的进程，如播放、暂停和停止等；
- 两个 TextBox 控件：在程序运行的过程中，用于显示当前动画的总桢数和当前播放桢数；
- 两个 PictureBox 控件：在程序运行过程中充当动画显示的容器。

其中添加到窗体上的各个控件的属性设置如下所示：

```
   Begin Visual Basic.TextBox Text2
Height  =   375
Left    =   2640
Text    =   "Text2"
Top =   4200
Width   =   1095
   End
   Begin Visual Basic.TextBox Text1
Text    =   "Text1"
Top =   3600
   End
   Begin Visual Basic.CommandButton Command3
Caption =   "结束运行"
Height  =   510
Left    =   4560
Top =   2520
Width   =   1485
   End
   Begin Visual Basic.CommandButton Command2
```

```
Caption =   "停止播放"
Top =   1920
  End
   Begin Visual Basic.CommandButton Command1
Caption =   "暂停播放"
Top =   1320
  End
   Begin Visual Basic.CommandButton Command4
Caption =   "播放动画"
Top =   720
  End
   Begin Visual Basic.PictureBox Picture6
Height  =   2325
Left    =   840
Top =   720
Width   =   2910
     Begin Visual Basic.PictureBox Picture1
Appearance  =   0
AutoSize    =   -1
Height  =   1185
Left    =   720
Top =   600
Width   =   1305
     End
  End
   Begin Visual Basic.Timer Timer1
Interval    =   10
Left    =   5760
Top =   240
  End
   Begin Visual Basic.Label Label2
       Caption     =  "当前播放桢："
  End
   Begin Visual Basic.Label Label1
       Caption     =  "动画总桢数："
  End
   Begin Visual Basic.Image Image1
        Index       =   17
  End
  ……
   Begin Visual Basic.Image Image1
        Index       =   0
  End
```

经过以上属性设置后的控件具有如下所示的特性；

- Picture1 **控件在程序运行的过程中能够根据图像文件的大小自动调节控件的尺寸来适应图像；**
- 18 **个** Image **控件组成一个控件数组** Image1**；**

- Label1 和 Label2 控件能够根据其中文本的长度自动调节控件的尺寸来适应文本；
- Picture2 控件在程序运行的过程中充当 Picture1 控件显示的容器。

3. 添加 ActiveX 控件

在本示例程序设计的过程中，需要用到两个 ActiveX 控件——Slider 控件和 PictureClip 控件。为此，在程序设计的过程中，用鼠标左键单击菜单“工程”中的“部件”选项，在屏幕上就会弹出一个如图 6-3 所示添加 ActiveX 的对话框。

注意：

ActiveX 控件是第三方程序开发商提供的“即取即用”控件，在 Visual Basic 6.0 的“工具箱”上没有，需要将它们调用出来。

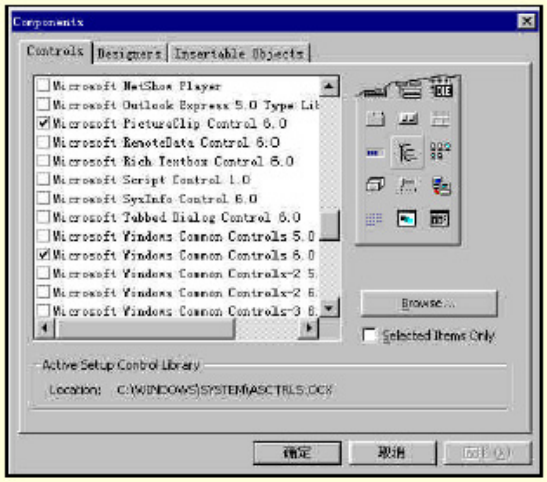


图 6-3 “部件”对话框

在添加 ActiveX 控件的“部件”对话框中选择 Microsoft PictureClip Control 6.0 和 Microsoft Windows Common Controls 6.0 两个选项，单击“确定”按钮，程序就会自动的向当前的项目文件中添加了示例程序所需要的 ActiveX 控件，这两个 ActiveX 控件的声明如下所示：

```
Object = "{27395F88-0C0C-101B-A3C9-08002B2F49FB}#1.1#0"; "PICCLP32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
```

在程序设计的过程中，向当前空白的窗体上添加一个 Slider 控件和一个 PictureClip 控件，它们的属性设置如下所示：

```
   Begin MSComctlLib.Slider Slider1
Height  =   375
Left    =   600
Top =   3120
Width   =   3255
LargeChange =   3
Max =   17
   End
   Begin PicClip.PictureClip PictureClip2
Left    =   960
Top =   0
Rows    =   3
Cols    =   6
   End
```

说明：

其中添加 ActiveX 控件的方法与添加普通控件的方法一样，在“工具箱”上用鼠标双击相应的 ActiveX 控件图标即可。

添加 ActiveX 控件后的窗体如图 6-4 所示。

图 6-4 添加 ActiveX 控件后的窗体

添加到窗体上的两个 ActiveX 控件的作用如下所示：

- Slider 控件：在程序运行的过程中动态的显示动画播放进程，而且用户还可以通过滚动条来调节动画播放进程；
- PictureClip 控件：在控件中存储了 18 幅图像，在程序运行的过程中通过调用这 18 幅图像来显示动画。

其中 PictureClip 控件可以通过如图 6-5 所示的对话框来设置。

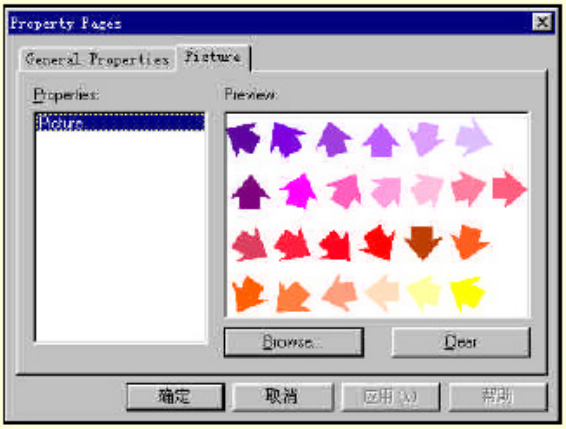


图 6-5 PictureClip 控件属性设置

4. 程序初始化

在本示例程序中，所谓程序初始化，指的就是为窗体 Private Sub Form_Load()事件所添加的响应代码。

在程序设计的过程中，用鼠标左键双击窗体上的空白处，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的事件列表中选择窗体 Form，在对应的事件列表中选择事件 Private Sub Form_Load()，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Form_Load()
Picture1.Picture = PictureClip2.GraphicCell(0)
y = 0
Picture1.Left = (Picture6.ScaleWidth - Picture1.Width) / 2
Picture1.Top = (Picture6.ScaleHeight - Picture1.Height) / 2
Text1.Text = PictureClip2.Cols * PictureClip2.Rows
Text2.Text = 0
End Sub
```

在程序运行的初期，事件 Private Sub Form_Load()中的代码就会被执行，程序通过 Picture1.Picture = PictureClip2.GraphicCell(0)语句来显示第一帧动画，然后通过两条语句来设置图像控件的显示位置，在程序初始化的最后，通过语句 Text1.Text = PictureClip2.Cols * PictureClip2.Rows 和 Text2.Text = 0 完成对文本框控件中文

本的初始化。

经过程序初始化后的运行窗体如图 6-6 所示。

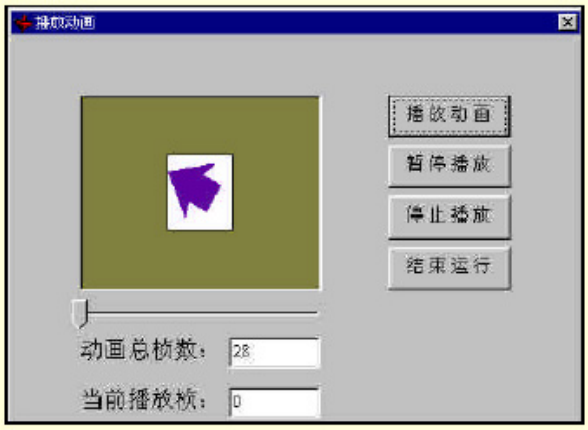


图 6-6 经过程序初始化后的运行窗体

5. 响应计时器事件

窗体上的 Timer 控件在程序运行的过程中充当计时器，即每隔一定的时间就自动的重复某一个动作。在本示例程序中，用于动画的显示要不断的从图像文件中读取信息，所以要利用计时器。

计时器控件的属性设置如图 6-7 所示。

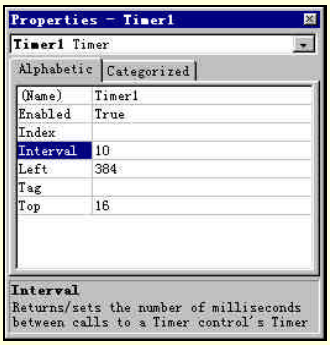


图 6-7 计时器控件的属性设置

Timer1 控件的 Interval 属性设置为 10，即每隔 10 微秒就会自动激活一个 Private Sub Timer1_Timer()事件。

在程序设计的过程中，用鼠标左键双击窗体上的 Timer 控件，在屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择计时器控件 Timer1，在对应的事件列表中选择 Private Sub Timer1_Timer()事件，把光标移动到事件处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Timer1_Timer()
If toggle = 1 Then runtop
'调用函数显示动画
End Sub
```

在程序运行的过程中，每隔 10 微秒，程序就会自动激活一个 Private Sub Timer1_Timer()事件，程序首先判断标志变量 toggle 的值，如果 toggle=1，那么就调用 Private Sub runtop()事件来显示动画。

Private Sub runtop()事件是自定义的事件，它的代码结构如下所示：

```
Private Sub runtop()
y = y + 1: If y = 18 Then y = 0
'Picture1.Picture = Image1(y).Picture
Picture1.Picture = PictureClip2.GraphicCell(y)
'在图像控件中显示动画
Form1.Icon = Image1(y).Picture
```

```
'改变窗体的图标
Text2.Text = y
Slider1.Value = y
'设置控件显示值
End Sub
```

6. 响应按钮事件

在窗体放置有四个按钮控件，它们的作用如下所示：

- Command4 按钮：开始播放动画；
- Command1 按钮：暂停播放动画，如果再次单击此按钮，将会恢复播放动画；
- Command2 按钮：停止播放动画，同时返回到第一桢；
- Command3 按钮：结束程序运行。

为了实现以上四个按钮控件的程序功能，在窗体上双击鼠标左键，在弹出代码窗口的对象列表框中选择按钮对象 Command1、Command2、Command3 和 Command4，在对应的事件列表中分别选择 Private Sub Command1_Click()、Private Sub Command2_Click()、Private Sub Command3_Click() 和 Private Sub Command4_Click()事件，把光标移动到事件处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Command2_Click()
toggle = 0
'停止播放
Picture1.Picture = PictureClip2.GraphicCell(0)
'定格显示第一桢动画
Text2.Text = 1
Slider1.Value = 1
'设置控件中的显示数值
End Sub
```

程序说明：

在程序运行的过程中，当用户单击按钮“停止播放”时，就会激活控件的 Private Sub Command2_Click()事件，程序首先通过语句 toggle = 0 停止播放动画。

然后通过代码 Picture1.Picture = PictureClip2.GraphicCell(0)定格显示第一桢动画。

最后通过两条语句设置控件中的显示数值。

注意：

> Command1、Command3 和 Command4 控件的事件响应代码由于比较容易理解，在这里就不多加赘述了，请读者参看附后的源程序代码。

7. 响应滚动条事件

在窗体上放置的滚动条控件可以在程序运行的过程中动态的跟踪动画播放进程，同时用户还可以通过调节滚动条来控制动画播放进程。

为此，在程序设计的过程中，用鼠标左键双击窗体上的滚动条控件 Slider1，在弹出代码窗口的对象列表框中选择滚动条控件 Slider1，在对应的事件列表中选择 Private Sub Slider1_Scroll()事件，把光标移动到事件处理过程中。

添加如下所示的事件响应代码：

```
Private Sub Slider1_Scroll()
y = Slider1.Value
'拖动滚动条
Picture1.Picture = PictureClip2.GraphicCell(y)
'定格显示图像
Form1.Icon = Image1(y).Picture
Text2.Text = y
```

```
End Sub
```

程序说明：

在程序运行过程中，当用户利用鼠标拖动滚动条上的滑动块时，就会激活控件的 Private Sub Slider1_Scroll() **事件，程序首先通过语句** y = Slider1.Value、Picture1.Picture = PictureClip2.GraphicCell(y)**定格显示图像。**

然后语句 Form1.Icon = Image1(y).Picture **在窗体的标题栏中显示图标。**

最后在文本框中显示当前的动画播放桢数。

8. 运行程序

按照附后的源程序清单添加剩余的代码，设置项目的启动窗体为主窗体 Form1**，存储文件，运行程序，程序运行结果如图** 6-8 **所示。**

图 6-8 **程序运行结果**

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Object = "{27395F88-0C0C-101B-A3C9-08002B2F49FB}#1.1#0"; "PICCLP32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin Visual Basic.Form Form1
BackColor   =   &H00C0C0C0&
BorderStyle =   3
Caption =   "播放动画"
Icon    =   "Redtop.frx":0000
MaxButton   =   0
MinButton   =   0
Moveable    =   0
ScaleHeight =   306
ScaleWidth  =   459
StartUpPosition =   2
End

Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False
```

```
Dim y As Integer
'记数变量
Dim toggle As Integer
'标志变量

Private Sub Command1_Click()
    If toggle = 0 Then
        toggle = 1
    Else
        toggle = 0
    End If
'暂停与播放动作切换
End Sub

Private Sub Command2_Click()
toggle = 0
'停止播放
Picture1.Picture = PictureClip2.GraphicCell(0)
'定格显示第一桢动画
Text2.Text = 1
Slider1.Value = 1
'设置控件中的显示数值
End Sub

Private Sub Command3_Click()
End
'结束程序的运行
End Sub

Private Sub Command4_Click()
toggle = 1
'开始播放动画
End Sub

Private Sub Form_Load()
Picture1.Picture = PictureClip2.GraphicCell(0)
'显示第一桢动画
y = 0
'初始化变量
Picture1.Left = (Picture6.ScaleWidth - Picture1.Width) / 2
Picture1.Top = (Picture6.ScaleHeight - Picture1.Height) / 2
'设置控件显示位置
Text1.Text = PictureClip2.Cols * PictureClip2.Rows
Text2.Text = 0
'初始化文本框中的文本
```

```
End Sub

Private Sub runtop()
y = y + 1: If y = 18 Then y = 0
Picture1.Picture = PictureClip2.GraphicCell(y)
'在图像控件中显示动画
Form1.Icon = Image1(y).Picture
'改变窗体的图标
Text2.Text = y
Slider1.Value = y
'设置控件显示值
End Sub

Private Sub Slider1_Scroll()
y = Slider1.Value
'拖动滚动条
Picture1.Picture = PictureClip2.GraphicCell(y)
'定格显示图像
Form1.Icon = Image1(y).Picture
Text2.Text = y
End Sub

Private Sub Timer1_Timer()
If toggle = 1 Then runtop
'调用函数显示动画
End Sub
```

6.2 制作视频播放器

上面的示例程序作用是播放用户计算机上的一系列图像文件来实现动画的效果，但是在多媒体应用程序中还要经常用到另外一种动画，即以*.avi 为后缀的视频文件，在 Visual Basic 6.0 中为了实现多媒体的功能，特地为开发多媒体应用程序的用户提供了一个 MCI 控件——MMControl 控件，通过这个控件，用户通过简单的编程语句就可以实现驱动硬件设备的功能。

下面就以一个示例程序来说明，在 Visual Basic 6.0 中如何通过 MCI 控件来播放视频文件。

注意：

- 在示例程序运行的过程中，可以动态的显示动画文件的播放进程，更重要的是用户可以通过滚动条来调节动画播放的进度。

具体的程序设计步骤如下所示：

1．开始工作

首先用鼠标选择 Windows 操作系统“开始”菜单中的“程序”/Microsoft Visual Studio 中的 Microsoft Visual Basic 6.0 选项来激活 Visual Basic6 应用程序，在 Visual Basic 6.0 的集成开发环境中用鼠标选择“文件”菜单中的“新建工程”选项，在屏幕上就会弹出一个如图 6-9 所示的“新建工程”对话框。

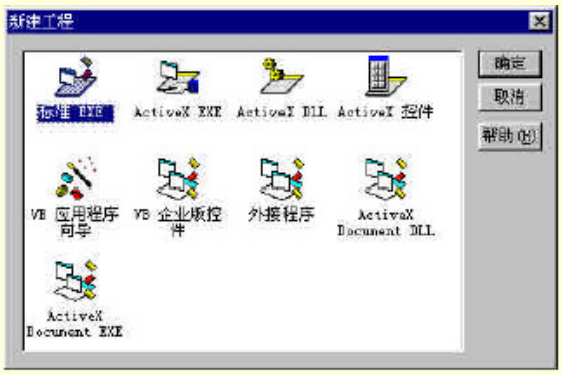


图 6-9 “新建工程”对话框

在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 6 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form Form1
BorderStyle =   3
MaxButton    =  0
MinButton   =   0
Moveable    =   0
StartUpPosition =   2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；
- 窗体的标题栏中为空，即在程序运行初期不显示任何文本或字符串；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

2. 添加控件

在程序运行的过程中，为了能够显示动画，需要有显示容器，为此要向当前空白的窗体上添加一个 PictureBox 控件，同时向窗体上添加七个按钮控件来控制动画播放，另外添加的两个 Label 控件和两个 TextBox 控件用于显示动画文件的基本信息和播放进度，添加控件后的窗体如图 6-10 所示。

图 6-10 添加控件后的窗体

其中添加到窗体上的各个控件的属性设置如下所示：

```
Begin Visual Basic.TextBox Text2
 Text    =   "Text2"
End
```

```
   Begin Visual Basic.TextBox Text1
    Text    =   "Text1"
   End
   Begin Visual Basic.CommandButton Command7
Caption =   "跳到结尾"
Height  =   495
Left    =   5280
Top =   960
Width   =   1335
   End
   Begin Visual Basic.CommandButton Command6
Caption =   "返回开头"
Height  =   495
Left    =   5280
Top =   360
Width   =   1335
   End
   Begin Visual Basic.CommandButton Command5
Caption =   "后退一桢"
Height  =   495
Left    =   240
Top =   2760
Width   =   1335
   End
   Begin Visual Basic.CommandButton Command4
Caption =   "步进一桢"
Height  =   495
Left    =   240
Top =   2160
Width   =   1335
   End
   Begin Visual Basic.CommandButton Command3
Caption =   "暂停/停止"
Height  =   495
Left    =   240
Top =   1560
Width   =   1335
   End
   Begin Visual Basic.CommandButton Command2
Caption =   "播放视频"
Height  =   495
Left    =   240
Top =   960
Width   =   1335
   End
   Begin Visual Basic.PictureBox Picture1
```

```
Height  =   2895
Left    =   1800
Top =   360
Width   =   3255
  End
   Begin Visual Basic.CommandButton Command1
Caption =   "打开文件"
Height  =   495
Left    =   240
Top =   360
Width   =   1335
  End
   Begin Visual Basic.Label Label2
Caption =   "当前桢数："
  End
   Begin Visual Basic.Label Label1
Caption =   "总桢数："
  End
```

添加到窗体上的控件的作用如下所示：

- Text1 **控件和** Text2 **控件 在程序运行的过程中，分别用来显示播放动画文件的总桢数和当前播放桢数；**
- Picture1 **控件：在程序运行的过程中充当动画显示的容器；**
- Label1 **控件和** Label2 **控件：显示固定的提示文本“ 总桢数：”和“ 当前桢数 ”；**
- Command **按钮控件：在程序运行的过程中完成对动画文件的各种操作，如打开、播放和暂停等。**

3. **添加** ActiveX **控件**

提示：

以下仍然需要用到 ActiveX **控件，调用方法与前面所讲的方法相同。**

在本示例程序设计的过程中，需要用到三个 ActiveX **控件——**CommonDialog **控件、**Slider **控件和** MMControl **控件。为此，在程序设计的过程中，用鼠标左键单击菜单“ 工程 ”中的“ 部件 ”选项，或者用鼠标右键单击工具箱，在弹出的菜单中选择“ 部件 ”选项，在屏幕上都会弹出一个如图** 6-11 **所示添加** ActiveX **的对话框。**

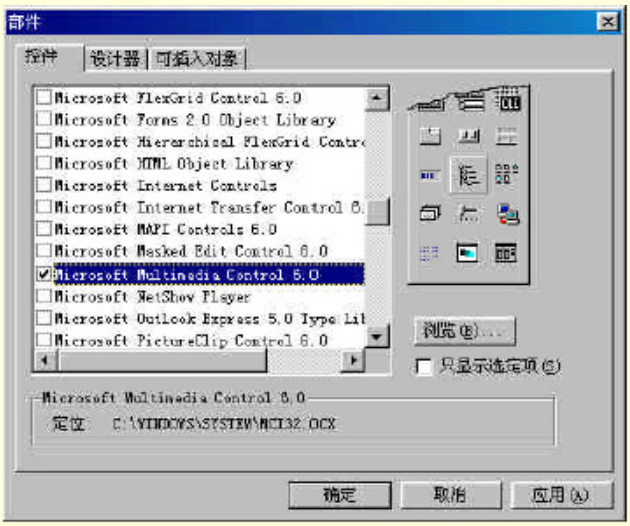


图 6-11 Components **对话框**

在添加 ActiveX **控件的“ 部件 ”对话框中选择** Microsoft Common Dialog Control 6.0、Microsoft Multimedia Control 6.0 **和** Microsoft Windows Common Controls 6.0 **三个选项，单击“ 确定 ”按钮，程序就会自动的向当前的**

项目文件中添加了示例程序所需要的 ActiveX 控件，这三个 ActiveX 控件的声明如下所示：

```
Object = "{C1A8AF28-1257-101B-8FB0-0020AF039CA3}#1.1#0"; "MCI32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
```

添加三个 ActiveX 控件后的工具箱如图 6-12 所示。

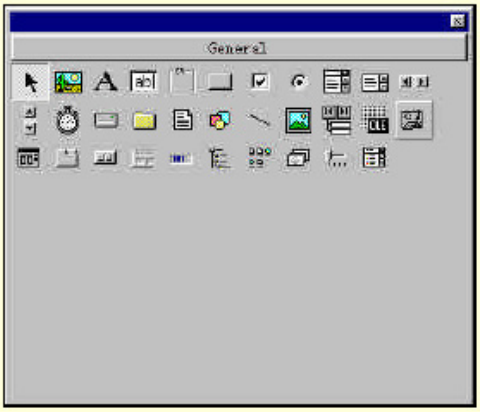


图 6-12　添加 ActiveX 控件后的工具箱

在程序设计的过程中，向当前空白的窗体上添加一个 CommonDialog 控件、一个 Slider 控件和一个 MMControl 控件，它们的属性设置如下所示：

```
   Begin MCI.MMControl Avi
Height  =   495
Left    =   1680
Top =   3720
Visible =   0
Width   =   3540
UpdateInterval  =   100
DeviceType  =   "avivideo"
FileName    =   ""
   End
   Begin MSComDlg.CommonDialog CommonDialog1
Left    =   5520
Top =   3720
   End
   Begin MSComctlLib.Slider Slider1
Height  =   375
Left    =   1560
Top =   3360
Width   =   3615
   End
```

添加到窗体上的三个 ActiveX 控件的作用如下所示：

- CommonDialog 控件：在程序运行的过程中，通过调用这个控件相应的方法，可以显示一个对话框，用户在其中可以选择一个有效的动画文件；
- Slider 控件：在播放动画文件的过程中，通过这个控件可以动态的显示动画播放进程，也可以通过这个控件手动调节动画播放进度；
- MMControl 控件：在多媒体播放设备和用户之间起一个连接的作用，在程序运行的过程中，通过调用这个控件的一些简单的命令就可以实现播放、暂停和步进等动画文件操作功能。

添加控件后的窗体如图 6-13 所示。

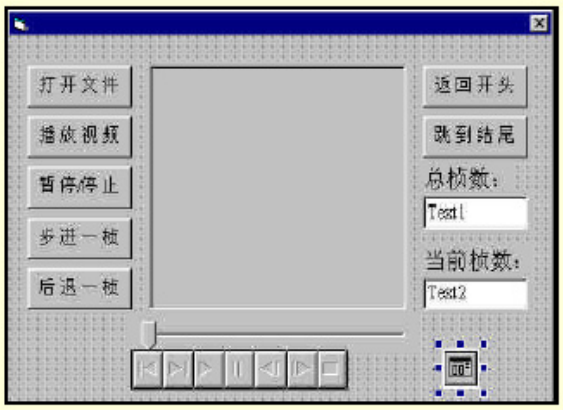


图 6-13 添加 ActiveX 控件后的窗体

在以上三个 ActiveX 控件属性设置的过程中，用户可以在属性列表框中修改相应控件的属性，也可以单击属性列表框中 Custom 选项右侧的按钮，在弹出的对话框中自定义 ActiveX 控件属性，如图 6-14 即为自定义的 CommonDialog 控件属性设置对话框。

图 6-14 自定义的 CommonDialog 控件属性对话框

经过自定义设置属性后的 CommonDialog 控件具有如下所示的特性：

- CommonDialog 控件所打开的对话框标题栏中显示字符串“请选择一个合适的动画文件”；
- 对话框的缺省工作路径为 d:\microsofi visual studio；
- 在 CommonDialog 控件所打开的对话框中只能够显示以*.avi 为文件后缀的动画文件。

而如图 6-15 为自定义的 MMControl 控件属性设置对话框。

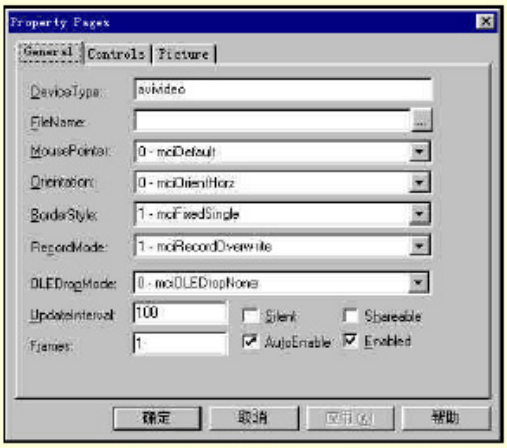


图 6-15 自定义的 MMControl 控件属性设置对话框

4. 程序初始化

在本示例程序中，所谓程序初始化，指的就是为窗体 Private Sub Form_Load()事件所添加的响应代码。

在程序设计的过程中，用鼠标左键双击窗体上的空白处，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中选择窗体 Form，在对应的事件列表中选择事件 Private Sub Form_Load()，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Form_Load()
Form1.Avi.hWndDisplay = Form1.Picture1.hWnd
'设置动画显示窗口
Form1.Avi.DeviceType = "AviVideo"
'设置多媒体类型
Form1.CommonDialog1.DialogTitle = "请选择一个合适的动画文件:"
'设置对话框标题
Form1.CommonDialog1.Filter = "动画文件(*.avi)|*.avi"
'设置文件过滤器
Form1.CommonDialog1.InitDir = "d:\microsofi visual studio"
'设置缺省工作路径
Form1.Command2.Enabled = False
Form1.Command3.Enabled = False
Form1.Command4.Enabled = False
Form1.Command5.Enabled = False
Form1.Command6.Enabled = False
Form1.Command7.Enabled = False
'设置按钮控件的有效状态
Form1.Text1.Text = ""
Form1.Text2.Text = ""
'清空文本框
End Sub
```

在程序运行的初期，事件 Private Sub Form_Load()中的代码就会被执行，程序通过 Form1.Avi.hWndDisplay = Form1.Picture1.hWnd 语句设置动画的显示容器为窗体上的 Picture1 控件，设置多媒体设备类型为 AviVideo，即播放动画文件，然后通过三条语句：

```
Form1.CommonDialog1.DialogTitle = "请选择一个合适的动画文件:"
Form1.CommonDial-og1.Filter = "动画文件(*.avi)|*.avi"
Form1.CommonDialog1.InitDir = "d:\microsofi visual studio"
```

设置可对话框标题、文件过滤器和缺省工作路径，在程序初始化的最后，程序通过 Form1.Command2.Enabled = False 等八条语句来设置按钮控件的有效状态和清空窗体上的两个文本框。经过程序初始化后的运行窗体如图 6-16 所示。

图 6-16 经过程序初始化后的运行窗体

5. 响应按钮事件

在窗体上放置有七个 CommandButton 控件，它们的作用如下所示：

- Command1 控件：显示一个打开的对话框，在其中用户可以选择一个有效的动画文件；
- Command7 控件：把当前所打开动画文件的当前桢定位在文件的末尾；
- Command6 控件：把当前所打开动画文件的当前桢定位在文件的开头；
- Command5 控件：如果当前桢不是动画文件的第一桢，就把当前桢定位在前一桢；
- Command4 控件：如果当前桢不是动画文件的最后一桢，就把当前桢定位在后面一桢；
- Command3 控件：暂停播放动作，如果再次单击这个按钮，又会重新播放动画文件；
- Command2 控件：在程序运行的过程中，如果用户选择了一个有效的动画文件，那么单击这个按钮就可以播放动画。

注意：

在这里由于篇幅的关系，仅对 Command1 按钮的事件响应代码加以说明，其他六个按钮的代码添加过程请读者参看附后的源程序清单。

在程序设计的过程中，用鼠标左键双击窗体上的空白处，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中选择控件 Command1，在对应的事件列表中选择事件 Private Sub Command1_Click()，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Command1_Click()
Form1.CommonDialog1.ShowOpen
'显示一个对话框
If Form1.CommonDialog1.FileName <> 0 Then
   Avi.FileName = Form1.CommonDialog1.FileName
   '设置播放文件
   Avi.Command = "Open"
   '打开多媒体播放设备
   Form1.Caption = DialogCaption + Avi.FileName
   '显示播放文件的路径和文件名
   Form1.Slider1.Min = 0
   Form1.Slider1.Max = Form1.Avi.Length
   Form1.Slider1.Value = 0
   '初始化滚动条控件
   Form1.Text1.Text = Form1.Avi.Length
   Form1.Text2.Text = Form1.Avi.Position
   '初始化文本框控件
   Form1.Command2.Enabled = True
   Form1.Command3.Enabled = True
   Form1.Command4.Enabled = True
   Form1.Command5.Enabled = True
   Form1.Command6.Enabled = True
   Form1.Command7.Enabled = True
   '设置按钮控件的有效状态
Else
    Exit Sub
    '退出处理过程
End If
End Sub
```

在程序运行的过程中，当用户用鼠标左键单击按钮“打开文件”时，就会激活控件的 Private Sub

Command1_Click()事件，程序首先通过 Form1.CommonDialog1.ShowOpen 语句来显示一个如图 6-17 所示的对话框。

图 6-17 选择打开文件的对话框

在选择打开文件的对话框中用户可以选择一个以*.avi 为后缀的动画文件，单击“打开”按钮后，程序就会通过 Avi.FileName = Form1.CommonDialog1.FileName 为多媒体播放设备设置播放文件，然后通过代码 Avi.Command = "Open"来打开多媒体播放设备，并且对滚动条控件的 Min、Max 和 Value 属性进行了设置，最后通过 Form1.Command2.Enabled = True 等六条语句设置了按钮控件的有效状态。

6. 响应 MCI 控件事件

在程序设计的过程中，用鼠标左键双击窗体上的 MCI 控件 MMControl1，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中选择控件 MMControl1，在对应的事件列表中选择事件 Private Sub avi_StatusUpdate()，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub avi_StatusUpdate()
Form1.Slider1.Value = CInt(Form1.Slider1.Max * Form1.Avi.Position / Form1.Avi.Length)
'动态的显示当前动画播放进程
Form1.Text2.Text = Form1.Avi.Position
'显示当前播放桢
End Sub
```

程序说明：

在控件属性设置的过程中，我们把 MMControl1 控件的 Interval 属性设置为 100，即在动画播放的过程中每隔 100 微秒程序就会自动的激活一个 Private Sub avi_StatusUpdate()事件。

在程序运行的过程中，当用户选择一个有效的动画文件并且开始播放后，每隔 100 微秒就会自动的激活一个 Private Sub avi_StatusUpdate()事件，程序首先通过 Form1.Slider1.Value = CInt(Form1.Slider1.Max * Form1.Avi.Position / Form1.Avi.Length)语句在滚动条控件中动态的显示当前动画播放进程，然后通过代码 Form1.Text2.Text = Form1.Avi.Position 在文本框 Text2 中显示动画文件的当前播放桢。

结果如图 6-18 所示。

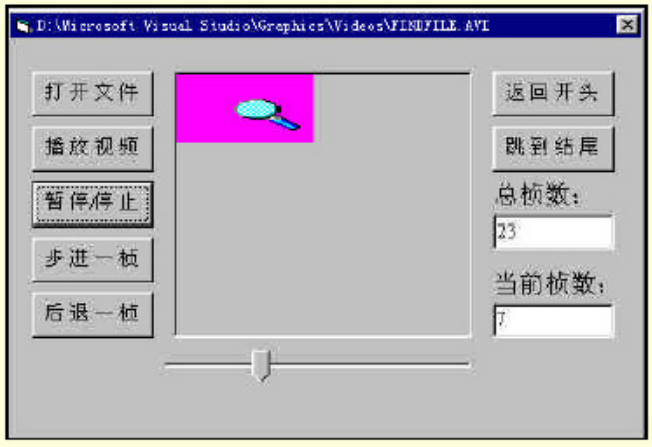


图 6-18 滚动条控件动态显示动画播放进程

7. 响应滚动条事件

提示：

> **在本示例程序运行的过程中，滚动条控件可以动态的显示动画播放进程，用户还可以通过调节滚动条上的滑动块来手动调节动画播放进程。**

为了能够实现以上的功能，在程序设计的过程中，用鼠标左键双击窗体上的滚动条控件 Slider1**，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中选择控件** Slider1**，在对应的事件列表中选择事件** Private Sub Slider1_Scroll()**，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：**

```
Private Sub Slider1_Scroll()
Form1.Avi.From = CInt(Form1.Avi.Length * Form1.Slider1.Value / Form1.Slider1.Max)
'设置动画起始点
Form1.Avi.To = CInt(Form1.Avi.Length * Form1.Slider1.Value / Form1.Slider1.Max)
'设置动画结束点
Form1.Avi.Command = "play"
'开始播放
End Sub
```

程序说明：

在程序运行的过程中，当用户用鼠标拖动滚动条上的滑动块时，就会激活控件的 Private Sub Slider1_Scroll() **事件，程序首先通过语句** Form1.Avi.From = CInt(Form1.Avi.Length * Form1.Slider1.Value / Form1.Slider1.Max)**和** Form1.Avi.To = CInt(Form1.Avi.Length * Form1.Slider1.Value / Form1.Slider1.Max)**设置了播放动画的起始点和结束点，然后通过语句** Form1.Avi.Command = "play"**实现定格显示动画的效果。**

8. 运行程序

按照附后的源程序清单添加剩余的代码，设置项目的启动窗体为主窗体 Form1**，存储文件，运行程序，程序运行结果如图** 6-19 **所示。**

图 6-19 **程序运行结果**

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Object = "{C1A8AF28-1257-101B-8FB0-0020AF039CA3}#1.1#0"; "MCI32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin Visual Basic.Form Form1
AutoRedraw =   -1
BorderStyle =   3
```

```
MaxButton   =   0
MinButton   =   0
Moveable    =   0
ScaleHeight =   4335
ScaleWidth  =   6915
ShowInTaskbar   =   0
StartUpPosition =   2
End
Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False

Private Sub Command1_Click()
Form1.CommonDialog1.ShowOpen
'显示一个对话框
If Form1.CommonDialog1.FileName <> "" Then
   Avi.FileName = Form1.CommonDialog1.FileName
   '设置播放文件
   Avi.Command = "Open"
   '打开多媒体播放设备
   Form1.Caption = DialogCaption + Avi.FileName
   '显示播放文件的路径和文件名
   Form1.Slider1.Min = 0
   Form1.Slider1.Max = Form1.Avi.Length
   Form1.Slider1.Value = 0
   '初始化滚动条控件
   Form1.Text1.Text = Form1.Avi.Length
   Form1.Text2.Text = Form1.Avi.Position
   '初始化文本框控件
   Form1.Command2.Enabled = True
   Form1.Command3.Enabled = True
   Form1.Command4.Enabled = True
   Form1.Command5.Enabled = True
   Form1.Command6.Enabled = True
   Form1.Command7.Enabled = True
   '设置按钮控件的有效状态
Else
    Exit Sub
End If
End Sub

Private Sub Command2_Click()
Form1.Avi.Command = "play"
'播放动画文件
```

```
End Sub

Private Sub Command3_Click()
Form1.Avi.Command = "pause"
'暂停/停止播放
End Sub

Private Sub Command4_Click()
Form1.Avi.Command = "step"
'步进一桢
End Sub

Private Sub Command5_Click()
Form1.Avi.Command = "back"
'后退一桢
End Sub

Private Sub Command6_Click()
Form1.Avi.Command = "prev"
'返回开头
End Sub

Private Sub Command7_Click()
Form1.Avi.Command = "next"
'跳到结尾
End Sub

Private Sub Form_Load()
Form1.Avi.hWndDisplay = Form1.Picture1.hWnd
'设置动画显示窗口
Form1.Avi.DeviceType = "AviVideo"
'设置多媒体类型
Form1.CommonDialog1.DialogTitle = "请选择一个合适的动画文件:"
'设置对话框标题
Form1.CommonDialog1.Filter = "动画文件(*.avi)|*.avi"
'设置文件过滤器
Form1.CommonDialog1.InitDir = "d:\microsofi visual studio"
'设置缺省工作路径
Form1.Command2.Enabled = False
Form1.Command3.Enabled = False
Form1.Command4.Enabled = False
Form1.Command5.Enabled = False
Form1.Command6.Enabled = False
Form1.Command7.Enabled = False
'设置按钮控件的有效状态
Form1.Text1.Text = ""
```

```
Form1.Text2.Text = ""
'清空文本框
End Sub

Private Sub avi_StatusUpdate()
Form1.Slider1.Value = CInt(Form1.Slider1.Max * Form1.Avi.Position / Form1.Avi.Length)
'动态的显示当前动画播放进程
Form1.Text2.Text = Form1.Avi.Position
'显示当前播放桢
End Sub

Private Sub Slider1_Scroll()
Form1.Avi.From = CInt(Form1.Avi.Length * Form1.Slider1.Value / Form1.Slider1.Max)
'设置动画起始点
Form1.Avi.To = CInt(Form1.Avi.Length * Form1.Slider1.Value / Form1.Slider1.Max)
'设置动画结束点
Form1.Avi.Command = "play"
'开始播放
End Sub
```

6.3 制作音频播放器

一个真正的多媒体应用程序应该不但能够实现播放动画、声音等文件的功能，而且还能够获得媒体文件的各种信息（如长度、动画的桢数、CD 的音轨数等），以便于进行进一步的处理。

如图 6-20 所示为一个声音播放器运行时的窗体外观。

图 6-20 声音播放器

下面就以一个制作声音播放器的示例来说明如何在多媒体应用程序的制作过程中得到媒体文件的各种信息，它的具体步骤如下。

1. 开始工作

首先启动一个新的项目，向缺省的工具箱中添加一个 CommonDialog 控件和一个 MMControl 控件，其中 CommonDiaolog 控件用于显示一个打开声音文件的对话框。如图 6-21 所示。

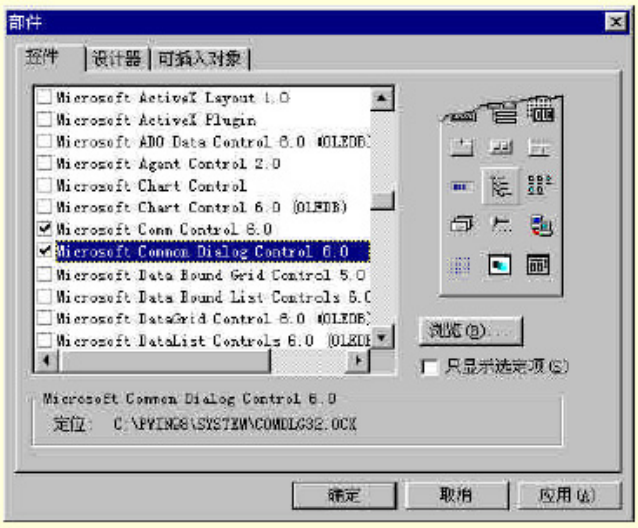


图 6-21　添加控件对话框

2．向窗体添加控件

把一个 CommonDialog 控件和一个 MMControl 控件放置到窗体上，它们的属性设置如表 6-1 所示。

表 6-1　控件的属性设置

属　性	设　置
(Name)	MMControl1
AutoEnable	True
DeviceType	
Enabled	True
FileName	
Height	330
Left	0
Top	2280
UpdateInterval	1000
Visible	False
Width	3540
(Name)	CommonDialog1
DialogTitle	请选择一个有效的声音文件
FileName	*.wav
Filter	*.wav
InitDir	c:\pwin98
Left	3600
Top	2160

这样设置的 MMControl 控件有以下的特性：

- 控件中各个按钮的有效状态有控件自己决定；
- 播放动画的设备名称由程序决定；
- 每隔 1000 毫秒激活一次 MMControl1_StatusUpdate()事件；
- 在程序的运行过程中 MMControl 控件处于不可见的状态。

如表 6-1 所示的属性设置使得 CommonDialog 控件有如下的特性：

- 显示的对话框的标题栏中显示字符串“请选择一个有效的声音文件”；
- 对话框的缺省目录为 c:\pwin95；
- 在对话框中只能够显示以“*.wav”为结尾的动画文件。

3．向窗体添加按钮

在窗体的设计阶段向窗体上添加七个 CommandButton 控件，它们的属性设置及其相应的功能如表 6-2 所示。

表 6-2 按钮的属性设置

属 性	设 置
(Name)	Open
Caption	打开文件
Enabled	True
Height	495
Left	240
Top	240
Width	975
(Name)	Play
Caption	播放文件
Enabled	True
Height	495
Left	240
Top	840
Width	975
(Name)	Pause
Caption	暂停播放
Enabled	True
Height	495
Left	240
Top	1440
Width	975
(Name)	Close1
Caption	关闭文件
Enabled	True
Height	495
Left	1320
Top	1440
Width	975
(Name)	Stop1
Caption	停止播放
Enabled	True
Height	495
Left	2400
Top	1440
Width	975
(Name)	Prev
Caption	返回开头
Enabled	True
Height	495
Left	3480
Top	1440
Width	975

其中各个按钮的功能如下所示：

- Open 按钮的作用是显示一个打开文件的对话框，并且把文件路径和文件名传递给控件 MMControl1；
- Play 按钮的作用是播放一个打开的声音文件；
- Pause 按钮的作用是暂停播放声音文件；
- Close1 按钮的作用是关闭一个打开的声音文件；
- Stop1 按钮的作用是停止播放声音文件；
- Prev 按钮的作用是把一个播放到文件末尾的声音文件重新定位到文件开头。

4. 添加 Label 控件

向窗体上添加三个 Label 控件，作用是显示播放文件的信息，它们的属性设置如表 6-3 所示。

表 6-3 控件 Label 的属性设置

属 性	设 置
(Name)	Label1
Caption	文件类型:
Height	255
Left	120
Top	240

Width	2655
(Name)	Label2
Alignment	0 - Left Justify
AutoSize	True
Caption	**文件的长度**:
Height	180
Left	120
Top	480
Width	990
(Name)	Label3
Alignment	0 - Left Justify
AutoSize	True
Caption	**播放状态**:
Height	180
Left	120
Top	720
Width	810

如表 6-4 **设置的属性使得控件** Label1 **和** Label2 **具有如下的特性：**

- **控件中文字的对齐方式都是左对齐；**
- **控件会改变自身的尺寸来适应文字的大小。**

其中各个控件的功能如下所示：

- Label1 **控件的作用是显示播放文件的类型；**
- Label2 **控件的作用是显示文件的播放长度；**
- Label3 **控件的作用是显示程序中文件的播放状态。**

5. **添加计时器控件**

向窗体上添加一个计时器控件，它的属性设置如图 6-22 **所示。**

添加控件后的窗体如图 6-23 **所示。**

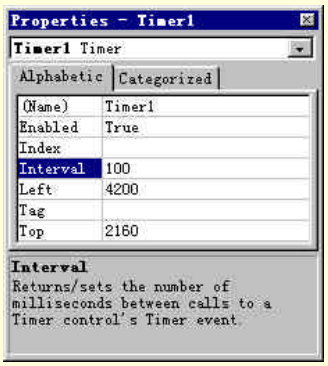


图 6-22 **计时器控件的属性设置**

图 6-23 **加控件后的窗体**

6. **程序的初始化**

在窗体的设计阶段双击窗体，在窗体的 Form_Load()**事件和** Form_Unload()**事件中分别添加下列代码：**

```
Private Sub Form_Load()
```

```
MMControl1.DeviceType = "waveaudio"
'设置播放文件的类型
Label1.Caption = "文件的类型:" & MMControl1.DeviceType
'显示文件的类型
MMControl1.AutoEnable = False
'设置由程序控制控件的状态
Play.Enabled = False
Close1.Enabled = False
Pause.Enabled = False
Stop1.Enabled = False
Prev.Enabled = False
'设置控件的初始化状态
Timer1.Enabled = False
'设置计时器控件的有效状态
End Sub
```

程序首先设置多媒体设备类型为 WaveAudio，并且通过 Label1.Caption = "文件的类型:" & MMControl1.DeviceType 语句来显示文件的类型，然后通过语句 MMControl1.AutoEnable = False 设置 MMControl1 控件中的各个按钮的状态都由程序来控制。

注意：

> ✍ 为了保证合理的利用多媒体资源和程序的正常结束，所以在关闭窗体的同时要关闭打开的多媒体设备。

为了实现上述功能，初始化过程还需要添加如下的程序代码：

```
Private Sub Form_Unload(Cancel As Integer)
MMControl1.Command = "close"
'关闭窗体的同时关闭文件
End Sub
```

7. 添加代码

下面仅以 Open_Click()事件为例来说明添加代码的过程，其他代码的详细添加过程请读者参见附后的程序源代码。

在程序的设计阶段，双击“打开文件”按钮，在它的 Open_Click()事件中添加下列代码：

```
Private Sub Open_Click()
CommonDialog1.Action = 1
'显示一个打开文件的对话框
MMControl1.FileName = CommonDialog1.FileName
'选择播放文件
If MMControl1.FileName = "" Then
   MsgBox "请选择一个有效的声音文件"
   CommonDialog1.Action = 1
Else
   MMControl1.Command = "open"
   Timer1.Enabled = True
   Close1.Enabled = True
End If
Label2.Caption = "文件的长度:" & MMControl1.Length & "微秒"
'显示文件的信息
End Sub
```

在程序的运行过程中单击“打开文件”按钮，就会激活 Open_Click()事件，程序首先显示一个打开文件的对话框，提示用户选择一个有效的声音文件，然后把选中的声音文件路径和文件名传递给 MMControl1 控件；

注意：

- 打开一个有效的声音文件后通过 Label2.Caption = "文件的长度:" & MMControl1.Length & "微秒" 语句来显示声音文件的播放长度。

8. 运行程序

存储文件，运行程序，初始画面如图 6-24 所示。

图 6-24 程序运行的初始画面

单击“打开文件”按钮，就会弹出如图 6-25 所示的打开文件的对话框。

图 6-25 打开文件对话框

在打开文件的对话框中选择一个有效的声音文件后，程序就会自动的打开它，并且会读出文件的各种信息，窗体如图 6-26 所示。

图 6-26 程序的运行结果

附程序源代码：

程序清单

```
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{C1A8AF28-1257-101B-8FB0-0020AF039CA3}#1.1#0"; "MCI32.OCX"
Begin Visual Basic.Form Form1
Caption =   "Form1"
ClientHeight    =   2865
ClientLeft  =   60
ClientTop   =   345
ClientWidth =   4680
LinkTopic   =   "Form1"
ScaleHeight =   2865
ScaleWidth  =   4680
StartUpPosition =   3  'Windows Default
Begin Visual Basic.Timer Timer1
Interval    =   100
Left    =   4200
Top =   2160
End

Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False
Private Sub Close1_Click()
MMControl1.Command = "close"
'关闭文件
End Sub

Private Sub Form_Load()
MMControl1.DeviceType = "waveaudio"
'设置播放文件的类型
Label1.Caption = "文件的类型:" & MMControl1.DeviceType
'显示文件的类型
MMControl1.AutoEnable = False
'设置由程序控制控件的状态
Play.Enabled = False
Close1.Enabled = False
Pause.Enabled = False
Stop1.Enabled = False
Prev.Enabled = False
'设置控件的初始化状态
Timer1.Enabled = False
'设置计时器控件的有效状态
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
MMControl1.Command = "close"
'关闭窗体的同时关闭文件
End Sub

Private Sub Open_Click()
CommonDialog1.Action = 1
'显示一个打开文件的对话框
MMControl1.FileName = CommonDialog1.FileName
'选择播放文件
If MMControl1.FileName = "" Then
   MsgBox "请选择一个有效的声音文件"
   CommonDialog1.Action = 1
Else
   MMControl1.Command = "open"
   Timer1.Enabled = True
   Close1.Enabled = True
End If
Label2.Caption = "文件的长度:" & MMControl1.Length & "微秒"
'显示文件的信息
End Sub

Private Sub Pause_Click()
MMControl1.Command = "pause"
'暂停播放
End Sub

Private Sub Play_Click()
MMControl1.Command = "play"
'播放声音文件
End Sub

Private Sub Prev_Click()
MMControl1.Command = "prev"
'返回到文件开头
End Sub

Private Sub Stop1_Click()
MMControl1.Command = "stop"
'停止播放文件
End Sub

Private Sub Timer1_Timer()
Select Case MMControl1.Mode
  Case 524
```

```
    Label3.Caption = "播放状态:文件未打开"
  Case 525
    Label3.Caption = "播放状态:打开文件"
  Case 526
    Label3.Caption = "播放状态:播放文件"
  Case 529
    Label3.Caption = "播放状态:暂停播放"
End Select
'显示播放的状态
Play.Enabled = MMControl1.CanPlay
Stop1.Enabled = MMControl1.CanPlay
Pause.Enabled = MMControl1.CanPlay
'设置控件的有效状态
If MMControl1.Position < MMControl1.Length Then
   Prev.Enabled = False
Else
   Play.Enabled = False
   Pause.Enabled = False
   Stop1.Enabled = False
   Prev.Enabled = True
End If
'判断播放的位置
End Sub
```

6.4 MCIWnd 多媒体制作

利用前面提到的 MMControl 控件可以很简单的就实现播放动画、视频和声音等多媒体文件。

Visual Basic 6.0 为了用户开发动画应用程序的方便，还提供了另外一个更加简单的动画播放控件 MCIWnd。下面通过一个示例程序向读者说明 MCIWnd 控件的使用方法和常用技巧。

提示：

> 读者在以下学习的过程中，应当重点掌握 MMControl 控件和 MCIWnd 控件各自的特点，充分了解多媒体开发的各种不同手段。

在示例程序运行的过程中，用户在播放动画文件的同时，还可以对动画文件的音量、缩放比例和播放速度等属性进行调节。

具体的程序设计步骤如下所示。

1. 开始工作

首先用鼠标选择 Windows 操作系统“开始”菜单中的“程序”/Microsoft Visual Studio 中的 Microsoft Visual Basic 6.0 选项来激活 Visual Basic6 应用程序，在 Visual Basic6 的集成开发环境中新建一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form Form1
BorderStyle =   3
Caption =   "动画播放程序"
MaxButton   =   0
MinButton   =   0
```

```
Moveable    =   0
ShowInTaskbar   =   0
StartUpPosition =   2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

- 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；
- 窗体的标题栏中显示字符串“动画播放程序”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

2. 添加控件

在程序运行的过程中，为了能够选择播放的动画文件，需要有一个控制按钮。

为此要向当前空白的窗体上添加一个 CommandButton 控件，同时向窗体上添加三个 Label 控件。

添加控件后的窗体如图 6-27 所示。

图 6-27 添加控件后的窗体

添加到窗体上的控件的作用如下所示：

- CommandButton 控件：在程序运行的过程中，单击这个按钮就可以打开一个对话框，在其中用户可以选择一个有效的动画文件；
- 三个 Label 控件：显示固定的提示文本“速度调节：”、“缩放调节：”和“音量调节：”。

四个控件的属性设置如下所示：

```
Begin Visual Basic.CommandButton Command1
  Caption   =   "打开文件"
  Height=   495
  Left  =   1440
  Top   =   3720
  Width =   1455
End
Begin Visual Basic.Label Label3
  Caption   =   "缩放调节："
  Height=   240
  Left  =   3360
  Top   =   3840
  Width =   1050
End
```

```
Begin Visual Basic.Label Label2
  Caption   =   "速度调节："
  Height=   240
  Left  =   240
  Top   =   1200
  Width =   1050
End
Begin Visual Basic.Label Label1
  Caption   =   "音量调节："
  Height=   240
  Left  =   5760
  Top   =   1080
  Width =   1050
End
```

3. 添加 ActiveX 控件

在本示例程序设计的过程中，需要用到三个 ActiveX 控件——CommonDialog 控件、Slider 控件和 MCIWnd 控件。

为此，在程序设计的过程中，用鼠标左键单击菜单“工程”中的“部件”选项，或者用鼠标右键单击工具箱，在弹出的菜单中选择“部件”选项，在屏幕上都会弹出一个如图 6-28 所示添加 ActiveX 控件的对话框。

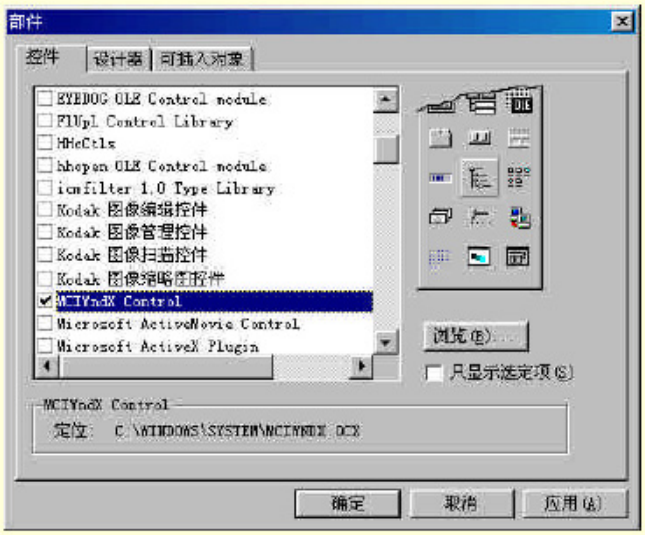


图 6-28 “部件”对话框

在添加 ActiveX 控件的 Components 对话框中选择 MCIWndX Control、MCIWndX Control 和 Microsoft Common Dialog Control 6.0 三个选项。

单击“确定”按钮，程序就会自动的向当前的项目文件中添加了示例程序所需要的 ActiveX 控件。

这三个 ActiveX 控件的声明如下所示：

```
Object = "{288F1520-FAC4-11CE-B16F-00AA0060D93D}#1.0#0"; "MCIWNDX.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
```

添加三个 ActiveX 控件后的工具箱如图 6-29 所示。

图 6-29 添加三个 ActiveX 控件后的工具箱

在程序设计的过程中，向窗体上放置一个 MCIWnd 控件、一个 CommonDialog 控件和三个 Slider 控件，这五个 ActiveX 控件的属性设置如下所示：

```
   Begin MSComDlg.CommonDialog
         CommonDialog1
Left    =   6000
Top =   3480
   End
   Begin MCIWndX.MCIWnd MCIWnd1
Height  =   2895
Left    =   1320
Top =   480
Width   =   4215
   End
   Begin MSComctlLib.Slider Slider1
Left    =   5880
Top =   1320
Orientation =   1
   End
   Begin MSComctlLib.Slider Slider2
Left    =   360
Top =   1440
Orientation =   1
   End
   Begin MSComctlLib.Slider Slider3
Left    =   4560
Top =   3720
   End
```

添加了五个 ActiveX 控件后的窗体如图 6-30 所示。

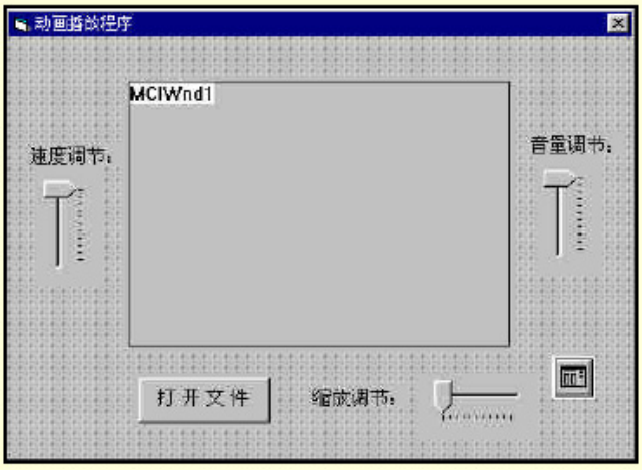


图 6-30 添加了五个 ActiveX 控件后的窗体

其中各个控件的作用如下所示：

- CommonDialog 控件：在程序运行的过程中，通过调用 CommonDialog 控件的相应方法就可以显示一个打开文件的对话框，用户在其中可以选择一个待播放的以*.avi 为后缀的动画文件；
- Slider1 控件：在程序运行的过程中，用户可以通过滑动这个控件的滑动块来调节动画播放的音量；
- Slider2 控件：在程序运行的过程中，用户可以通过滑动这个控件的滑动块来调节动画播放的速度；
- Slider1 控件：在程序运行的过程中，用户可以通过滑动这个控件的滑动块来调节动画播放的缩放比例；
- MCIWnd 控件：显示动画文件的容器。

4. 程序初始化

在本示例程序中，所谓程序初始化，指的就是为窗体 Private Sub Form_Load()事件所添加的响应代码。在程序设计的过程中，用鼠标左键双击窗体上的空白处，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中选择窗体 Form，在对应的事件列表中选择事件 Private Sub Form_Load()，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Form_Load()
Form1.CommonDialog1.DialogTitle = "请选择一个有效的动画文件："
'设置对话框标题
Form1.CommonDialog1.Filter = "动画文件(*.avi)|*.avi"
'设置文件过滤器
Form1.CommonDialog1.InitDir = "c:\windows"
'设置缺省工作路径
Form1.Command1.Enabled = True
'设置按钮控件的有效状态
Form1.Slider1.Enabled = False
Form1.Slider2.Enabled = False
Form1.Slider3.Enabled = False
'设置滚动条控件的有效状态
End Sub
```

在程序运行初期，事件 Private Sub Form_Load()中的代码会被执行，程序通过三条语句：

Form1.CommonDialog1.DialogTitle = "请选择一个有效的动画文件："

Form1.CommonDia-log1.Filter = "动画文件(*.avi)|*.avi"

Form1.CommonDialog1.InitDir = "c:\windows"

分别设置了对话框标题、文件过滤器和缺省的工作路径，然后设置了窗体上按钮控件 Command1 和三个滚动条控件的有效状态。

经过程序初始化后，由 CommonDialog1 控件所打开的对话框如图 6-31 所示。

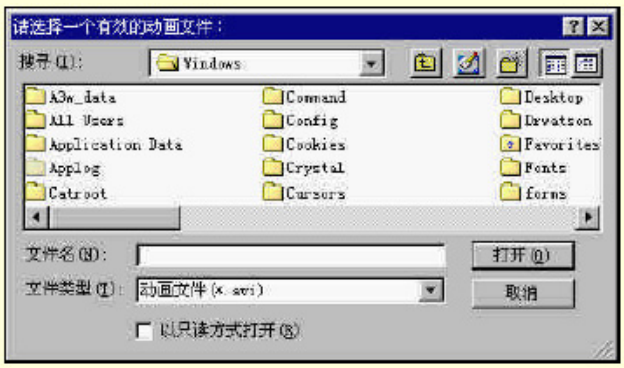


图 6-31 程序初始化后的对话框

5. 响应滚动条事件

在程序设计的过程中，用鼠标左键双击窗体上的空白处，在屏幕上就会弹出一个空白的代码窗口，在代码窗口的对象列表中选择控件 Slider1、Slider2 和 Slider3，在对应的事件列表中分别选择 Private Sub Slider1_Scroll() 事件、Private Sub Slider2_Scroll()事件和 Private Sub Slider3_Scroll()事件，把光标移动到事件的处理过程中，并且添加如下所示的事件响应代码：

```
Private Sub Slider1_Scroll()
Form1.MCIWnd1.Volume = Form1.Slider1.Value * 100
'改变动画播放中的音量
End Sub
Private Sub Slider2_Scroll()
Form1.MCIWnd1.Speed = Form1.Slider2.Value * 100
'改变动画播放中的速度
End Sub
Private Sub Slider3_Scroll()
Form1.MCIWnd1.Zoom = Form1.Slider3.Value * 100
'改变动画播放中的缩放比
End Sub
```

在程序运行的过程中，用户选择了一个有效的动画文件，窗体上的三个滚动条控件 Slider1、Slider2 和 Slider3 就会变为有效的状态，如果用户利用鼠标拖动滚动条控件上的滑动条，就会激活控件的 Private Sub Slider1_Scroll() 事件、Private Sub Slider2_Scroll() 事件和 Private Sub Slider3_Scroll() 事件，然后分别执行相应的 Form1.MCIWnd1.Volume = Form1.Slider1.Value * 100、Form1.MCIWnd1.Speed = Form1.Slider2.Value * 100 和 Form1.MCIWnd1.Zoom = Form1.Slider3.Value * 100 语句来改变动画播放中的音量、播放速度和文件显示的缩放比。

6. 运行程序

按照附后的源程序清单添加剩余的代码，设置项目的启动窗体为主窗体 Form1，存储文件，运行程序，程序运行结果如图 6-32 所示。

图 6-32 程序运行结果

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00

Object = "{288F1520-FAC4-11CE-B16F-00AA0060D93D}#1.0#0"; "MCIWNDX.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"

Begin Visual Basic.Form Form1
BorderStyle =   3
Caption =   "动画播放程序"
Moveable    =   0
ScaleHeight =   4335
ScaleWidth  =   6915
ShowInTaskbar   =   0
StartUpPosition =   2
   Begin Visual Basic.CommandButton Command1
Caption =   "打开文件"
   End
   Begin Visual Basic.Label Label3
Caption =   "缩放调节："
   End
   Begin Visual Basic.Label Label2
Caption =   "速度调节："
   End
   Begin Visual Basic.Label Label1
Caption =   "音量调节："
   End
   Begin MSComDlg.CommonDialog CommonDialog1
Left    =   6000
Top =   3480
   End
```

```
   Begin MCIWndX.MCIWnd MCIWnd1
Height  =   2895
Left    =   1320
Top =   480
Width   =   4215
   End
   Begin MSComctlLib.Slider Slider1
Left    =   5880
Top =   1320
Orientation =   1
   End
   Begin MSComctlLib.Slider Slider2
Left    =   360
Top =   1440
Orientation =   1
   End
   Begin MSComctlLib.Slider Slider3
Left    =   4560
Top =   3720
   End
End

Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False

Private Sub Command1_Click()
Form1.CommonDialog1.ShowOpen
'显示一个对话框
If Form1.CommonDialog1.FileName <> "" Then
   Form1.MCIWnd1.FileName = Form1.CommonDialog1.FileName
   '设置播放的动画文件
   Form1.Caption = DialogCaption + Form1.CommonDialog1.FileName
   '显示播放文件的路径和文件名
   Form1.Slider1.Enabled = True
   Form1.Slider2.Enabled = True
   Form1.Slider3.Enabled = True
   '设置滚动条控件的有效状态
Else
   Exit Sub
End If
End Sub

Private Sub Form_Load()
```

```
Form1.CommonDialog1.DialogTitle = "请选择一个有效的动画文件："
'设置对话框标题
Form1.CommonDialog1.Filter = "动画文件(*.avi)|*.avi"
'设置文件过滤器
Form1.CommonDialog1.InitDir = "c:\windows"
'设置缺省工作路径
Form1.Command1.Enabled = True
'设置按钮控件的有效状态
Form1.Slider1.Enabled = False
Form1.Slider2.Enabled = False
Form1.Slider3.Enabled = False
'设置滚动条控件的有效状态
End Sub

Private Sub Slider1_Scroll()
Form1.MCIWnd1.Volume = Form1.Slider1.Value * 100
'改变动画播放中的音量
End Sub

Private Sub Slider2_Scroll()
Form1.MCIWnd1.Speed = Form1.Slider2.Value * 100
'改变动画播放中的速度
End Sub

Private Sub Slider3_Scroll()
Form1.MCIWnd1.Zoom = Form1.Slider3.Value * 100
'改变动画播放中的缩放比
End Sub
```

6.5 小 结

图文并茂、生动活泼是多媒体程序的特点之一，在当今计算机领域中，多媒体技术已经成为计算机领域中发展最为迅速的领域之一，在应用程序中适当的插入一些动画，会使应用程序变得生动、活泼，所以多媒体应用程序设计中动画程序设计是很重要的一部分内容。

在本章中，通过几个示例程序向读者介绍 Visual Basic 6.0 中动画应用程序设计得一般方法和常用技巧，其中包括动画程序制作，动画播放器的制作等方面，希望读者能够在理解的基础上能够编制出自己的多媒体应用程序。

第七章 数据库开发和应用

Visual Basic 从初级的版本开始到现在的 6.0 版本，给与数据库的访问和操作以强有力的支持。特别是 6.0 版本在原有支持的 Access、FoxPro 和 Dbase 的基础上拓展了功能，对于目前最流行的 Oracle 和 SQL Sever 数据库以强有力的支持，与这两个大型数据库的接口更为方便。可以说，Visual Basic 中最吸引人的地方，同时也是人们最关心的地方，就是 Visual Basic 的强大的数据库开发功能。

本章，我们主要就 Visual Basic 6.0 中经典的数据库开发技术作一介绍，通过学习让读者掌握 Visual Basic 中高级的数据库开发知识。

7.1 VB 中的数据库概述

7.1.1 数据库概述

数据库应用程序分为三个部分：用户界面、数据库引擎和数据库。

用户界面包括一些应用程序代码，这部分是用来让用户访问数据库并与数据库交互的界面，显然该界面允许用户对于数据库进行查询、修改等操作。但是这些操作并不是用户界面直接向数据库进行交换，而是通过数据库引擎。

数据库引擎是用来对接用户界面要求的操作翻译成数据库能够理解的操作，而把数据库储存的资料介绍给用户界面。数据库是由包含多个文件和表的集合，该结合提供了一定顺序的大量资料。

下面介绍一下使用数据库常用的一些基本术语：

■ 表

表是一种按行和列排列的相关信息的逻辑组，类似于工作表单。例如一张表可以包含一个学生的和多信息，姓名、年龄大小等等。

■ 字段

数据库表中的每一列称作一个字段。表是由其包含的各种字段定义的，每个字段描述了它所包含的数据。创建一个数据库时，为每个字段分配一个数据类型、最大长度和其他属性。字段何以包含各种字符、数字甚至是图形。

■ 记录

各个不同的学生的资料存放在表的行，被称之为记录。一般来说表在创建时，任意两行不能相同。

■ 索引

为更快的访问数据库，大多数数据库都有索引。数据库表的索引是比表搜索更快的序列。

实质上，整个数据库的构造和使用即查询、修改是一个很大的工程。而且数据库的使用和类型很不一样，在这里只能做简单的介绍。

7.1.2 VB 与数据库

在本章中将重点放在应用程序界面上。不会对引擎和数据库做太多的介绍。

一般情况下，VB 对于数据库的访问可有以下几种方式：

■ 访问 VB 数据库

可以直接方便的访问 VB 数据库,该数据库主要是 Access 格式。

■ 访问外部数据库

可以方便的访问 ISAM 数据库。不仅可以方便的访问 DBase、FoxPro 和 Paradox 数据库，还可以方便的使用索引顺序方法来方便的访问类 Ascii 文本文件的数据库；

■ 访问 ODBC 数据库

可以以良好的接口性能实现对于远程大型数据库的访问，比如对于 Oracle 和 SQL Sever 远程数据库的访问和操作。

在设计过程中可以灵活的对以上几种方式的进行访问和操作。

7.2 第一个数据库程序

在 Visual Basic 6.0 中的缺省工具箱上有一个数据控件——Data 控件，同时 TextBox 控件和 Label 控件都是 Data 控件的束缚控件，在程序设计过程中，可以把这两个控件捆绑到 Data 控件上，这样，在程序运行的过程中，就可以利用 TextBox 控件和 Label 控件显示数据库中的数据。

下面就以一个示例程序来说明，如何在程序设计和运行的过程中，协调 TextBox 控件、Label 控件和 Data 控件来显示数据库中的内容。具体的程序设计步骤如下所示：

1. 开始工作

首先，在 Visual Basic 的集成开发环境中用鼠标选择“文件”菜单中“新建工程”选项，在屏幕上就会弹出一个如图 7-1 所示的“新建工程”对话框。

图 7-1 新建工程对话框

在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.Form Form1
BorderStyle      =     3
Caption     =     "数据库浏览程序："
MaxButton=     0
MinButton=     0
Moveable  =     0
ScaleHeight      =     4590
ScaleWidth=     6885
ShowInTaskbar  =     0
StartUpPosition  =     2
End
```

经过以上属性设置后的窗体具有如下所示的特性：

■ 程序运行过程中，窗体位于屏幕的中央，并且用户不能够移动窗体；

- 窗体的标题栏中显示文本“数据库浏览程序”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

在本示例程序中，由于要利用利用数据控件 Data 及其束缚控件（TextBox 控件和 Label 控件）进行数据显示操作，所以在程序设计的过程中要向空白的窗体上添加一个 Data 控件、四个 Label 控件和四个 TextBox 控件，它们的作用在后面会加以叙述，添加数据显示控件后的窗体如图 7-2 所示。

图 7-2 添加数据显示控件后的窗体

添加到窗体上的各个控件的属性设置如下所示：

```
    Begin Visual Basic.TextBox Text4
DataField  =     "Address"
DataSource=      "Data1"
Height     =     495
Left  =     2280
Top  =     1800
Width      =     3375
    End
    Begin Visual Basic.TextBox Text3
DataField  =     "Telephone"
DataSource=      "Data1"
Height     =     495
Left  =     2280
Top  =     2520
Width      =     3375
    End
    Begin Visual Basic.TextBox Text2
DataField  =     "Company
                                    Name"
DataSource=      "Data1"
Height     =     495
Left  =     2280
Top  =     1080
Width      =     3375
    End
    Begin Visual Basic.TextBox Text1
DataField  =     "Name"
DataSource=      "Data1"
```

```
Height      =     495
Left  =     2280
Top   =     360
Width       =     3375
    End
    Begin Visual Basic.Data Data1
Align=      2
Connect     =     "Access"
DatabaseName    =     "D:\MicrosoftVisualStudio\Visual Basic98\
                  Biblio.mdb"
ReadOnly  =     0
RecordsetType   =     1
RecordSource    =     "Publishers"
Visible     =     0
    End
    Begin Visual Basic.Label Label4
Caption     =     "公司地址："
    End
    Begin Visual Basic.Label Label3
Caption     =     "公司电话："
    End
    Begin Visual Basic.Label Label2
Caption     =     "公司全名："
    End
    Begin Visual Basic.Label Label1
Caption     =     "名称："
    End
```

添加到窗体上的各个控件的作用如下所示：

- Data **控件：在程序设计和运行的过程中，为用户和数据库中间提供一个接口，通过这个接口，用户可以对数据库进行操作；**
- Label **控件：在程序运行的过程中显示固定的提示文本“名称：”、“公司全名：”、“公司地址：”和“公司电话”；**
- Text1 **控件：在程序运行的过程中，显示指定数据库位置的名称信息；**
- Text2 **控件：在程序运行的过程中，显示指定数据库位置的公司名称信息；**
- Text3 **控件：在程序运行的过程中，显示指定数据库位置的公司电话信息；**
- Text4 **控件：在程序运行的过程中，显示指定数据库位置的公司地址信息。**

2．添加控制按钮

从前面数据控件的属性设置中我们可以看出，在程序运行的过程中，数据控件是处于不可见状态的，为了能够在数据控件隐藏的条件下也能够浏览数据库，就要向项目中添加控件按钮。

注意：

✍ 数据库控件是一种隐性控件，和我们在前面学习的可视化控件不同。

为此，在程序设计的过程中，向窗体上添加四个 CommandButton **控件，它们的属性设置如下所示：**

```
    Begin Visual Basic.CommandButton Command4
Caption     =     "最后一条记录"
Height      =     495
Left  =     3840
```

```
Top  =    3960
Width     =    1815
   End
   Begin Visual Basic.CommandButton Command3
Caption   =    "后一条记录"
Height    =    495
Left  =    960
Top  =    3960
Width     =    1815
   End
   Begin Visual Basic.CommandButton Command2
Caption   =    "前一条记录"
Height    =    495
Left  =    3840
Top  =    3240
Width     =    1815
   End
   Begin Visual Basic.CommandButton Command1
Caption   =    "第一条记录"
Height    =    495
Left  =    960
Top  =    3240
Width     =    1815
   End
```

添加控制按钮后的窗体如图 7-3 **所示。**

图 7-3　**添加按钮控件后的窗体**

添加到窗体上的四个控制按钮的作用如下所示：

- Command1 **按钮：在程序运行的过程中，显示当前数据库中的第一条记录；**
- Command2 **按钮：在程序运行的过程中，显示当前数据库中的前一条记录；**
- Command3 **按钮：在程序运行的过程中，显示当前数据库中的后一条记录；**
- Command4 **按钮：在程序运行的过程中，显示当前数据库中的最后一条记录。**

3．程序初始化

在本示例程序中，所谓程序的初始化，指的就是为窗体事件 Private Sub Form_Load()**添加响应代码。**

在程序设计的过程中，首先用鼠标双击窗体上的空白处，在屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择窗体 Form**，在对应的事件列表中选择窗体的初始化事件** Private Sub Form_Load()**，打开的代码编辑窗口如图** 7-4 **所示。**

图 7-4 代码编辑窗口

在代码编辑窗口把光标移动到 Private Sub Form_Load()**事件的处理过程中，并且添加如下所示的程序初始化代码：**

```
Private Sub Form_Load()
Form1.Data1.DatabaseName = "D:\Microsoft Visual Studio\Visual Basic98\Biblio.mdb"
'设置打开的数据库路径及名字
Form1.Data1.RecordSource = "Publishers"
'设置显示数据源
Form1.Text1.DataField = "name"
Form1.Text2.DataField = "company name"
Form1.Text3.DataField = "telephone"
Form1.Text4.DataField = "address"
'设置文本框显示字段名称
Form1.Command1.Enabled = True
Form1.Command2.Enabled = False
Form1.Command3.Enabled = True
Form1.Command4.Enabled = True
'设置控件的有效状态
End Sub
```

窗体 Private Sub Form_Load()**事件中的代码在对话框窗体被激活的初期就被执行，程序首先会通过语句** Form1.Data1.DatabaseName = "D:\Microsoft Visual Studio\Visual Basic98\ Biblio.m-db" **和** Form1.Data1.RecordSource = "Publishers"**设置了打开的数据库路径、名字及显示数据源，然后程序通过** Form1.Text1.DataField = "name"**等四条语句设置了文本框显示字段的名称。**

技巧：

✍ **在程序初始化的最后，通过四条语句设置了按钮的有效状态。**

经过程序初始化后的运行窗体如图 7-5 **所示。**

图 7-5 程序初始化后的运行窗体

4. 响应按钮操作

在窗体上放置有四个按钮控件，它们是在数据控件处于不可见的状态执行数据浏览操作，为此，在程序设计的过程中，首先用鼠标激活程序设计的代码窗口，在代码窗口的对象列表中分别选择按钮 Command1、Command2、Command3 和 Command4，在对应的事件列表中选择按钮的响应事件 Private Sub Command1_Click()、Private Sub Command2_Click()、Private Sub Command3_Click()和 Private Sub Command4_Click()，把光标移动到事件处理过程中，并且添加如下所示的响应代码：

```
Private Sub Command1_Click()
Form1.Data1.Recordset.MoveFirst
End Sub

Private Sub Command2_Click()
If Not Form1.Data1.Recordset.BOF Then
    Form1.Data1.Recordset.MovePrevious
    Else
    Form1.Data1.Recordset.MoveFirst
End If
End Sub
……
Private Sub Command4_Click()
Form1.Data1.Recordset.MoveLast
End Sub
```

在程序运行的过程中，当用户单击窗体上的“第一条记录”按钮、“前一条记录”按钮、“后一条记录”按钮 和“最后一条记录”按钮时，就会激活控件的 Private Sub Command1_Click()、Private Sub Command2_Click()、Private Sub Command3_Click()和 Private Sub Command4_Click()事件，程序就会通过数据控件 Recordset 方法中的 MoveFirst 方法、MovePrevious 方法、MoveNext 方法和 MoveLast 方法在当前的数据库中分别显示第一条记录、前一条记录、后一条记录和最后一条记录。

5. 响应数据控件事件

注意：

- 在窗体上放置有四个按钮，但是有两个按钮值得注意——“前一条记录”按钮和“后一条记录”按钮。

如果数据库的指针移动到数据库的开头或数据库的结尾，那么单击“前一条记录”按钮和“后一条记录”按钮时，程序就不能够正常的显示数据库中的内容。为了避免以上错误的发生，在程序设计的过程中，首先用鼠标激活程序设计的代码窗口，在代码窗口的对象列表中选择数据控件 Data1，在对应的事件列表中选择数据控

件响应事件 Private Sub Data1_Validate(Action As Integer, Save As Integer)，把光标移动到事件处理过程中，并且添加如下所示的响应代码：

```
Private Sub Data1_Validate(Action As Integer, Save As Integer)
If Form1.Data1.Recordset.BOF Then
    Form1.Command2.Enabled = False
    Else
    Form1.Command2.Enabled = True
End If
If Form1.Data1.Recordset.EOF Then
    Form1.Command3.Enabled = False
    Else
    Form1.Command3.Enabled = True
End If
End Sub
```

在程序运行的过程中，当用户完成一次对数据库的有效操作后，就会激活数据控件的 Private Sub Data1_Validate(Action As Integer, Save As Integer)事件，然后程序通过两条条件语句来设置“前一条记录”按钮和“后一条记录”按钮的有效状态，由于实行了动态跟踪的技术，所以避免了数据库显示错误的发生。

6. 运行程序

按照附后的源程序清单添加剩余的程序代码。

用鼠标选择菜单“文件”中的“保存”来存储文件，然后在键盘上按下功能键 F5 运行程序，程序运行结果如图 7-6 所示。

图 7-6 程序运行结果

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Begin Visual Basic.Form Form1
BorderStyle     =     3
Caption     =     "Form1"
ClientHeight     =     4590
ClientLeft  =     45
ClientTop  =     330
ClientWidth     =     6885
```

```
LinkTopic =     "Form1"
MaxButton=      0
MinButton =     0
Moveable  =     0
ScaleHeight     =       4590
ScaleWidth=     6885
ShowInTaskbar  =        0
StartUpPosition =       2
End

Attribute Visual Basic_Name = "Form1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = False
Attribute Visual Basic_PredeclaredId = True
Attribute Visual Basic_Exposed = False

Private Sub Command1_Click()
Form1.Data1.Recordset.MoveFirst
'移动到第一条记录
End Sub

Private Sub Command2_Click()
If Not Form1.Data1.Recordset.BOF Then
    '如果没有到达数据库开头
    Form1.Data1.Recordset.MovePrevious
    '移动到前一条记录
    Else
    '如果到达了数据库开头
    Form1.Data1.Recordset.MoveFirst
    '移动到第一条记录
End If
End Sub

Private Sub Command3_Click()
If Not Form1.Data1.Recordset.EOF Then
    '如果没有到达数据库结尾
    Form1.Data1.Recordset.MoveNext
    '移动到下一条记录
    Else
    '如果到达了数据库结尾
    Form1.Data1.Recordset.MoveLast
    '移动到最后一条记录
End If
End Sub

Private Sub Command4_Click()
```

```
Form1.Data1.Recordset.MoveLast
'移动到最后一条记录
End Sub

Private Sub Data1_Validate(Action As Integer, Save As Integer)
If Form1.Data1.Recordset.BOF Then
    Form1.Command2.Enabled = False
    Else
    Form1.Command2.Enabled = True
End If
'设置“前一条记录”按钮的有效状态
If Form1.Data1.Recordset.EOF Then
    Form1.Command3.Enabled = False
    Else
    Form1.Command3.Enabled = True
End If
'设置“后一条记录”按钮的有效状态
End Sub

Private Sub Form_Load()
Form1.Data1.DatabaseName = "D:\Microsoft Visual Studio\Visual Basic98\Biblio.mdb"
'设置打开的数据库路径及名字
Form1.Data1.RecordSource = "Publishers"
'设置显示数据源
Form1.Text1.DataField = "name"
Form1.Text2.DataField = "company name"
Form1.Text3.DataField = "telephone"
Form1.Text4.DataField = "address"
'设置文本框显示字段名称
Form1.Command1.Enabled = True
Form1.Command2.Enabled = False
Form1.Command3.Enabled = True
Form1.Command4.Enabled = True
'设置控件的有效状态
End Sub
```

7.3 数据库编辑和查询

通过使用 Visual Basic 6.0 的数据控件，不但可以打开并浏览已经存在的各种数据库文件，在显示数据的同时还可以对数据进行编辑和修改，但是它的一个不足就是单独的利用并不能够生成一个新的数据库。

7.3.1 数据库编辑

下面就是一个利用数据控件和文本框控件来制作数据库方面应用程序的示例，它的功能是打开一个已经存在的数据库，并且可以编辑其中的数据，实现此功能的程序的具体步骤如下所示：

1．开始工作

首先启动一个新的项目，在屏幕上就会出现一个空白的窗体，从工具栏上选取一个数据控件，并且把它放

置在空白的窗体上，其中数据控件的作用是为程序的运行提供一个数据库环境窗体，控件的属性设置如表 7-1 所示。

表 7-1 窗体与控件的属性设置

项 目	属 性	设 置
窗体	（Name）	Form1
	Caption	"Form1"
	ScaleHeight	2496
	ScaleWidth	3744
	BorderStyle	3 'Fixed Dialog
	StartUpPosition	2 'CenterScreen
Data	（Name）	Data1
	Align	2 'Align Bottom
	Caption	"数据控件与文本框的综合应用示例"
	Connect	"Access"
	DatabaseName	"D:\xiongao\test.mdb"
	DefaultType	2 'UseODBC
	Height	732
	Left	0
	ReadOnly	0 'False
	RecordsetType	1 'Dynaset
	RecordSource	"学生信息"
	Top	1764
	Width	3744

添加控件后的窗体如图 7-7 所示。

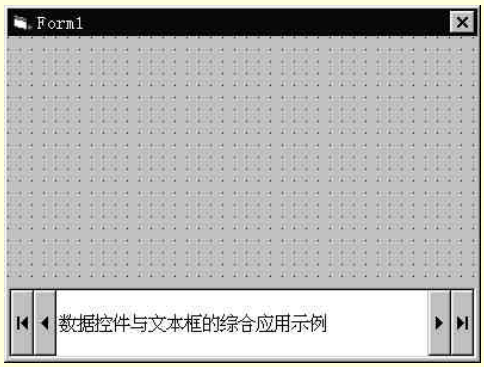


图 7-7 添加控件后的窗体

如表 7-1 设置的窗体具有如下的特性：

- 窗体位于屏幕的中央；
- 在程序的运行过程中窗体不能够改变大小；
- 在程序的运行过程中窗体不能够移动。

其中数据库 D:\test.mdb 的显示如图 7-8 所示：

图 7-8 数据库文件的内容

2. 添加显示控件

在窗体上添加三个 Label 控件和三个 TextBox 控件，其中 Label 控件的作用是显示字段名，而 TextBox 控件的作用是显示相应字段中的数据信息，添加控件后的窗体如图 7-9 所示。

图 7-9 添加控件后的窗体

到窗体上的控件的属性设置如表 7-2 所示。

表 7-2 控件的属性设置

控 件	属 性	设 置
控件 TextBox	（Name）	显示 1
	DataField	"学号"
	DataSource	"Data1"
	Height	492
	Left	1320
	Top	0
	Width	2412
控件 TextBox	（Name）	显示 2
	DataField	"性别"
	DataSource	"Data1"
	Height	492
	Left	1320
	Top	600
	Width	2412
控件 TextBox	（Name）	显示 3
	DataField	"姓名"
	DataSource	"Data1"
	Height	492
	Left	1320
	Top	1200

	Width	2412
控件 Label	（Name）	字段 1
	Height	372
	Left	120
	Top	0
	Width	972
控件 Label	（Name）	字段 2
	Height	372
	Left	120
	Top	600
	Width	972
控件 Label	（Name）	字段 3
	Height	372
	Left	120
	Top	1200
	Width	972

如表 7-2 设置的控件具有如下的特性：

- 数据控件始终位于窗体的下方；
- 当在程序的运行过程中，控件会自动的适应窗体的改变；
- 标题栏中显示字符串“数据控件与文本框的综合应用示例”；
- 打开的数据库类型为 Access；
- 数据库文件名称为“D:\xa\test.mdb”；
- 打开的数据库可读可写；
- 显示的数据库中表“学生信息”中的内容。

在打开的数据库文件 D:\test.mdb 中只有一个表，在这个表中有三个字段，它们分别是“学号”、“姓名”和“性别”，所以在本示例程序中设置了三个 Label 控件来显示这三个字段名，同时添加了三个 TextBox 控件来显示相应字段的数据。

注意：

> ✍ 在控件的属性设置中，TextBox 控件的 DataSource 属性一定要设置为相应的数据控件的名称，而且在 DataField 属性中指定要显示的字段名称。

3．添加代码

在程序的设计阶段双击窗体，就会弹出一个如图“ ”所示的空白代码窗口，在其中找到窗体的 Form_Load() 事件，并且在其中添加下列代码：

```
Private Sub Form_Load()
Data1.ReadOnly = True
'设置数据库处于只读的状态
Data1.Refresh
'打开数据库
字段 1.Caption = Data1.Database.TableDefs("学生信息").Fields(0).Name
字段 2.Caption = Data1.Database.TableDefs("学生信息").Fields(1).Name
字段 3.Caption = Data1.Database.TableDefs("学生信息").Fields(2).Name
'显示各个字段名称
End Sub
```

其中代码窗口如图 7-10 所示。

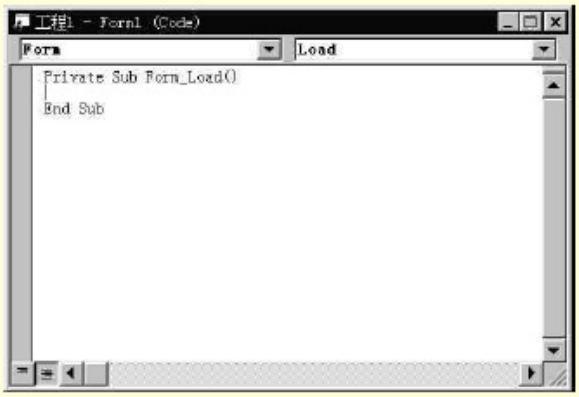


图 7-10 代码窗口

在程序开始运行时，马上执行窗体的 Form_Load()事件中的代码，首先设置数据库文件 D:\test.mdb 为只读的状态，然后通过语句 Data1.Refresh 最后通过三条语句来显示数据库的表中三个字段的名称。

提示：

- 程序段中一定要添加 Data1.Refresh 这一条语句，否则程序不能正常运行。

4. 运行程序

保存程序文件，在弹出的对话框中选择文件的路径和文件名进行保存，然后按键盘上的功能键 F5 运行程序。程序的运行结果如图 7-11 所示。

图 7-11 程序的运行结果

在程序的运行过程中，可以单击下列按钮：

：跳转到下一条记录；

：跳转到最后一条记录；

：跳转到前一条记录；

：跳转到第一条记录。

7.3.2 数据库查询

前面的程序虽然可以查看在数据库中的记录，同时如果把数据控件的 ReadOnly 属性设置为 False，就可以修改数据库中的记录，但是由于还不能够完成诸如查找、添加记录等功能，所以还只是一个简单的示例程序，下面就对这个程序做一下改进，使得它具有查找记录和显示指定记录的功能。

改进程序的具体步骤如下所示：

1. 改变数据控件的属性

首先要改变数据控件的属性设置，在前面的示例程序中，由于设置了数据控件的 ReadOnly 属性为 True，所以在程序的运行过程中用户不可以改变数据库中的数据，更改后的书库控件属性设置如图 7-12 所示。

图 7-12 数据控件的属性设置

提示：

- 由于控件的 ReadOnly 属性设置为 False，所以在程序的运行过程中用户可以更改记录。又因为数据控件的 Visible 属性设置为 False，所以在程序的运行过程中数据控件处于不可见的状态。

其中各个按钮所实现的功能如下所示：

- Command4 功能：跳转到最后一条记录；
- Command3 功能：跳转到上一条条记录；
- Command2 功能：跳转到下一条条记录；
- Command1 功能：跳转到第一条记录。

2. 添加控制按钮

向窗体上添加四个 CommandButton 控件，它们的作用是为显示数据控件所打开的数据库提供诸如“跳转到前一条记录”和“跳转到最后一条记录”等功能，控制按钮的属性设置如表 7-3 所示。

表 7-3 控制按钮的属性设置

控 件	属 性	设 置
控件 CommandButton	(Name)	Command4
	Caption	"最后一条记录"
	Height	492
	Left	2880
	Top	2640
	Width	1935
控件 CommandButton	(Name)	Command3
	Caption	"上一条条记录"
	Height	492
	Left	120
	Top	2640
	Width	1935
控件 CommandButton	(Name)	Command2
	Caption	"下一条条记录"
	Height	492
	Left	2880
	Top	1920
	Width	1935
控件 CommandButton	(Name)	Command1
	Caption	"第一条记录"
	Height	492
	Left	120
	Top	1920
	Width	1935

3. 添加控制代码

在程序的设计阶段，双击窗体中添加的控制按钮，在弹出的代码窗口中添加下列控制代码：

```
Private Sub Command1_Click()
Data1.Recordset.MoveFirst '跳转到第一条记录
```

```
End Sub

Private Sub Command2_Click()
If Not Data1.Recordset.EOF Then
Data1.Recordset.MoveNext
Else
Data1.Recordset.MoveLast
End If '跳转到下一条记录
End Sub

Private Sub Command3_Click()
If Not Data1.Recordset.BOF Then
Data1.Recordset.MovePrevious
Else
Data1.Recordset.MoveFirst
End If '跳转到上一条记录
End Sub

Private Sub Command4_Click()
Data1.Recordset.MoveLast '跳转到最后一条记录
End Sub
```

下面以控件 Command2 为例来说明添加代码的原理，首先在数据库中跳转到下一条记录的功能是依靠控件 Data1 中 Recordset 对象的 MoveNext 方法来实现的，但是在程序的运行过程中，如果数据库已经移动到了最后一条记录，但是用户仍然单击控件 Command2，就会产生一个错误，弹出如图 7-13 所示的对话框。

图 7-13 错误对话框

为了防止上述情况的出现，在控件 Command2 的 Command2_Click()事件中就要添加如下的错误处理代码：

```
If Not Data1.Recordset.EOF Then
    Data1.Recordset.MoveNext
Else
    Data1.Recordset.MoveLast
End If
```

如果数据库还没有移动到最后一条记录，那么就通过语句 Data1.Recordset.MoveNext 移动到数据库的下一条记录；反之，如果已经移动到了数据库的最后一条记录，那么通过语句 Data1.Recordset.MoveLast 来维持在数据库的最后一条记录上，其中数据库的结束标志存储在控件 Data1 的 Data1.Recordset.EOF 中。

说明：

✍ 其他几个控制按钮的代码添加过程在这里就不加赘述了，它们的思路与控件 Command2 大体相同。

4. 添加查找功能

为了实现在数据库中查找指定字符串的功能，首先要在窗体上添加一个 CommandButton 控件，利用它来激

活窗体的查找事件，然后在窗体上放置三个 OptionButton 控件，利用它们可以实现不同的查找功能。

添加控件后的窗体如图 7-14 所示。

图 7-14 添加控件后的窗体

其中控件的属性设置如表 7-4 所示。

表 7-4 控件的属性设置

控 件	属 性	设 置
控件 OptionButton	(Name)	Option1
	Caption	"按性别查找"
	Height	375
	Left	3600
	Top	1320
	Width	1335
控件 OptionButton	(Name)	Option1
	Caption	"按姓名查找"
	Height	375
	Index	1
	Left	3600
	Top	960
	Width	1335
控件 OptionButton	(Name)	Option1
	Caption	"按学号查找"
	Height	375
	Index	0
	Left	3600
	Top	600
	Value	-1 'True
	Width	1335
控件 CommandButton	(Name)	Command5
	Caption	"查 找"
	Height	495
	Left	3600
	Top	0
	Width	1215

控件 Command5 的功能就是为程序中激活查找过程提供一个控制，在程序中通过三个 OptionButton 控件来选择不同的查找方式，如“按学号查找”、“按姓名查找”等。

在控件的属性设置中要注意一点，就是三个 OptionButton 组成了一个控件数组，这样单击 OptionButton 控件时就可以通过它们的 Index 属性来判断用户选择的查找方式。

实现查找功能的代码如下所示：

```
Private Sub Command5_Click()
Dim s1 As String
Dim s2 As String
'定义两个字符串变量
```

```
s2 = InputBox("请输入查找内容")
'显示一个输入对话框
s1 = "[" + flag + "] Like '" + s2 + "'"
'为查找变量赋值
Data1.Recordset.FindFirst s1
'查找并且跳转到指定记录
End Sub
```

在程序的运行过程中，当用户用鼠标的左键单击控件 Command5 时，就会激活控件的 Command5_Click()事件。

程序首先定义了两个字符串型的变量 s1 和 s2，同时显示一个提示用户输入查找字符串的对话框，并且把它赋值给变量 s2，然后再为变量 s1（待查找的字符串）赋值，最后通过一条语句 Data1.Recordset.FindFirst s1 来查找用户指定的字符串并且会自动的跳转到指定的字符串位置处。

5. 选择查找方式

查找方式（包括：按学号查找、按性别查找和按姓名查找）的选择是通过三个 OptionButton 控件来完成的，而且控件选择状态的改变激活的事件为 Option1_Click(Index As Integer)，所以在其中添加下列代码：

```
Private Sub Option1_Click(Index As Integer)
Select Case Index
'判断用户的查找方式
Case 0
    flag = "学号"
Case 1
    flag = "性别"
Case 2
    flag = "姓名"
End Select
End Sub
```

6. 运行程序

在本示例应用程序中所调用的数据库文件的内容如图 7-15 所示，它由一个表——学生信息——所组成，在这个表中又有三个字段——学号、性别和姓名。

图 7-15 数据库文件的显示情况

选择菜单“文件”中的“保存...”项，存储文件，按键盘上的功能键 F5，运行程序，程序运行的初始画面如图 7-16 所示。

图 7-16 程序运行的初始画面

首先选择一种查找的方式，如“ 按学号查找 ”，然后单击“ 查找 ”按钮，就会弹出一个输入对话框，如图 7-17 所示。

图 7-17 输入对话框

在其中输入待查找的字符串，如 3502，单击 OK 按钮，在程序运行的窗体中就会显示出查找到的记录，如图 7-18 所示。

图 7-18 找到指定的记录

附程序的部分代码如下，供读者参考学习：

程序清单

```
Dim flag As String
'定义一个字符串型全局变量

Private Sub Command1_Click()
Data1.Recordset.MoveFirst
'跳转到第一条记录
End Sub

Private Sub Command2_Click()
If Not Data1.Recordset.EOF Then
```

```
Data1.Recordset.MoveNext
Else
Data1.Recordset.MoveLast
End If
'跳转到下一条记录
End Sub

Private Sub Command3_Click()
If Not Data1.Recordset.BOF Then
Data1.Recordset.MovePrevious
Else
Data1.Recordset.MoveFirst
End If
'跳转到上一条记录
End Sub
Private Sub Command4_Click()
Data1.Recordset.MoveLast
'跳转到最后一条记录
End Sub

Private Sub Command5_Click()
Dim s1 As String
Dim s2 As String
'定义两个字符串变量
s2 = InputBox("请输入查找内容")
'显示一个输入对话框
s1 = "[" + flag + "] Like '" + s2 + "'"
'为查找变量赋值
Data1.Recordset.FindFirst s1
'查找并且跳转到指定记录
End Sub

Private Sub Form_Load()
Data1.Refresh
'打开数据库
字段 1.Caption = Data1.Database.TableDefs("学生信息").Fields(0).Name
字段 2.Caption = Data1.Database.TableDefs("学生信息").Fields(1).Name
字段 3.Caption = Data1.Database.TableDefs("学生信息").Fields(2).Name
'显示各个字段名称
End Sub

Private Sub Option1_Click(Index As Integer)
Select Case Index
'判断用户的查找方式
Case 0
        flag = "学号"
```

```
Case 1
    flag = "性别"
Case 2
    flag = "姓名"
End Select
End Sub
```

7.4 网络数据库操作

前面的示例程序虽然能够浏览一个数据库中的数据信息，但是它只能够在本地机上浏览数据，随着网络技术的发展，经常需要开发一些支持网络操作的数据库程序，在 Internet 上发表数据等。为此，下面举了一个把数据库操作移植到网络上的应用程序示例，具体的程序设计步骤如下所示：

1．开始工作

首先，在 Visual Basic 的集成开发环境中用鼠标选择菜单“文件”中的“新建工程”选项，在屏幕上就会弹出一个如图 7-19 所示的“新建工程”对话框。

图 7-19 “新建工程”对话框

在“新建工程”对话框中选择“ ActiveX 文档 EXE”选项，单击 OK 按钮，在 Visual Basic 中就新建了一个 ActiveX 文档项目，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin Visual Basic.UserDocument UserDocument1
AutoRedraw      =       -1
ClientHeight    =       4995
ClientLeft      =       0
ClientTop       =       0
ClientWidth     =       7005
    BeginProperty Font
Name        =       "Times New Roman"
Size        =       10.5
Charset     =       0
Weight      =       400
Underline   =       0
Italic      =       0
Strikethrough   =       0
    EndProperty
HScrollSmallChange  =       255
KeyPreview      =       -1
```

```
ScaleHeight     =     4995
ScaleWidth=     7005
End
```

经过以上属性设置后的 ActiveX **文档设计窗口如图** 7-20 **所示。**

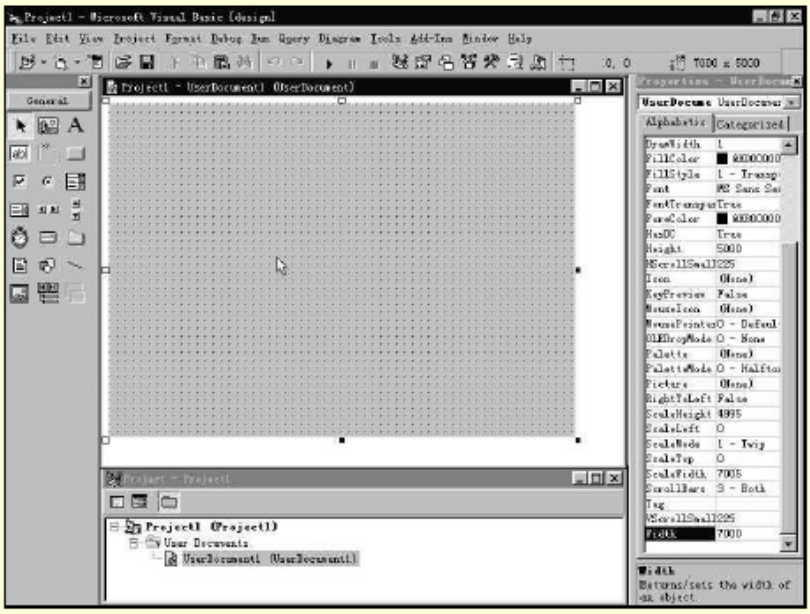


图 7-20　**属性设置后的** ActiveX **文档设计窗口**

2．**添加数据显示控件**

在本示例程序中，首先要利用利用数据控件 Data **及其** TextBox **控件和** Label **控件来进行数据库中的信息显示操作，所以在程序设计的过程中要向空白的窗体上添加一个** Data **控件、四个** Label **控件和四个** TextBox **控件，它们的作用在后面会加以叙述，添加到窗体上的各个控件的属性设置如下所示：**

```
    Begin Visual Basic.TextBox Text4
DataField  =     "Address"
DataSource=     "Data1"
Text  =     "Text4"
    End
    Begin Visual Basic.TextBox Text3
DataField  =     "Telephone"
DataSource=     "Data1"
Text  =     "Text3"
    End
    Begin Visual Basic.TextBox Text2

DataField="Company Name"
DataSource=     "Data1"
Text  =     "Text2"
    End
    Begin Visual Basic.TextBox Text1
DataField  =     "Name"
DataSource=     "Data1"
Text  =     "Text1"
    End
    Begin Visual Basic.Data Data1
```

```
Connect    =    "Access"
DatabaseName   = D:\Microsoft
                VisualStudio\Visual Basic98\Biblio.mdb"
ReadOnly  =    0
RecordsetType   =    1
RecordSource    =    "Publishers"
    End
    Begin Visual Basic.Label Label4
Caption    =    "公司地址："
    End
    Begin Visual Basic.Label Label3
Caption    =    "公司电话："
    End
    Begin Visual Basic.Label Label2
Caption    =    "公司全名："
    End
    Begin Visual Basic.Label Label1
Caption    =    "公司名称："
    End
```

添加到窗体上的各个控件的作用如下所示：

- Data **控件：在程序设计和运行的过程中，为用户和数据库中间提供一个接口，通过这个接口，用户可以对数据库进行操作；**
- Label **控件：在程序运行的过程中显示固定的提示文本“公司名称：”、“公司全名：”、“公司地址：”和“公司电话：”；**
- Text1 **控件：在程序运行的过程中，显示指定数据库位置的名称信息；**
- Text2 **控件：在程序运行的过程中，显示指定数据库位置的公司名称信息；**
- Text3 **控件：在程序运行的过程中，显示指定数据库位置的公司电话信息；**
- Text4 **控件：在程序运行的过程中，显示指定数据库位置的公司地址信息。**

添加了数据显示控件后的窗体如图 7-21 **所示。**

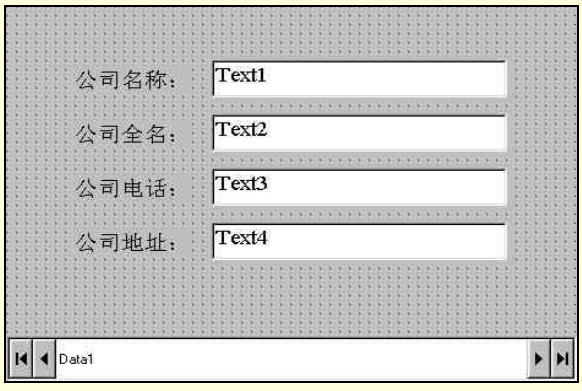


图 7-21　**添加数据显示控件后的窗体**

3．**添加控制按钮**

注意：

> **同前面的程序不同，在本示例程序运行的过程中，用户不但可以通过数据控件** Data1 **来浏览数据库中的信息，还可以通过控制按钮来进行添加信息、删除记录等操作。**

为此，在程序设计的过程中，向窗体上添加三个 CommandButton **控件，它们的属性设置如下所示：**

```
    Begin Visual Basic.CommandButton Command1
Caption     =     "保存数据"
    End
    Begin Visual Basic.CommandButton Command3
Caption     =     "删除数据"
    End
    Begin Visual Basic.CommandButton Command2
Caption     =     "追加数据"
    End
```

添加了控制按钮后的程序窗体如图 7-22 **所示。**

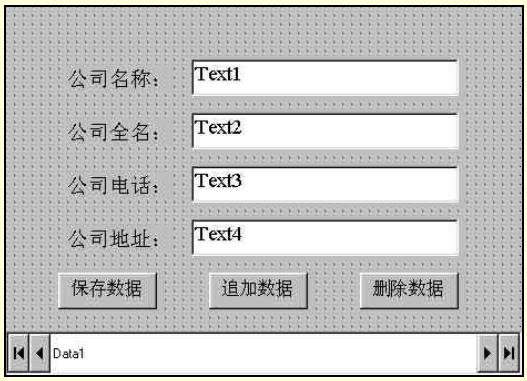


图 7-22　**添加控制按钮后的程序窗体**

三个控制按钮的作用如下所示：

- Command1 **按钮：在程序运行的过程中，把当前对数据库的修改操作存储起来，同时更新数据库的显示；**
- Command2 **按钮：在程序运行的过程中，向当前的数据末尾追加一条新的记录，同时更新数据库的显示；**
- Command1 **按钮：在程序运行的过程中，删除当前数据信息，同时更新数据库的显示。**

4. 程序初始化

在本示例程序中，所谓程序的初始化，指的就是为窗体事件 Private Sub Form_Load()**添加响应代码。**

在程序设计的过程中，首先用鼠标双击窗体上的空白处，在屏幕上就会弹出一个代码窗口，在代码窗口的对象列表中选择窗体 UserDocument**，在对应的事件列表中选择窗体的初始化事件** Private Sub UserDocument_Initialize()**，同时在代码编辑窗口把光标移动到** Private Sub UserDocument_Initialize()**事件的处理过程中，并且添加如下所示的程序初始化代码：**

```
Private Sub UserDocument_Initialize()
UserDocument.Data1.DatabaseName = "D:\Microsoft Visual Studio\Visual Basic98\Biblio.mdb"
'设置打开数据库的路径和文件名
UserDocument.Data1.RecordSource = "Publishers"
'设置显示数据源
UserDocument.Text1.DataField = "name"
UserDocument.Text2.DataField = "company name"
UserDocument.Text3.DataField = "telephone"
UserDocument.Text4.DataField = "address"
'设置文本框显示字段名称
UserDocument.Command1.Enabled = True
UserDocument.Command2.Enabled = True
UserDocument.Command3.Enabled = True
'设置按钮控件的有效状态
```

```
End Sub
```

窗体 Private Sub UserDocument_Initialize()事件中的代码在对话框窗体被激活的初期就被执行，程序首先会通过语句 UserDocument.Data1.DatabaseName = "D:\Microsoft Visual Studio\Visual Basic98\Biblio.mdb" 和 UserDocument.Data1.RecordSource = "Publishers"设置了打开的数据库路径、名字及显示数据源，然后程序通过 UserDocument.Text1.DataField = "name"等四条语句设置了文本框显示字段的名称。

提示：

✍ 在程序初始化的最后，通过四条语句设置了按钮的有效状态。

经过程序初始化后的运行窗体如图 7-23 所示。

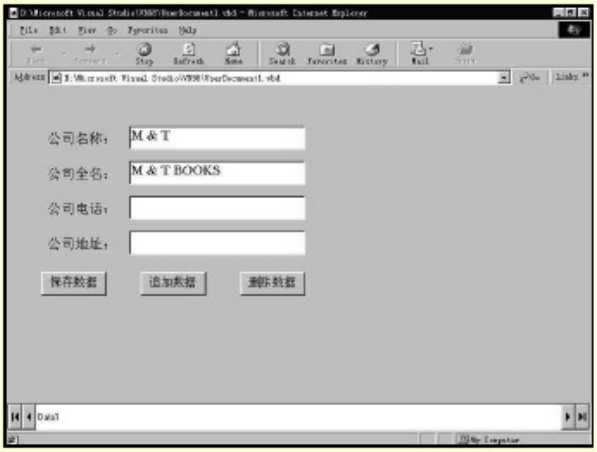


图 7-23 经过程序初始化后的运行窗体

5. 响应按钮事件

在窗体上放置有三个按钮控件，它们是在程序运行的过程中完成对数据库的修改操作。为此，在程序设计的过程中，首先用鼠标激活程序设计的代码窗口，在代码窗口的对象列表中分别选择按钮 Command1、Command2 和 Command3，在对应的事件列表中选择按钮的响应事件 Private Sub Command1_Click()、Private Sub Command2_Click()和 Private Sub Command3_Click()，把光标移动到事件处理过程中，并且添加如下所示的响应代码：

```
Private Sub Command1_Click()
UserDocument.Data1.Recordset.Update
End Sub
Private Sub Command2_Click()
UserDocument.Data1.Recordset.MoveLast
UserDocument.Data1.Recordset.AddNew
End Sub
Private Sub Command3_Click()
UserDocument.Data1.Recordset.Delete
UserDocument.Data1.Refresh
End Sub
```

在程序运行的过程中，当用户单击窗体上的“保存数据”按钮、“追加记录”按钮和“删除记录”按钮时，就会激活控件的 Private Sub Command1_Click()、Private Sub Command2_Click()和 Private Sub Command3_Click()事件，程序就会通过数据控件 Recordset 方法中的 Update 方法、AddNew 方法和 Delete 方法在当前的数据库中完成保存数据、追加记录和删除记录的操作。

6. 运行程序

按照附后的源程序清单添加剩余的程序代码。用鼠标选择菜单“文件”中的“保存”来存储文件，然后在键盘上按下功能键 F5 运行程序，程序运行结果如图 7-24 所示。

图 7-24　程序运行结果

附程序完整源代码如下所示：

程序清单

```
VERSION 6.00
Attribute Visual Basic_Name = "UserDocument1"
Attribute Visual Basic_GlobalNameSpace = False
Attribute Visual Basic_Creatable = True
Attribute Visual Basic_PredeclaredId = False
Attribute Visual Basic_Exposed = True

Option Explicit

Private Sub UserDocument_Initialize()
UserDocument.Data1.DatabaseName = "D:\Microsoft Visual Studio\Visual Basic98\Biblio.mdb"
'设置打开数据库的路径和文件名
UserDocument.Data1.RecordSource = "Publishers"
'设置显示数据源
Form1.Text1.DataField = "name"
Form1.Text2.DataField = "company name"
Form1.Text3.DataField = "telephone"
Form1.Text4.DataField = "address"
'设置文本框显示字段名称
UserDocument.Command1.Enabled = True
UserDocument.Command2.Enabled = True
UserDocument.Command3.Enabled = True
'设置按钮控件的有效状态
End Sub

Private Sub Command1_Click()
UserDocument.Data1.Recordset.Update
'保存修改结果
End Sub
```

```
Private Sub Command2_Click()
UserDocument.Data1.Recordset.MoveLast
'移动到数据库最后一条记录
UserDocument.Data1.Recordset.AddNew
'添加新记录
End Sub

Private Sub Command3_Click()
UserDocument.Data1.Recordset.Delete
'删除当前记录
UserDocument.Data1.Refresh
'刷新数据库
End Sub
```

7.5 小 结

随着软件编程技术的发展，Visual Basic 的数据库功能异军突起，在 Visual Basic 中可以独立的设计出数据库应用程序，同时完全不会依靠外部的数据库，但是它的功能却一点也不逊色。

本章我们主要介绍了数据管理器和数据控件的应用，同时还详细介绍了在 Visual Basic 中开发数据库应用程序的一般步骤。通过具体的示例学习仍然是本章学习的途径，相信读者在实践中能够独立开发出自己满意的程序。

另外，Visual Basic 数据库编程是一项庞大而复杂的课题，读者若想成为一个 Visual Basic 的数据库开发的大师，笔者建议你继续钻研学习以下方面的内容：

- Microsoft 的数据库控件以及内建的 Jet 数据库引擎技术；
- Jet 数据库引擎在 Visual Basic 中的对象模型和各种 DAO（Data Access Objects）；
- 通过 ODBC 驱动程序利用 ODBC 的 API 操作其他的外部数据库（External Database）；
- 深入学习 SQL 语言和 VBSQL 的扩展特性。

第八章 ActiveX 控件与网络应用

Visual Basic 的新增特性中，ActiveX 控件是很重要的一部分。ActiveX 控件通常用于针对网络的设计，所以也被称为网络控件。利用 ActiveX 控件，不但可以在各个应用程序之间建立一种标准的通信机制，而且通过 ActiveX 控件制作的 ActiveX 文档可以在 Internet 上发布。

从本质上说前面几章中介绍的 CommandButton 控件、TextBox 和 Timer 等控件都是 ActiveX 控件，在本章中所指的 ActiveX 控件主要是用户自己定制的 ActiveX 控件，同时还要向用户介绍 ActiveX 控件在网络上应用——ActiveX 文档和网络浏览器的制作等。

8.1 定制 ActiveX 控件

在本节中将向用户介绍如何在 Visual Basic 中定制一个符合自己需要的 ActiveX 控件，示例 Activex 控件可以把普通的 CommandButton 控件和 Timer 控件结合起来，即可以在程序运行的过程中通过设置 ActiveX 控件的 Interval 属性来激活计时器事件，具体的程序设计步骤如下所示。

1．开始工作

首先启动一个新的 Visual Basic 项目，在弹出的“新建工程”对话框中选择“标准 EXE”选项，如图 8-1 所示。

图 8-1 “新建工程”对话框

在“新建工程”对话框中选择“ActiveX 控件”选项，单击“确定”按钮，在 Visual Basic 就新建了一个 ActiveX 控件的工程文件，同时打开了一个空白的窗体，新建的 ActiveX 控件工程窗体如图 8-2 所示。

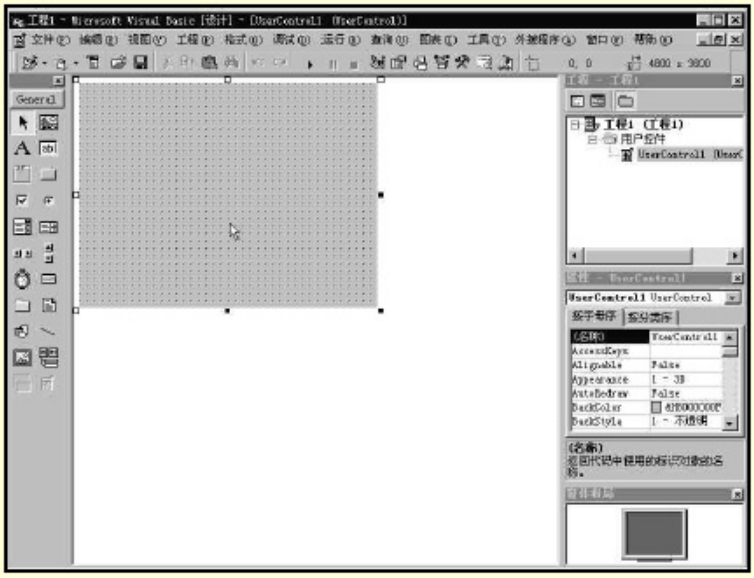


图 8-2 新建的 ActiveX 控件工程窗体

窗体的属性设置如表 8-1 所示：

表 8-1 窗体的属性设置

属性名	属性值
名称	UserControl1
ClientHeight	510
ClientLeft	0
ClientTop	0
ClientWidth	1950
ScaleHeight	510
ScaleWidth	1950

2. 添加控件

向空白的窗体上添加一个 Timer 控件和一个 CommandButton 控件，其中 CommandButton 控件用来接受 ActiveX 控件的各种操作（如鼠标、键盘等），而 Timer 控件则用来帮助 ActiveX 控件产生计时器事件，两个控件的属性设置如下代码所示。

```
   Begin Visual Basic .Timer Timer1
Interval    =   500
Left    =   1440
Top =   0
  End
   Begin Visual Basic .CommandButton Command1
Caption =   "Command1"
Height  =   495
Left    =   0
Style   =   1
TabIndex    =   0
Top =   0
Width   =   1935
  End
```

添加 CommandButton 控件和 Timer 控件后的窗体如图 8-3 所示。

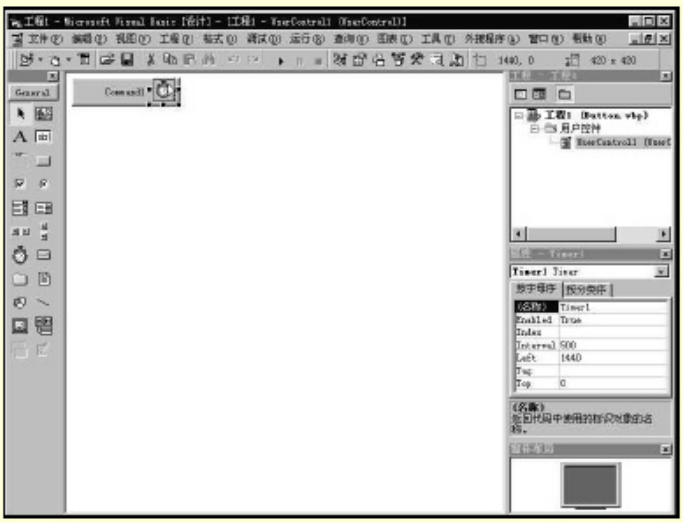


图 8-3 添加 CommandButton 控件和 Timer 控件后的窗体

3. 添加代码

在 ActiveX 控件设计的过程中，用鼠标左键双击窗体上的空白处，在弹出的代码窗口中把光标移动到事件 Private Sub UserControl_Initialize()的处理过程中，并且添加如下所示的 ActiveX 控件初始化代码。

```
Private Sub UserControl_Initialize()
   UserControl.Command1.Enabled = True
   UserControl.Timer1.Enabled = True
   '设置控件有效状态
   UserControl.Command1.Caption = Year(Date) & ":" & Month(Date) & ":" & Day(Date)
   '显示当前日期
   UserControl.Command1.Left = 0
   UserControl.Command1.Top = 0
   '初始化控件位置
   UserControl.Command1.Width = UserControl.Width
   UserControl.Command1.Height = UserControl.Height
   '初始化控件大小
End Sub
```

提示：

- 在 ActiveX 控件加载的初期，窗体 Private Sub UserControl_Initialize()事件中的代码就会被执行。

程序首先通过 UserControl.Command1.Enabled = True 和 UserControl.Timer1.Enabled= True 语句设置 ActiveX 控件中的两个子控件的有效状态，然后设置了 CommandButton 控件的显示文本为系统当前的日期，最后通过四条语句对 ActiveX 控件的大小和位置进行了设置。

经过以上初始化后的 ActiveX 控件测试结果如图 8-4 所示。

图 8-4 初始化设置后的 ActiveX 控件

注意：

- 在 ActiveX 控件的设计过程中，值得注意的一点就是要对控件的尺寸改变做出正确的判断和处理，由于 ActiveX 控件的设计都是依赖于一定的窗体（本示例程序中为 UserControl 窗体）。

所以要在窗体的 Private Sub UserControl_Resize()事件中添加对控件尺寸改变的响应代码（本程序中的响应代码如下所示）。

```
Private Sub UserControl_Resize()
   UserControl.Command1.Left = 0
   UserControl.Command1.Top = 0
   '设置控件位置
   UserControl.Command1.Width = UserControl.Width
   UserControl.Command1.Height = UserControl.Height
   '设置控件大小
End Sub
```

程序说明：

在 ActiveX 控件运行的过程中，如果 ActiveX 控件所在窗体的尺寸发生改变，就会激活 Private Sub UserControl_Resize（）事件，然后程序通过语句 UserControl.Command1.Left=0、UserControl.Command1.Top=0 来设置 ActiveX 控件的位置，之后再通过两条语句：

UserControl.Command1.Width=UserControl.Width

UserControl.Command1.Height=UserControl.Height

来设置 ActiveX 控件大小，这样，ActiveX 就能够实现跟踪尺寸改变的功能了。

另外为了能够在程序运行的过程中 ActiveX 控件能够自动的判断自身可见状态，需要添加如下所示代码。

```
Private Sub UserControl_Show()
MsgBox "ActiveX 控件处于可见状态"
End Sub
Private Sub UserControl_Hide()
MsgBox "ActiveX 控件处于不可见状态"
End Sub
```

4. 设计 ActiveX 控件

以上进行的工作是手动的，下面利用 Visual Basic 6 提供的 Visual Basic ActiveX 控件界面向导程序来生成基本的 ActiveX 外壳（如属性、事件和方法等）。

用鼠标选择菜单“工程”中的“添加用户控件”选项，就会弹出一个如图 8-5 所示的“添加用户控件”对话框。

图 8-5 “添加用户控件”对话框

在“添加用户控件”对话框中选择“Visual Basic ActiveX 控件界面向导”选项，单击“打开”按钮就会激

活 Visual Basic 6 的 ActiveX 控件界面向导，并且弹出一个如图 8-6 所示的“ ActiveX 控件接口向导-介绍 ”对话框。

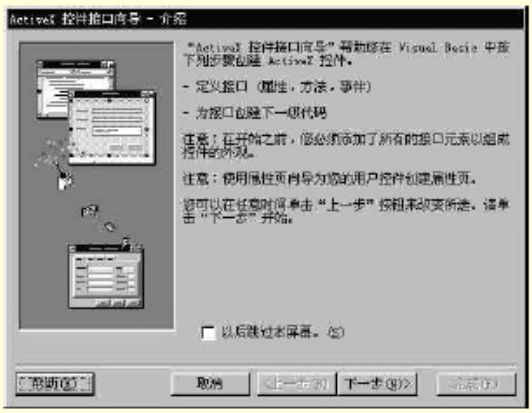


图 8-6 “ ActiveX 控件接口向导-介绍 ”对话框

“ ActiveX 控件接口向导-介绍 ”对话框中是 ActiveX 控件界面向导程序的的说明部分，用户可以通过复选“ 以后跳过本屏幕 ”选项来跳过这个窗口，并且向导程序以后都不会显示这个对话框。单击“ 下一步 ”按钮按钮，进入 ActiveX 控件界面向导程序的第二个窗口，如图 8-7 所示。

图 8-7 “ 选定接口成员 ”对话框

“ 选定接口成员 ”对话框列举了在定制 ActiveX 控件中可能会用到的属性、事件和方法，用户可以在其中选择定制 ActiveX 控件的属性，方法和事件。左边的“ 可用名称 ”列表框中列举了可以放入定制控件的标准属性，方法和事件，右边的“ 选定名称 ”列表框中列举了选定的定制控件的标准属性，方法和事件。

提示：

- 用户可以通过“ 选定接口成员 ”对话框中的四个按钮来完成常用属性、事件和方法的添加和删除操作。

在本示例程序中，在“ 可用名称 ”列表框中选择 Caption 选项、Interval 选项、Picture 选项、Timer 选项和 Style 选项，单击“ 下一步 ”按钮进入下一步，显示的“ 创建自定义接口成员 ”对话框如图 8-8 所示。

图 8-8 “创建自定义接口成员”对话框

在“创建自定义接口成员”对话框中用户可以为 ActiveX 控件添加自己的属性、方法和事件。

通过单击“新建”按钮，就会弹出如图 8-9 所示的“添加自定义成员”对话框，用户可以在“名称”输入框中输入自定义成员的名字，在“类型”中选择自定义成员的类型，如属性、方法和事件。

图 8-9 “添加自定义成员”对话框

在本示例程序中由于不会涉及到自定义属性、方法和事件，所以在“创建自定义接口成员”对话框中直接单击“下一步”按钮进入下一步。如图 8-10 所示。

图 8-10 “设置映射”对话框

在“设置映射”对话框中用户可以把定制 ActiveX 控件的属性、方法和事件映射到 Visual Basic 中已经存在的 ActiveX 控件的属性、方法和事件上。

从左边的“公有名称”列表框中选择一个或多个属性、方法或事件，然后在“控件”列表框中选择对应容器控件，在“成员”列表框中选择对应的接口成员即可。

例如在本示例程序中要把 ActiveX 控件的 Timer 属性映射到 Timer 控件的 Timer 成员上。

结束映射后，单击“下一步”按钮进入下一个窗口。如图 8-11 所示。

在“设置属性”对话框中，用户可以设置未映射成员的属性，如设置属性的数据类型、缺省值、运行/设计过程中的读写状态等，同时对未做改变的成员使用缺省值。

例如在本示例程序中，设置 Style 属性的数据类型为 Integer，缺省值为 0，运行和设计的过程中都是可读、可写的。属性设置完成后，单击“下一步”按钮进入下一个窗口。如图 8-12 所示。

在“已完成”对话框中用户可以选择是否查看 ActiveX 控件界面设计的总结报告，如果一切都准备就绪，单击“完成”按钮就完成了对 ActiveX 控件的界面设计。

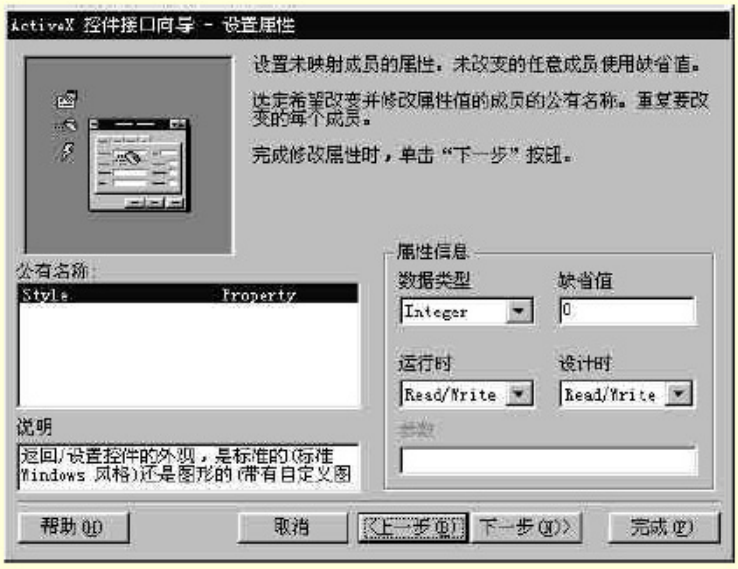


图 8-11 “设置属性”对话框

图 8-12 “已完成”对话框

5. 为 ActiveX 控件添加代码

按照以上的步骤完成对 ActiveX 控件的界面设计后，参考如下所示的程序清单在窗体 UserControl 中添加代码。

程序清单

```
Event Timer()  '事件声明:
 Event Click()
Event KeyDown(KeyCode As Integer, Shift As Integer)
```

```
Event KeyPress(KeyAscii As Integer)
Event KeyUp(KeyCode As Integer, Shift As Integer)
Event MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Event MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Event MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

Private Sub Command1_Click()
    RaiseEvent Click
End Sub

Private Sub Command1_KeyDown(KeyCode As Integer, Shift As Integer)
    RaiseEvent KeyDown(KeyCode, Shift)
End Sub

Private Sub Command1_KeyPress(KeyAscii As Integer)
    RaiseEvent KeyPress(KeyAscii)
End Sub

Private Sub Command1_KeyUp(KeyCode As Integer, Shift As Integer)
    RaiseEvent KeyUp(KeyCode, Shift)
End Sub

Private Sub Command1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    RaiseEvent MouseDown(Button, Shift, X, Y)
End Sub

Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    RaiseEvent MouseMove(Button, Shift, X, Y)
End Sub

Private Sub Command1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    RaiseEvent MouseUp(Button, Shift, X, Y)
End Sub

'从存贮器中加载属性值
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
    Command1.BackColor = PropBag.ReadProperty("BackColor", &H8000000F)
    Command1.Enabled = PropBag.ReadProperty("Enabled", True)
    Set Command1.Font = PropBag.ReadProperty("Font", Ambient.Font)
    Command1.Caption = PropBag.ReadProperty("Caption", "Command1")
    Set Picture = PropBag.ReadProperty("Picture", Nothing)
    Timer1.Interval = PropBag.ReadProperty("Interval", 500)
    Command1.BackColor = PropBag.ReadProperty("BackColor", &H8000000F)
    Command1.Enabled = PropBag.ReadProperty("Enabled", True)
    Set Picture = PropBag.ReadProperty("Picture", Nothing)
    Set Command1.Font = PropBag.ReadProperty("Font", Ambient.Font)
```

```
    Timer1.Interval = PropBag.ReadProperty("Interval", 500)
    Command1.Caption = PropBag.ReadProperty("Caption", "Command1")
End Sub

'将属性值写到存储器
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
    Call PropBag.WriteProperty("BackColor", Command1.BackColor, &H8000000F)
    Call PropBag.WriteProperty("Enabled", Command1.Enabled, True)
    Call PropBag.WriteProperty("Font", Command1.Font, Ambient.Font)
    Call PropBag.WriteProperty("Caption", Command1.Caption, "Command1")
    Call PropBag.WriteProperty("Picture", Picture, Nothing)
    Call PropBag.WriteProperty("Interval", Timer1.Interval, 500)
    Call PropBag.WriteProperty("BackColor", Command1.BackColor, &H8000000F)
    Call PropBag.WriteProperty("Enabled", Command1.Enabled, True)
    Call PropBag.WriteProperty("Picture", Picture, Nothing)
    Call PropBag.WriteProperty("Font", Command1.Font, Ambient.Font)
    Call PropBag.WriteProperty("Interval", Timer1.Interval, 500)
    Call PropBag.WriteProperty("Caption", Command1.Caption, "Command1")
End Sub

Private Sub Timer1_Timer()
    RaiseEvent Timer
End Sub

Public Property Get BackColor() As OLE_COLOR
    BackColor = Command1.BackColor
End Property

Public Property Let BackColor(ByVal New_BackColor As OLE_COLOR)
    Command1.BackColor() = New_BackColor
    PropertyChanged "BackColor"
End Property
Public Property Get Enabled() As Boolean
    Enabled = Command1.Enabled
End Property

Public Property Let Enabled(ByVal New_Enabled As Boolean)
    Command1.Enabled() = New_Enabled
    PropertyChanged "Enabled"
End Property

Public Property Get Picture() As Picture
    Set Picture = Command1.Picture
End Property

Public Property Set Picture(ByVal New_Picture As Picture)
```

```
    Set Command1.Picture = New_Picture
    PropertyChanged "Picture"
End Property

Public Property Get Font() As Font
    Set Font = Command1.Font
End Property

Public Property Set Font(ByVal New_Font As Font)
    Set Command1.Font = New_Font
    PropertyChanged "Font"
End Property

Public Sub Refresh()
    Command1.Refresh
End Sub

Public Property Get Interval() As Long
    Interval = Timer1.Interval
End Property

Public Property Let Interval(ByVal New_Interval As Long)
    Timer1.Interval() = New_Interval
    PropertyChanged "Interval"
End Property
Public Property Get Caption() As String
    Caption = Command1.Caption
End Property

Public Property Let Caption(ByVal New_Caption As String)
    Command1.Caption() = New_Caption
    PropertyChanged "Caption"
End Property
```

6. 利用 ActiveX 控件并运行

完成对以上工作的保存，然后用鼠标选择菜单“文件”中的“生成 OCX”选项，在 Visual Basic 集成开发环境中就会弹出一个询问路径和文件名称的对话框，在列表框中和“文件名”文本框中选择合适的路径和名称后，单击“保存”按钮，Visual Basic 就会自动的生成自定义的 ActiveX 控件。

重新启动 Visual Basic ，在屏幕上弹出的“新建工程”对话框中选择“标准 EXE”选项，单击“打开”按钮，启动一个标准的工程项目，同时打开了一个空白的窗体。选择菜单“工程”中的“部件”选项，在屏幕上就会弹出一个如图 8-13 所示的“部件”对话框。

图 8-13 “部件”对话框

单击“部件”对话框中的“浏览”按钮来选择刚才制作的 ActiveX 控件，单击“应用”按钮就可以把自定义的 ActiveX 控件加载到当前的项目文件中，并且在程序设计的过程中把自定义的 ActiveX 控件添加到窗体上，其中窗体和控件的属性设置如下所示。

```
Begin Visual Basic .Form Form1
Caption =   "Form1"
ClientTop   =   345
ClientWidth =   6225
ScaleHeight =   4170
ScaleWidth  =   6225
StartUpPosition =   3
   Begin 工程1.UserControl1 UserControl11
Height  =   615
Left    =   1320
Interval    =   500
Top =   720
Width   =   2775
Caption =   "1997:1:1"
   End
End
```

经过以上属性设置后的窗体如图 8-14 所示。

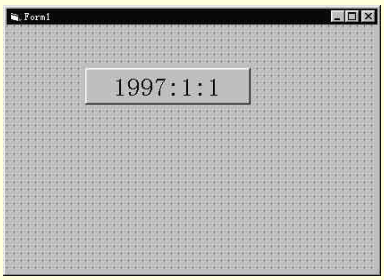


图 8-14 经过属性设置后的程序窗体

在程序设计的过程中，用鼠标双击窗体上的自定义 ActiveX 控件，在屏幕上就会弹出一个代码窗口，把光标移动到事件 Private Sub UserControl11_Timer()的处理过程中，并且添加如下所示的响应代码。

```
Private Sub UserControl11_Timer()
Form1.UserControl11.Caption = Hour(Time) & ":" & Minute(Time) & ":" & Second(Time)
'设置控件显示内容
End Sub
```

由于把自定义 ActiveX 控件的 Interval 属性设置为 500，所以每隔 500 微秒就会激活一个 Private Sub UserControl11_Timer()事件，程序通过代码 Form1.UserControl11.Caption = Hour(Time) & ":" & Minute(Time) & ":" & Second(Time)在自定义 ActiverX 控件上动态的显示系统当前的时间。

存储文件，运行程序，结果如图 8-15 所示。

图 8-15　程序运行结果

8.2　ActiveX 文档

以上我们介绍了自定义 ActiveX 控件的制作过程，其实作为 ActiveX 技术本身来讲，最吸引人的地方还是如何将 ActiveX 控件发布到网络上，即 ActiveX 文档技术。

其实在 Visual Basic 6 中制作在网络上发布的 ActiveX 文档并不是很困难的事情，下面就以示例程序来说明。

1．开始工作

首先启动一个新的 Visual Basic 6 项目，在屏幕上就会弹出一个“新建工程”对话框，在其中选择“ActiveX Document EXE”选项，在 Visual Basic 6 的集成开发环境中就会自动的启动一个 ActiveX 文档工程，如图 8-16 所示。

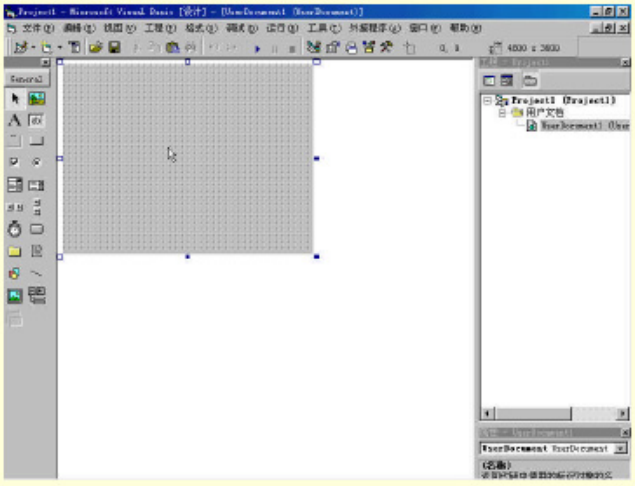


图 8-16　新建的 ActiveX 文档工程

在键盘上按下功能键[Ctrl+R]激活工程资源管理器，把系统默认的用户文档 UserDocument1 删除，然后选择菜单“工程”中的“添加窗体”选项，在屏幕上就会弹出一个如图 8-17 所示的对话框。

图 8-17 “添加窗体”对话框

在“添加窗体”对话框中选择“窗体”选项，单击“打开”按钮，系统就会把一个标准的空白窗体添加到当前的 ActiveX 文档工程中。

2. 添加控件

在程序设计的过程中，向当前空白的窗体上放置两个 Frame 控件、两个 CheckBox 控件、一个 TextBox 控件、三个 OptionButton 控件和一个 ListBox 控件，它们的属性设置如下所示。

```
Begin Visual Basic .Form Form1
StartUpPosition =   3
   Begin Visual Basic .Frame Frame2
Caption =   "字体名称："
      Begin Visual Basic .ListBox List1
Height  =   1860
Left    =   120
Top =   240
Width   =   2775
     End
   End
   Begin Visual Basic .Frame Frame1
Caption =   "字体风格："
      Begin Visual Basic .OptionButton Option1
Caption =   "斜体"
Index   =   2
     End
      Begin Visual Basic .OptionButton Option1
Caption =   "粗斜体"
Index   =   3
     End
      Begin Visual Basic .OptionButton Option1
Caption =   "粗体"
Index   =   1
     End
      Begin Visual Basic .CheckBox Check1
Caption =   "删除线"
Index   =   1
     End
      Begin Visual Basic .CheckBox Check1
Caption =   "下划线"
```

```
Index   =   0
     End
  End
  Begin Visual Basic .TextBox Text1
Text    =   "Text1"
  End
End
```

添加控件后的窗体如图 8-18 **所示。**

图 8-18 **添加控件后的窗体**

它们的作用如下：

- TextBox **控件：用来演示设置字体属性的各种效果；**
- ListBox **控件：在程序运行的过程中显示系统字体；**
- CheckBox **控件：为用户选择字体下划线和删除线的形式提供容器；**
- OptionButton **控件：为用户选择字体粗体和斜体的形式提供容器；**
- Frame **控件：充当控件容器。**

3. **添加代码**

用鼠标双击窗体，在弹出的代码窗口中选择程序初始化事件 Private Sub Form_Load()**，并且添加如下所示的响应代码。**

```
Private Sub Form_Load()
Form1.Text1.Font = "宋体"
Form1.Text1.FontBold = True
Form1.Text1.FontItalic = False
Form1.Text1.FontStrikethru = False
'初始化文本框中的字体
For i = 0 To Screen.FontCount - 1 Step 1
   Form1.List1.AddItem Screen.Fonts(i), i
Next
'向列表框中添加系统字体
End Sub
```

在程序运行的初期，Private Sub Form_Load()**事件中的代码就被执行，程序首先通过四条语句初始化文本框中的字体形式为宋体、黑体，并且没有下划线和删除线，然后通过一个循环语句向窗体中的列表框控件添加系统字体。**

经过以上初始化后的程序窗体如图 8-19 **所示。**

图 8-19 程序初始化后的窗体

窗体上的三个 OptionButton 控件组成一个控件数组，分别用来设置文本框中文本显示的粗体、斜体和粗斜体，为此，首先激活 Option1 控件数组，然后把光标移动到 Private Sub Option1_Click(Index As Integer)事件的处理过程中，并且添加如下所示的程序响应代码。

```
Private Sub Option1_Click(Index As Integer)
Select Case Index
      Case 1, 2
          Form1.Text1.FontBold = Form1.Option1(1).Value
          Form1.Text1.FontItalic = Form1.Option1(2).Value
          '设置粗体和斜体
      Case 3
          Form1.Text1.FontBold = True
          Form1.Text1.FontItalic = True
          '设置粗斜体
End Select
End Sub
```

这样，在程序运行的过程中，当用户用鼠标或键盘选中控件数组 Option1 中的任何一个控件时，都会激活 Private Sub Option1_Click(Index As Integer)事件，程序通过判断控件的 Index 属性就可以知道用户对字体风格的选择。如果选择的是粗体或斜体，都会执行代码 Form1.Text1.FontBold = Form1.Option1(1).Value 和 Form1.Text1.FontItalic = Form1.Option1 (2).Value 来分别设置文本框中文本显示为粗体或斜体，而当用户选择粗斜体时，就会执行 Form1.Text1.FontBold = True 和 Form1.Text1.FontItalic = True 来设置文本框中文本显示为粗斜体。

其他的设置字体风格和字体名称的代码的添加过程在这里就不再一一的加以叙述了，请读者参看下面的代码自己添加。

```
Private Sub Check1_Click(Index As Integer)
Form1.Text1.FontUnderline = Form1.Check1(0).Value
'设置下划线
Form1.Text1.FontStrikethru = Form1.Check1(1).Value
'设置删除线
End Sub

Private Sub List1_Click()
Form1.Text1.FontName = Form1.List1.Text
'对文本框应用选中字体
End Sub
```

4. 利用向导

以上我们所做的工作都是自己手动添加程序代码，但 Visual Basic 6 为了开发 ActiveX 文档的方便，特地制

作了一个“ VB ActiveX Document 向导 ”程序。

下面我们就来利用这个向导程序帮助我们来生成 ActiveX 文档。

首先选择菜单“ 工程 ”中的“ 添加用户文档 ”，在 Visual Basic 6 集成开发环境中就会弹出一个如图 8-20 所示的“ 添加用户文档 ”对话框。

图 8-20 “ 添加用户文档 ”对话框

在“ 添加用户文档 ”对话框中选择“ VB ActiveX Document 向导 ”选项，单击“ 打开 ”按钮，系统就会自动的向当前的项目文件中添加一个空白的 ActiveX 文档，同时启动“ VB ActiveX Document 向导 ”应用程序，弹出一个对话框。

在“ 介绍 ”对话框中显示有向导程序的简单介绍，同时在对话框中还有一个“ 以后跳过本屏幕 ”的复选框。

如果用户选中了这个复选框，那么以后再启动这个向导程序时就不会看到这个“ 介绍 ”对话框。

在“ 介绍 ”对话框中选中“ 以后跳过本屏幕 ”复选框，单击“ 下一步 ”按钮进入下一个窗口，将会显示一个如图 8-21 所示的对话框。

图 8-21 “ 选定窗体 ”对话框

在“ 选定窗体 ”对话框中，用户可以选择希望移植 ActiveX 文档的工程窗体，在本示例程序中，由于只有一个工程窗体 Form1，所以在“ 选定窗体 ”对话框中选择 Form1 选项，单击“ 下一步 ”按钮进入向导程序的下一步，将会显示如图 8-22 所示的对话框。

图 8-22 “选项”对话框

在“选项”对话框中用户可以对两个选项进行选择，一个是是否在添加 ActiveX 文档后取消无效代码的注释，另外一个就是是否在添加 ActiveX 文档后删除原始窗体。

注意：

- 在本示例程序中，选中“取消无效代码的注释”和“转换之后删除原始窗体”两个复选框，单击“下一步”按钮进入向导程序的下一步。

在“已完成”对话框中向导程序将会提示用户是否成功的移植 ActiveX 文档，同时用户在其中还可以选择是否将当前的设置作为缺省的设置加以保存。

在本示例程序中不复选“将当前设置作为缺省的加以保存”，同时还要选中“是”选项，单击“完成”按钮，就成功的完成了 ActiveX 文档的移植工作。

成功移植 ActiveX 文档后的工程窗体如图 8-23 所示。

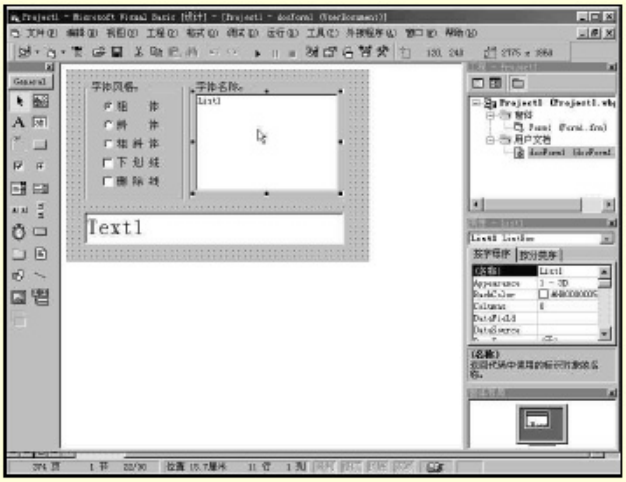


图 8-23 成功移植 ActiveX 文档后的工程窗体

5. 运行

做完以上的工作后，用鼠标选择工具栏上的“保存”按钮，在弹出的对话框中选择合适的文件存储路径和文件名，单击“保存”按钮完成文件的存储工作。

选择菜单“文件”中的“生成 Project1.EXE 选项”，在弹出的对话框中选择生成可执行文件的路径和文件名，单击“保存”按钮系统就生成了一个可以单独执行的 EXE 文件，同时在该目录下还会生成一个以*.vbd 为文件后缀的文件，利用 Internet Explorer 打开这个以*.vbd 为文件后缀的文件，结果如图 8-24 所示。

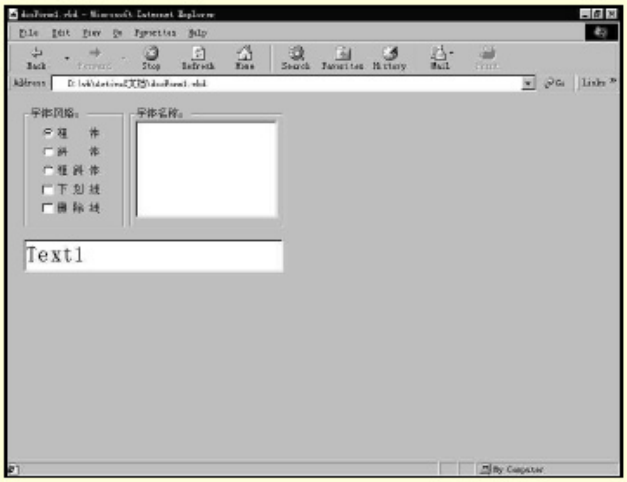


图 8-24 设计完成的 ActiveX 文档

8.3 网络浏览器的制作

现在的软件市场上有很多的网络浏览器，其中比较出色的是 Internet Explorer 和 Netscape，如图 8-25 所示即为浏览指定网页信息的 Internet Explorer 浏览器。

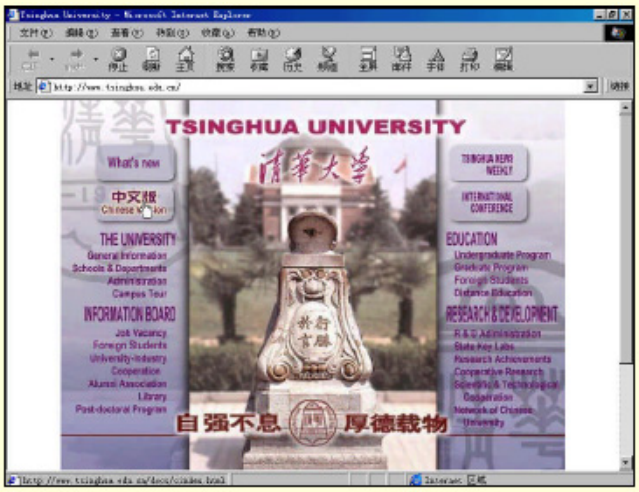


图 8-25 Internet Explorer 浏览器

在 Visual Basic 6 中，利用系统提供的 ActiveX 控件可以很轻松的制作出自己的网络浏览器，下面就以一个示例程序来说明，在 Visual Basic 6 中如何制作在 Internet 上可以遨游的网络浏览器，具体的程序设计步骤如下所示。

1．开始工作

首先启动一个新的 Visual Basic 6 项目，在弹出的“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 6 就新建了一个标准的工程文件，同时打开了一个空白的窗体，窗体的属性设置如表 8-2 所示。

表 8-2 窗体的属性设置

属性名	属性值
名称	Form1
BorderStyle	3
Caption	"网络浏览器"
ScaleHeight	4590
ScaleWidth	6885
ShowInTaskbar	0
StartUpPosition	2

经过以上属性设置后的窗体具有如下所示的特性：

- 程序运行过程中，窗体位于屏幕的中央；
- 窗体的标题栏中显示文本“网络浏览器”；
- 窗体的标题栏中没有最大化和最小化按钮，只有关闭按钮；
- 在程序运行的过程，用户不能够改变窗体的大小。

示例程序需要用到 WebBrowser 控件、ToolBar 控件和 ImageList 控件。为此，在程序设计的过程中，用鼠标左键单击菜单“工程”中的“部件”选项，在屏幕上就会弹出一个如图 8-26 所示的添加 ActiveX 的对话框。

图 8-26 “部件”对话框

在“部件”对话框中选择 Microsoft Internet Controls 和 Microsoft Windows Common Controls 5.0 (SP2)两个 ActiveX 控件，单击“确定”按钮，Visual Basic 6 就会自动的向当前的项目中添加程序设计所需要的 WebBrowser 控件、ToolBar 控件和 ImageList 控件。

2. 添加控件

在空白的窗体上添加一个 Label 控件、一个 ComboBox 控件、一个 Timer 控件、一个 WebBrowser 控件、一个 ToolBar 控件和一个 ImageList 控件，它们的属性设置如下所示。

```
   Begin ComctlLib.Toolbar tbToolBar
     ImageList      =   "imlIcons"
     BeginProperty Buttons {0713E452-850A-101B-AFC0-4210102A8DA7}
NumButtons  =   6
        BeginProperty Button1 {0713F354-850A-101B-AFC0-4210102A8DA7}
Key =   "Back"
Object.ToolTipText  =   "向后"
       EndProperty
        BeginProperty Button2 {0713F354-850A-101B-AFC0-4210102A8DA7}
Key =   "Forward"
Object.ToolTipText  =   "向前"
       EndProperty
        BeginProperty Button3 {0713F354-850A-101B-AFC0-4210102A8DA7}
```

```
Key =   "Stop"
Object.ToolTipText =   "停止"
        EndProperty
        BeginProperty Button4 {0713F354-850A-101B-AFC0-4210102A8DA7}
Key =   "Refresh"
Object.ToolTipText =   "刷新"
        EndProperty
        BeginProperty Button5 {0713F354-850A-101B-AFC0-4210102A8DA7}
Key =   "Home"
Object.ToolTipText =   "复位"
        EndProperty
        BeginProperty Button6 {0713F354-850A-101B-AFC0-4210102A8DA7}
Key =   "Search"
Object.ToolTipText =   "搜索"
        EndProperty
     EndProperty
  End

   Begin Visual Basic .ComboBox cboAddress
Text    =   "http://www.tsinghua.edu.cn"
  End

   Begin Visual Basic .Timer timTimer
Enabled =   0
Interval    =   50
  End

   Begin SHDocVwCtl.WebBrowser brwWebBrowser
Height  =   3735
Left    =   120
Top =   1320
Width   =   6000
Location    =   ""
  End

   Begin ComctlLib.ImageList imlIcons
      BeginProperty Images {0713E8C2-850A-101B-AFC0-4210102A8DA7}
NumListImages   =   6
        BeginProperty ListImage1 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
Picture =   "frmBrowser.frx":0000
Key =   ""
        EndProperty
        BeginProperty ListImage2 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
Picture =   "frmBrowser.frx":0692
Key =   ""
        EndProperty
```

```
        BeginProperty ListImage3 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
Picture =   "frmBrowser.frx":0D24
Key =   ""
        EndProperty
        BeginProperty ListImage4 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
Picture =   "frmBrowser.frx":13B6
Key =   ""
        EndProperty
        BeginProperty ListImage5 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
Picture =   "frmBrowser.frx":1A48
Key =   ""
        EndProperty
        BeginProperty ListImage6 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
Picture =   "frmBrowser.frx":20DA
Key =   ""
        EndProperty
     EndProperty
   End

   Begin Visual Basic .Label lblAddress
Caption =   "地址(&A):"
   End
```

经过以上属性设置后的程序窗体如图 8-27 **所示。**

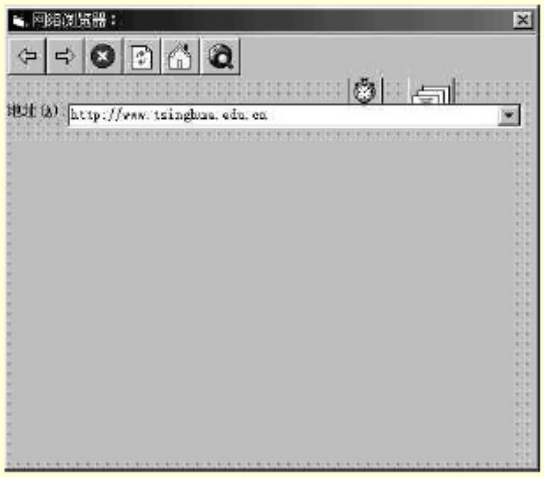


图 8-27　**设置属性后的程序窗体**

添加到窗体上的各个控件的作用如下所示：

- lblAddress **控件：显示固定的提示文本“ 地址 ”；**
- cboAddress **控件：用做用户输入地址的容器，并且可以存储网络地址；**
- tbToolBar **控件：在程序运行的过程中，通过单击其上的六个按钮就可以实现向前、向后、复位、刷新、停止和搜索等功能；**
- imlIcons **控件：用做存储图像的容器，在其中有工具栏按钮所需要的六幅图像；**
- brwWebBrowser **控件：在程序运行的过程中用来完成网络的具体操作，并且充当网页显示的容器；**
- timTimer **控件：充当计时器，每隔固定的时间就会激活一个计时器事件。**

3. **添加代码**

首先激活窗体的代码窗口，并且把光标移动到窗体的声明段中，添加如下所示的程序声明代码。

```
Public StartingAddress As String
```

在程序的声明段中，我们定义了一个全局变量 StartingAddress，它用来存储网络地址信息。然后把光标移动到窗体的 Private Sub Form_Load()事件中，添加如下所示的程序初始化代码。

```
Private Sub Form_Load()
    If Len(StartingAddress) > 0 Then
       cboAddress.Text = StartingAddress
       cboAddress.AddItem cboAddress.Text
       timTimer.Enabled = True
       brwWebBrowser.Navigate StartingAddress
    End If
End Sub
```

在程序运行的初期，窗体 Private Sub Form_Load()事件中的代码就会被执行，程序首先判断用户在地址输入文本框中给定的起始网络地址是否有效，如果有效，那么就开始读取网络信息（在本示例程序中，网络浏览器会浏览主页“ http://www.tsinghua.edu.cn ”的信息），然后设置计时器控件开始工作。

读取初始网页的浏览器如图 8-28 所示。

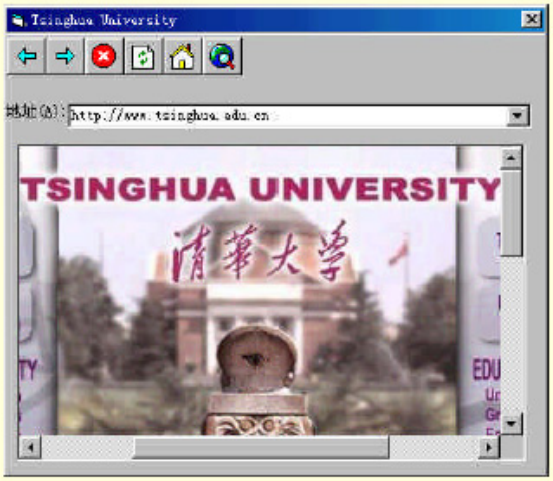


图 8-28 读取初始网页的浏览器

在窗体上添加的 Timer 控件的作用是充当计时器，在程序运行的过程中判断网络浏览器的工作状态，并且随时作出反应，为此，激活代码窗口后，把光标移动到事件 Private Sub timTimer_Timer()的处理过程中，并且添加如下所示的响应代码。

```
Private Sub timTimer_Timer()
If brwWebBrowser.Busy = False Then
   timTimer.Enabled = False
   '设置计时器有效状态
   Me.Caption = brwWebBrowser.LocationName
   '设置标题
Else
   Me.Caption = "运行中..."
   '设置标题
End If
End Sub
```

由于 Timer 控件的 Interval 属性设置为 50，所以在控件有效的情况下每隔 50 微秒系统就会激活一个 Private Sub timTimer_Timer()事件，如果网络浏览器正处于工作状态，那么就在窗体的标题栏中显示文本“ 运行中…”，否则就设置计时器控件处于无效状态，同时设置窗体的标题栏中显示当前网络的地址。

提示：

- 在本示例程序的具体制作过程中，除了以上的这些能够实现基本功能的代码外，还要添加一些必不可少的辅助代码。

由于篇幅的关系，在这里就不一一的加以叙述了，请读者参看下面的清单自己添加。

程序清单

```
Private Sub brwWebBrowser_DownloadComplete()
Me.Caption = brwWebBrowser.LocationName
'设置标题
End Sub

Private Sub brwWebBrowser_NavigateComplete(ByVal URL As String)
Dim i As Integer
Dim bFound As Boolean
Me.Caption = brwWebBrowser.LocationName
'设置标题
For i = 0 To cboAddress.ListCount - 1
    If cboAddress.List(i) = brwWebBrowser.LocationURL Then
      bFound = True
      Exit For
    End If
Next i
If bFound Then
   cboAddress.RemoveItem i
End If
cboAddress.AddItem brwWebBrowser.LocationURL, 0
cboAddress.ListIndex = 0
End Sub

Private Sub cboAddress_Click()
timTimer.Enabled = True
'设置计时器有效状态
brwWebBrowser.Navigate cboAddress.Text
'读取指定网络信息
End Sub

Private Sub cboAddress_KeyPress(KeyAscii As Integer)
If KeyAscii = vbKeyReturn Then
   cboAddress_Click
   '激活 Private Sub cboAddress_Click()事件
End If
End Sub

Private Sub tbToolBar_ButtonClick(ByVal Button As Button)
timTimer.Enabled = True
'设置计时器的有效状态
```

```
    Select Case Button.Key
          Case "Back"
              brwWebBrowser.GoBack
              '返回
          Case "Forward"
              brwWebBrowser.GoForward
              '前进
          Case "Refresh"
              brwWebBrowser.Refresh
              '刷新
          Case "Home"
              brwWebBrowser.GoHome
              '回到主页
          Case "Search"
              brwWebBrowser.GoSearch
              '查找
          Case "Stop"
              timTimer.Enabled = False
              '设置计时器有效状态
              brwWebBrowser.Stop
              '停止
              Me.Caption = brwWebBrowser.LocationName
              '设置程序标题
    End Select
    End Sub
```

4. 运行

作完以上的工作后，选择工具栏上的“保存”按钮来存储文件，然后在键盘上按下功能键 F5 运行程序，结果如图 8-29 所示。

图 8-29 程序运行结果

8.4 创建个人网页

Visual Basic 的一个显著特点就是网络功能的增强，不但体现在对 ActiveX 技术的支持上，还表现在直接的网络程序设计上，比如，利用 Visual Basic 可以很轻松的制作网页，而且可以存储为 HTML 格式。

下面就以一个示例程序来说明，利用 Visual Basic 6 如何来创建一个个人网站，具体的程序设计步骤如下所示：

1. 开始工作

首先激活 Visual Basic 6 应用程序，在 Visual Basic 6 的集成开发环境中就会弹出一个如图 8-30 所示的 New Project 对话框。

图 8-30 New Project 对话框

在 New Project 对话框中选择“ DHTML 应用程序 ”选项。

单击“ 打开 ”按钮，在 Visual Basic 6 中就新建了一个网络应用程序项目，同时打开了一个空白的窗体。

窗体的属性设置如下所示：

```
Begin {90290CCD-F27D-11D0-8031-00C04FB6C701} DHTMLPage1
AsyncLoad   =   0
id  =   "DHTMLPage1"
ShowBorder  =   -1
ShowDetail  =   -1
AbsPos  =   0
HTMLDocument    =   "DHTMLPage1.dsx":0000
End
```

2. 添加文本和控件

如果读者有过网页开发经验的话，那么对以下的步骤就不会陌生了，在 Visual Basic 6 中开发网页与普通的应用程序设计没有什么两样，都是添加控件，然后设置控件属性。

如果读者没有什么特别的要求，一般在 Visual Basic 中设计网页不用另外添加程序代码。

提示：

- 在本示例程序中，要建立一个个人网站，所以要向窗体上添加所需的图像和文本资料。

为此，在程序设计的过程中，向窗体上添加几行简单的文本、一个图像显示控件和四个超级连接控件。

添加控件和文本后的窗体如图 8-31 所示。

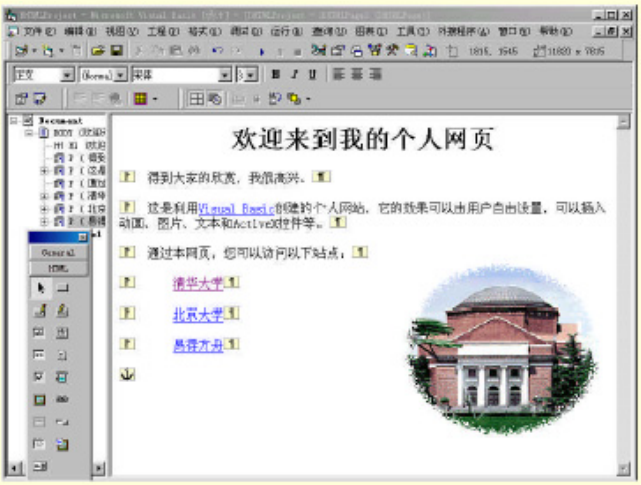


图 8-31　添加控件和文本后的网页设计窗体

在窗体上添加的四个超级连接控件的属性设置如下所示：

```
 (Id)   Hyperlink1
host    =home.microsoft.com:8
hostname    =home.microsoft.com
href    =http://home.microsof.com
name    =Hyperlink1
port    =80
scrollLeft  =0
scrollTop   =0
tabIndex    =0
(Id)    Hyperlink2
host    =www.tsinghua.edu.cn:80
hostname    =www.tsinghua.edu.cn
href    =http://www.tsinghua.edu.cn
name    =Hyperlink2
port    =80
scrollLeft  =0
scrollTop   =0
tabIndex    =0
(Id)    Hyperlink3
href    =http:
name    =Hyperlink3
pathname    =http:\\www.pku.edu.cn
port    =0
scrollLeft  =0
scrollTop   =0
tabIndex    =0
(Id)    Hyperlink4
host    =www.fanso.con:
hostname    =www.fanso.com
href    =http://www.fanso.com
name    =Hyperlink4
port    =80
```

```
scrollLeft  =0
scrollTop   =0
tabIndex    =0
```

超级连接控件的属性用户可以通过单击属性列表框中 Custom 栏右侧的 按钮来自定义设置，超级连接控件的自定义属性设置对话框如图 8-32 所示。

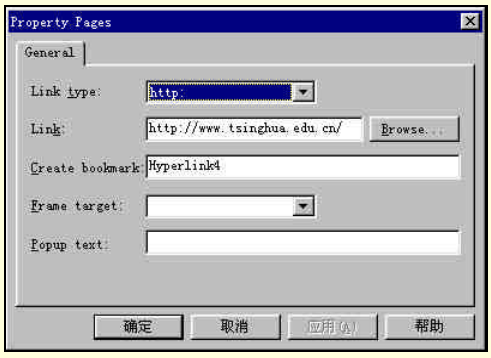


图 8-32 超级连接控件的自定义属性设置对话框

在窗体上添加的图像显示控件的属性设置如下所示：

```
 (Id)   Image1
border  =0
height  =165
hspace  =0
isMap   =False
loop    =1
name    =Image1
scrollLeft  =0
scrollTop   =0
src =file:///c:/my document/
Start   =fileopen
tabIndex    =0
vspace  =0
width   =280
```

同样，图像显示控件的属性用户可以通过单击属性列表框中 Custom 栏右侧的 按钮来自定义设置，图像显示控件的自定义属性设置对话框如图 8-33 所示。

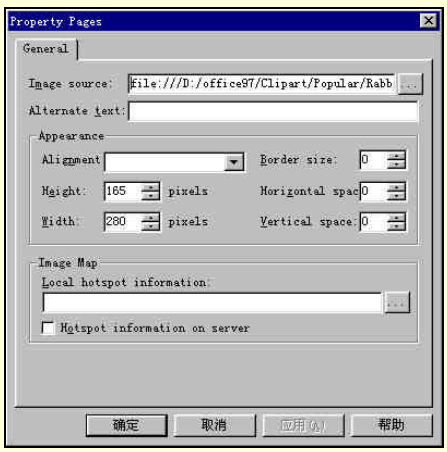


图 8-33 图像显示控件的自定义属性设置对话框

3. 启动 Internet Explorer

完成以上的工作后，就可以存储文件，在键盘上按下功能键 F5 运行程序了，系统会自动的启动 Internet Explorer 来浏览设计完成的网页，其中网页的源代码如下所示：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META content="text/html; " http-equiv=Content-Type>
<META content="MSHTML 5.00.2614.3500" name=GENERATOR></HEAD>
<BODY>
<!--METADATA TYPE="MsHtmlPageDesigner" startspan--><object id="DHTMLPage1"
classid="clsid:B302A0E7-CCF3-11D4-853F-0000E83DE614"width=0 height=0></object>
<!--METADATA TYPE="MsHtmlPageDesigner" endspan-->
<H1 align=center>欢迎来到我的个人网页
</H1>
<P align=left>  得到大家的欣赏，我很高兴。
</P>
<P align=left>  这是利用<A href="http://www.microsoft.com" id=Hyperlink1
name=Hyperlink1>Visual Basic
</A>创建的个人网站，它的效果可以由用户自由设置，可以插入动画、图片、文本和 ActiveX 控件等。</P>
<P align=left>  通过本网页，您可以访问以下站点：
</P>
<P align=left>nbsp;   <A href="http://www.tsinghua.edu.cn" id=Hyperlink2
name=Hyperlink2>清华大学
</A></P>
<P>      <A href="http://www.pku.edu.cn" id=Hyperlink3
name=Hyperlink3>北京大学</A></P>
<P>      
<A href="http://www.fanso.com" id=Hyperlink4 name=Hyperlink4>易得方舟</A></P>
<IMG alt="" border=0 height=169 hspace=0 id=Image1 name=Image1 src="file://C:\My
Documents\hall.gif" style="HEIGHT: 215px; LEFT: 368px; POSITION: absolute; TOP: 190px; WIDTH: 247px"
useMap="" width=247>
</BODY>
</HTML>
```

程序运行过程中浏览的网页如图 8-34 所示。

图 8-34 程序运行过程中浏览的网页

在程序运行过程中，用户可以通过单击网页上的四个超级连接来访问相应的站点。

比如单击“清华大学”链接，Internet Explorer 就会根据用户程序的设置访问清华大学的网站了，如图 8-35 所示。

图 8-35　在程序运行中访问“清华大学”网页

8.5　小　结

网络编程功能的增强是 Visual Basic 6.0 最吸引人的地方，为了方便用户开发的方便，Visual Basic 提供了几个 ActiveX 控件，如 Data 控件、WebBrowser 控件等，同时在 Visual Basic 中还有数据束缚控件和网络束缚控件，利用这些控件，可以大大的减少程序设计的工作量。

最后，我们还向读者介绍了 VBA 技术的简单应用。在 VBA 的程序开发环境中，利用 ActiveX 控件可以把 Office 作为（我们以 PowerPoint 为例）为 VB 程序设计的辅助工具；如同 VB 程序设计中的窗体一样，还可以将 ActiveX 控件添加 Office 设计环境中的文稿上，并且可以设置控件的属性、添加响应控件事件的代码。

第九章 API 函数的应用

API 就是 Application Programming Interface（应用程序接口）的英文缩写，API 是建立在 Windows 操作系统内部的 1000 多个函数的集合。

在本章中，我们将要学习如何在 Visual Basic 中调用 Windows 的 API 函数来实现对文字、图形、图像和多媒体的处理。

9.1 API 系统注册程序

用户平时经常可以看到 Windows 登录注册程序，比如进入 Windows NT 时的用户注册登录。为了了解 API，我们利用 Windows 提供的 API 函数制作一个系统简易的系统登录程序。它所实现的功能是，如果用户不能够输入正确的用户名和密码密码，就会被拒绝进入 Windows 操作系统，如果用户强行关闭登录程序，就会同时关闭计算机，所以就保护了数据的安全。

在这里就给出一个在 Visual Basic 通过调用 API 函数制作 Windows 登录程序的示例，具体的步骤如下：

1. 启动新的项目

在屏幕上会出现一个空白的窗体。但是在 Visual Basic 中自己提供了一个用于制作登录程序的窗体，所以首先删除空白的窗体，选择菜单“工程”/“删除 Form1”来删除窗体 1，然后选择菜单“工程”/“添加窗体”，就会弹出添加窗体对话框，选择其中的 Log in Dialog，如图 9-1 所示。

图 9-1 Log in Dialog 图标

单击 OK 按钮后的窗体如图 9-2 所示。

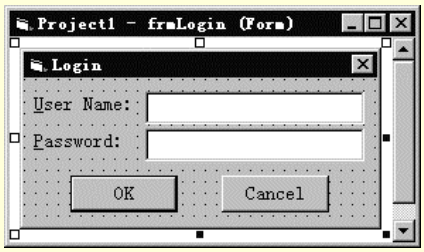


图 9-2 Log in Dialog 窗体

在这里系统已经自动的为我们生成了程序的框架，只要在上面添加一些代码和少量的控件就可以实现登录程序的功能。窗体的属性设置如表 9-1 所示。

表 9-1 窗体的属性设置

属 性	设 置
(Name)	FrmLogin
BorderStyle	3 - Fixed Dialog
Caption	Login
Enabled	True
Height	1920
Left	2790

Moveable	False
Picture	(None)
StartUpPosition	2 - CenterScreen
Top	3150
Visible	True
Width	3840

2．添加 Timer 控件

为了保证登录窗口始终位于其他窗口的前面，需要定时检测登录窗体的位置，如果有别的窗口在它的前面，那么就可以调用 API 函数改变各窗口的相对位置。

首先在窗体上放置一个 Timer 控件，它的属性设置如表 9-2 所示。

表 9-2 Timer 控件属性设置

属 性	设 置
(Name)	Timer1
Enabled	True
Interval	1000
Left	0
Top	567.2

提示：

✍ Interval 属性设置为 1000，也就意味着每个 1 秒钟就会激活一次 Timer1_Timer()事件。

除了 Timer 空间之外，为了改变各窗口的相对位置，还需要调用 SetWindowPos()函数，它的声明如下：

```
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
```

函数的参数及其说明如表 9-3 所示。

表 9-3 函数的参数及说明

参 数	说 明
Hwnd	窗口的句柄
HWndInsertAfter	指定各个窗口的顺序
X	窗口的左上角的 x 坐标
Y	窗口的左上角的 y 坐标
Cx	窗口的新宽度
Cy	窗口的新高度
WFlags	影响窗口位置和大小的一个参数

在 Timer1_Timer()事件中添加下列代码：

```
Private Sub Timer1_Timer()
'每隔 1 秒钟触发一次
SetWindowPos frmLogin.hwnd, HWND_TOP, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE
'登录位于其他窗体的前面
frmLogin.SetFocus
End Sub
```

HWndInsertAfter 参数设置为 HWND_TOP，表示 frmLogin 窗口位于所有窗体的前面，但是还要在程序的声明段中添加定义 HWND_TOP 常量的代码：

```
Private Const HWND_TOP = 0
```

WFlags 参数设置为 SWP_NOMOVE Or SWP_NOSIZE，它的意义是不改变原有窗体的大小和位置，所以就屏蔽了 x，y，cx 和 cy 参数。同样也要在程序的声明段中添加定义 SWP_NOMOVE 和 SWP_NOSIZE 常量的代码：

```
Private Const SWP_NOSIZE = &H1
Private Const SWP_NOMOVE = &H2
```

完成之后的窗体如图 9-3 所示。

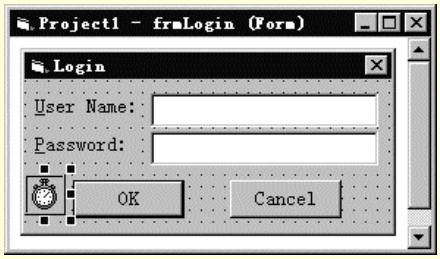


图 9-3 添加了 Timer 控件的窗体

注意：

- API 函数声明的所有语句都要写在同一行上。

3. 为 OK 按钮添加代码

用户输入用户名和密码后，如果密码不正确，程序就会给出提示用户重新输入密码的信息；如果密码正确，结束程序的运行，实现以上功能的代码如下：

```
Private Sub cmdOK_Click()
    If txtPassword = "password" Then
       '如果用户输入的密码正确
       End
       '退出程序的运行
    Else
       '如果用户输入的密码不正确
       MsgBox "Invalid Password, try again!", , "Login"
       '显示一个对话框
    End If
End Sub
```

其中 MsgBox“ Invalid Password，try again! ”，“ Login ”用于显示一个对话框，对话框中的信息为字符串“ Invalid Password，try again! ”，并且对话框的标题栏中的文字为“ Login ”。

4. 为 Cancel 控件添加代码

不论用户输入用户名和密码与否，单击 Cancel 按钮，程序都会给出一个提示的信息，询问是否关闭计算机，如果用户不想关闭计算机，仍然可以接受用户的输入，否则就会关闭计算机，代码如下：

```
Private Sub cmdCancel_Click()
    '用户单击 Cancel 按钮时触发
    Dim r As Integer
    r = MsgBox("Are you want to shoutdown?", 1 + 32 + 4096, "ShounDown")
    '显示一个对话框，提示是否关闭计算机
    If r = 1 Then
       '如果用户在对话框中单击“ 确定 ”按钮
       ExitWindowsEx 1, 66
       '关闭计算机
    End If
    txtUserName.SetFocus
End Sub
```

其中 r = MsgBox("Are you want to shoutdown?", 1 + 32 + 4096, "ShounDown")这条语句用于显示一个询问用户是否关闭计算机的对话框，对话框中有“ 确定 ”和“ 取消 ”两个按钮，标题栏中显示的文字为字符串“ ShounDown ”，并且全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。

变量 r 用于存储用户的反应，如果 r=1，表示用户在对话框中单击“ 确定 ”按钮。

在这里还要用到一个 API 函数——ExitWindowsEx()，用于退出 Windows 系统，同时关闭计算机，它的声明

如下：

```
Private Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long, ByVal dwReserved As Long) As Long
```

执行下面的语句就可以实现关闭计算机的功能：

```
ExitWindowsEx 1, 66
```

5. 处理异常情况

如果用户在没有输入正确的用户名和密码的情况下而强行关闭登录应用程序，那么就要屏蔽用户的操作，同时关闭系统，如下列代码：

```
Private Sub Form_Unload(Cancel As Integer)
ExitWindowsEx 1, 66 '关闭计算机
End Sub
```

6. 运行

保存项目文件，并且把它编译成 d:\vbstudio\登录.exe 文件。为了在启动 Windows 的同时开始登录，需要把“登录.exe”文件添加到启动程序组中，打开 c:\pwin98\win.ini 文件，在[Windows]项中的“run=”语句后面添加“d:\vbstudio\登录.exe 文件”，如图 9-4 所示。

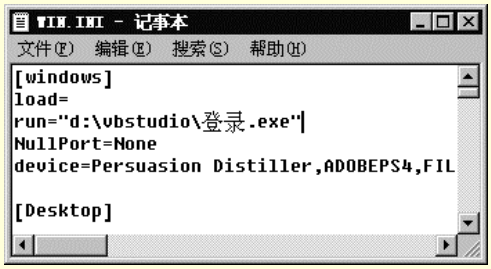


图 9-4 添加启动程序

重新启动计算机，启视频面后会出现如图 9-5 所示的登录画面。

图 9-5 登录画面

如果单击 Cancel 按钮，程序会给出一个如图 9-6 所示的提示信息，询问是否关闭计算机，如果用户不想关闭计算机，仍然可以接受用户的输入，否则就会关闭计算机，

图 9-6 ShounDown 对话框

用户输入用户名和密码，单击 OK 按钮后，如果密码不正确，程序就会给出提示用户重新输入密码的信息如图 9-7 所示。如果密码正确，结束程序的运行。

图 9-7 密码不正确的对话框

附源程序清单如下：

程序清单

```
Option Explicit
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
'窗体位置的函数说明
Private Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long, ByVal dwReserved As Long) As Long
'退出操作系统的函数说明
Private Const SWP_NOSIZE = &H1
Private Const SWP_NOMOVE = &H2
Private Const HWND_TOP = 0
'常量的说明

Private Sub Form_Unload(Cancel As Integer)
ExitWindowsEx 1, 66
'关闭计算机
End Sub

Private Sub Timer1_Timer()
'每隔 1 秒钟触发一次
SetWindowPos frmLogin.hwnd, HWND_TOP, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE
'登录位于其他窗体的前面
frmLogin.SetFocus
End Sub

Private Sub cmdCancel_Click()
    '用户单击 Cancel 按钮时触发
     'set the global var to false
     'to denote a failed login
    Dim r As Integer
    '定义一个整型变量
    r = MsgBox("Are you want to shoutdown?", 1 + 32 + 4096, "ShounDown")
    '显示一个对话框，提示是否关闭计算机
    '对话框中显示有“ 确定 ”和“ 取消 ”两个按钮
    '标题栏中显示的文字为字符串“ ShounDown ”
    '全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。
    If r = 1 Then
        '如果用户在对话框中单击“ 确定 ”按钮
```

```
            ExitWindowsEx 1, 66
            '关闭计算机
        End If
        txtUserName.SetFocus
    End Sub

    Private Sub cmdOK_Click()
        'check for correct password
        If txtUserName.Text = "" Then
            '如果不输入用户名
            MsgBox "Invalid UserName, try again!", , "UserName"
            '弹出一个对话框
            txtUserName.SetFocus
        Else
            If txtPassword = "password" Then
                '如果用户输入的密码正确
                'place code to here to pass the
                'success to the calling sub
                'setting a global var is the easiest
                End
                '退出程序的运行
            Else
                '如果用户输入的密码不正确
                MsgBox "Invalid Password, try again!", , "Login"
                '显示一个对话框
                txtPassword.SetFocus
                SendKeys "{Home}+{End}"
                '重新登录
            End If
        End If
    End Sub
```

9.2 API 图像处理

在 Windows **处理图像的** API **函数中，**Bitblt()**函数是非常重要而且常用的函数之一。**

通过这个函数我们可以实现图像的视频，图像的旋转和图像的动态显示等效果，同时如果想在图像处理方面有所突破的话，也必须能够熟练的掌握 Bitblt()**函数。它的声明如下：**

```
Declare Function BitBlt Lib "gdi32" Alias "BitBlt" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
```

它的参数及其说明如表 9-4 **所示。**

表 9-4 BitBlt()**参数及其说明**

参 数	说 明
hDestDC	接受位图的设备
x	目标矩形左上角的 x 坐标
y	目标矩形左上角的 y 坐标
nWidth	目标矩形的宽度

nHeight	目标矩形的高度
hSrcDC	源设备
xSrc	源位图左上角的 x 坐标
ySrc	源位图左上角的 y 坐标
dwRop	光栅操作

注意：

✍ 特别值得一提的是参数 dwRop，它提供了源图与目标图的 15 中组合方式，但是一般的都采用 SRCCOPY，即把源图复制给目标图，其余的方式请参阅附录一中的有关内容。

下面就以几个示例从不同的方面介绍 Bitblt()函数的应用。

9.2.1 图像的动态效果

这个应用程序的功能是能够以四种不同的效果动态显示一幅图像，如“飞入效果”、“两面向中间”、“从中间扩散”和“拉伸”等效果，具体的步骤如下：

1. 启动一个新的项目

在窗体上添加一个 PictureBox 控件，它的作用是充当图像动态变换中源图的容器，属性设置如下表 9-5 所示。

表 9-5 控件属性设置

属 性	设 置
(Name)	Picture1
AutoRedraw	True
AutoSize	True
Enabled	True
Picture	(Bitmap)
ScaleMode	1 – Twip
Visible	False

其中 Picture 属性初始化为源图所代表的位图。

2. 添加控件

在窗体上添加第二个 PictureBox 控件，作用与第一个 PictureBox 控件相对应，是充当图像动态变换中目标位图的容器，属性设置如下表 9-6 所示。

表 9-6 属性设置

属 性	设 置
(Name)	Picture2
AutoRedraw	False
AutoSize	False
Enabled	True
Picture	(None)
Visible	True

由于第二个 PictureBox 控件作为目标位图的容器，所以它的 Picture 属性一定要设置为 None。

在窗体上添加四个 Command 控件，在程序的运行过程中单击它们可以触发相应的动态效果，如单击“飞入效果”按钮，就会看到一幅图像从左边飞入 PictureBox 控件中。

另外还要把一个 Timer 控件放置到窗体上，用于图像动态显示时延时，它的属性设置如图 9-8 所示。

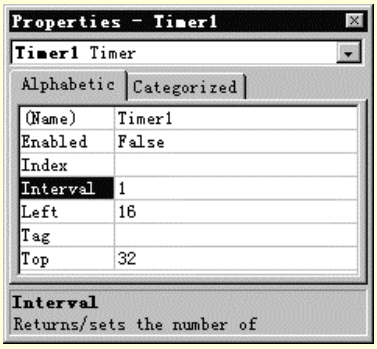


图 9-8 Timer 控件的属性设置

添加控件后的窗体如图 9-9 所示，其中四个按钮从上到下的名称分别为 Command1、Command2、Command3 和 Command4。

图 9-9 添加控件后的窗体

3. 添加代码

下面以“飞入效果”按钮为例来添加代码，其余动态效果的实现过程请参看后面的源程序代码。在设计阶段双击“飞入效果”按钮，在它的 Command1_Click()事件中添加下列代码：

```
Private Sub Command1_Click()
Picture2.Cls
'单击“飞入效果”按钮，清屏
For I = 0 To W Step 1
    BitBlt Picture2.hDC, 0, 0, I, H, Picture1.hDC, W - I, 0, srccopy
    '从左面飞入
    For g = 0 To 10000 Step 1
    Next g
    '延时
Next I
End Sub
```

在程序的运行过程中单击“飞入效果”按钮时，激活了 Command1_Click()事件，程序首先通过 Picture2.Cls 这一条语句清除 PictureBox 控件上原有的图像，为下面动态显示新的图像作准备。

在 Command1_Click()事件的代码段中有一个循环：

```
For I = 0 To W Step 1
    BitBlt Picture2.hDC, 0, 0, I, H, Picture1.hDC, W - I, 0, srccopy
    '从左面飞入
Next I
```

动态显示图像的主要功能都是在这里实现的，“飞入”的效果实现的过程相当于不断的拉着图像从左向右走，而运行时就好象是从左边飞入的一样。此外为了让读者更加清楚的看见“飞入”的效果，在程序中还有一段延时的代码：

```
    For g = 0 To 10000 Step 1
```

```
    Next g
    '延时
```

技巧：

> ✍ 其实“延时”的功能是通过一个有限次的循环来实现的，加入这段代码完全是为了控制动态显示的速度。

按照上面的方法给各个控件添加相应的代码，其余几个动态效果的实现代码见附后的源程序。

4. 运行

存储文件，按功能间 F5 **运行程序，结果如图** 9-10 **所示。**

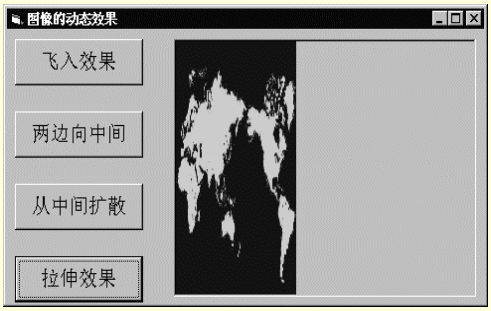


图 9-10　**运行单击“拉伸效果”按钮时的结果**

附源程序代码：

程序清单

```
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
'声明函数
Private Const srccopy = &HCC0020
'常量的说明
Dim H As Integer
Dim W As Integer
'定义两个全局变量用于存储 PictureBox 控件的高度和宽度
Dim I As Integer
Dim J As Integer
'定义两个全局变量用于控制循环

Private Sub Command1_Click()
Picture2.Cls
'单击"飞入效果"按钮，清屏
For I = 0 To W Step 1
    BitBlt Picture2.hDC, 0, 0, I, H, Picture1.hDC, W - I, 0, srccopy
    '从左面飞入
    For g = 0 To 10000 Step 1
    Next g
    '延时
Next I
End Sub

Private Sub Command2_Click()
```

```
Picture2.Cls
'单击"两面向中间"按钮，清屏
For I = 0 To W / 2 Step 1
    BitBlt Picture2.hDC, I, 0, 1, H, Picture1.hDC, I, 0, srccopy
    BitBlt Picture2.hDC, W - I, 0, W, H, Picture1.hDC, W - I, 0, srccopy
     '从两面向中间显示
    For g = 0 To 10000
    Next g
    '延时
Next I
End Sub

Private Sub Command3_Click()
Picture2.Cls
'单击"从中间扩散"按钮，清屏
Timer1.Enabled = True
'启动定时器
I = W / 2
J = H / 2
End Sub

Private Sub Command4_Click()
Picture2.Cls
'单击"拉伸"按钮，清屏
For k = 0 To 1.01 Step 0.01
  For I = W To 0 Step -1
      BitBlt Picture2.hDC, I * k, 0, 1, H, Picture1.hDC, I, 0, srccopy
      '从左边向右边拉伸
  Next I
Next k
End Sub

Private Sub Form_Load()
H = Picture2.Height
W = Picture2.Width
'程序的初始化
End Sub

Private Sub Timer1_Timer()
BitBlt Picture2.hDC, I, J, W - 2 * I, H - 2 * J, Picture1.hDC, I, J, srccopy
'从中间扩散
I = I - 1
J = J - 1
End Sub
```

以上是图像动态显示应用程序的源代码，读者也可以依据以上方法设计出另外的动态显示方式。

9.2.2 图像的旋转

这个应用程序的功能是能够选择四种不同的角度旋转一幅图像，如逆时针旋转 90 度和逆时针旋转 270 度效果，在此基础上，读者可以自己开发出将一幅图像旋转任意角度的应用程序，具体的步骤如下：

1. 启动一个新的项目

在窗体上添加一个 PictureBox 控件，它的作用是充当待旋转图像的临时容器，属性设置如下表 9-7 所示。

表 9-7 控件属性设置

属 性	设 置
(Name)	Picture1
AutoRedraw	True
AutoSize	True
Enabled	True
Picture	(Bitmap)
ScaleMode	1 – Twip
Visible	False

提示：

✍ 其中 Picture 属性初始化为待旋转的图像，并且在运行时不可见。

2. 添加控件

在窗体上添加第二个 PictureBox 控件，作用与第一个 PictureBox 控件相对应，它是充当图像旋转后的容器，属性设置如下表 9-8 所示。

表 9-8 控件属性设置

属 性	设 置
(Name)	Picture2
AutoRedraw	False
AutoSize	False
Enabled	True
Picture	(None)
Visible	True

注意：

✍ 由于在程序设计阶段尚未装入图像，所以它的 Picture 属性设置为 None。

在窗体上添加四个 Command 控件，在程序的运行过程中单击它们可以触发相应的旋转效果，如单击“旋转 90 度”按钮，就会看到原来的那幅图像在 PictureBox 控件中按逆时针方向旋转了 90 度。完成上述工作后的窗体如图 9-11 所示。

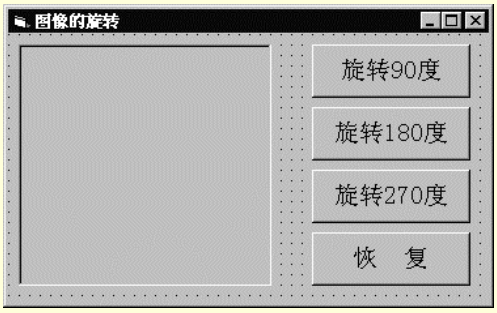


图 9-11 添加控件后的窗体

其中四个按钮从上到下的名称分别为 Command1、Command2、Command3 和 Command4。

3. 程序的初始化

在设计阶段双击窗体，在窗体的声明段中加入下列代码：

```
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long 声明函数

Private Const srccopy = &HCC0020    '声明常量
```

```
Dim H As Integer
Dim W As Integer
```

程序首先声明了 BitBlt()函数，然后又通过 Private Const srccopy = &HCC0020 这一条语句来声明一个常量——表示了在 BitBlt()函数中采取了把源图复制给目标位图的光栅操作。随后定义的两个全局变量 H 和 W 分别存储控件 Picture1 的高度和宽度。

4. 为控件添加代码

下面以“旋转 270 度”按钮为例来添加代码，其余旋转效果的实现原理与此大同小异，具体的代码请参看后面的源程序。

在设计阶段双击“旋转 270 度”按钮，在它的 Command3_Click()事件中添加下列代码：

```
Private Sub Command3_Click()
Picture2.Picture = LoadPicture("") '清屏
For I = H To 0 Step -1
        For J = 0 To W Step 1                    '从右到左的显示图像
            BitBlt Picture2.hDC, I, J, 1, 1, Picture1.hDC, W - J, H - I, srccopy
            '将图像旋转 270 度
        Next J
Next I
End Sub
```

在程序的运行过程中单击“旋转 270 度”按钮时，激活了 Command3_Click()事件，程序首先通过 Picture2.Picture = LoadPicture("")这一条语句清除 PictureBox 控件上原有的图像，为下面显示新的图像作准备。

在 Command3_Click()事件的代码段中有一个循环：

```
For I = H To 0 Step -1
        For J = 0 To W Step 1            '从右到左的显示图像
            BitBlt Picture2.hDC, I, J, 1, 1, Picture1.hDC, W - J, H - I, srccopy
            '将图像旋转 270 度
        Next J
Next I
```

旋转图像的主要功能都是在这里实现的，“旋转 270 度”的效果实现的过程相当于从源位图中读取一个象素，然后放到旋转 270 度后的相应位置上，由于是逐点旋转，所以运行时的效果就是整个图像是从右向左擦除显示，同时旋转了 270 度。

5. 运行

按照上面的方法给各个控件添加相应的代码，其余几个旋转效果的实现代码见附后的源程序。存储文件，运行程序，初始化面如图 9-12 所示。

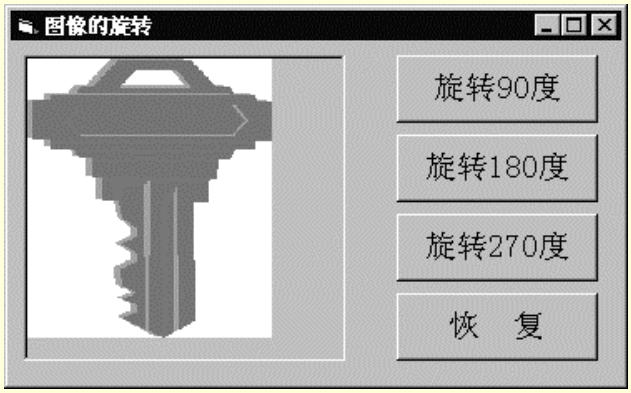


图 9-12 程序运行的初始化面

在运行时单击“旋转 270 度”按钮，结果如图 9-13 所示。

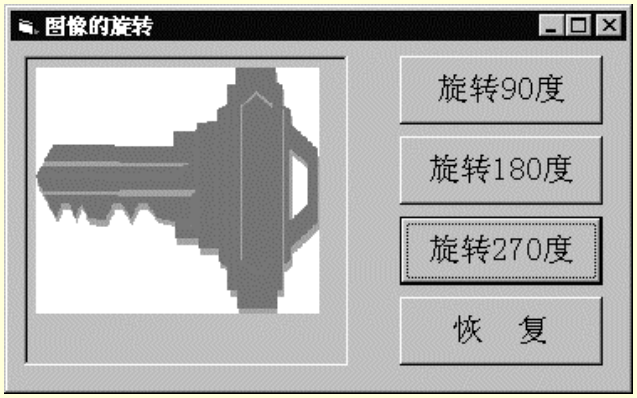


图 9-13 运行阶段单击“旋转 270 度”按钮

源程序源代码：

程序清单

```
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
'声明函数
Private Const srccopy = &HCC0020
'声明常量
Dim H As Integer
Dim W As Integer
'定义全局变量
Private Sub Command1_Click()
Picture2.Picture = LoadPicture("")
'清屏
For J = W To 0 Step -1
        For I = 0 To H Step 1
            '从下到上的显示图像
            BitBlt Picture2.hDC, I, J, 1, 1, Picture1.hDC, J, I, srccopy
            '将图像旋转 90 度
        Next I
Next J
End Sub

Private Sub Command2_Click()
Picture2.Picture = LoadPicture("")
'清屏
For J = 0 To H Step 1
        For I = 0 To W Step 1
            '从上到下的显示图像
            BitBlt Picture2.hDC, I, J, 1, 1, Picture1.hDC, W - I, H - J, srccopy
            '将图像旋转 180 度
        Next I
Next J
End Sub
```

```
Private Sub Command3_Click()
Picture2.Picture = LoadPicture("")
'清屏
For I = H To 0 Step -1
        For J = 0 To W Step 1
            '从右到左的显示图像
            BitBlt Picture2.hDC, I, J, 1, 1, Picture1.hDC, W - J, H - I, srccopy
            '将图像旋转 270 度
        Next J
Next I
End Sub

Private Sub Command4_Click()
Picture2.Picture = LoadPicture("")
'清屏
For I = 0 To W Step 1
        For J = 0 To H Step 1
            '从左向右的显示图像
            BitBlt Picture2.hDC, I, J, 1, 1, Picture1.hDC, I, J, srccopy
            '图像不旋转
        Next J
Next I
End Sub

Private Sub Form_Load()
Picture2.Picture = LoadPicture("e:\user\ylm\api\key.bmp")
'控件的初始化
H = Picture1.Height
W = Picture1.Width
End Sub
```

9.3 API 的多媒体应用

9.3.1 制作音乐播放程序

前面已经介绍过利用 MMControl 控件制作一个 CD 播放器，实现起来十分的方便，但是，以下将要介绍的利用 API 函数制作 CD 播放器则更加简单和易用，只需要了解一个 API 函数——mciExecute()函数即可。

在这里制作 CD 播放器不用添加多媒体控件，只需要写几行代码即可，具体的步骤如下：

1．首先启动一个新的项目

首先用鼠标选择 Windows 操作系统“开始”菜单中的“程序”/Microsoft Visual Studio 中的 Microsoft Visual Basic 选项来激活 Visual Basic 应用程序，在 Vbisual Basic 6.0 的集成开发环境中用鼠标选择“文件”菜单中的“新建工程”选项。在“新建工程”对话框中选择“标准 EXE”选项，单击“确定”按钮，在 Visual Basic 中就新建了一个标准的工程文件，同时打开了一个空白的窗体。

在空白窗体的声明段中添加对函数 mciExecute()的声明；

```
Private Declare Function mciExecute Lib "winmm.dll" (ByVal lpstrCommand As String) As Long
```

2．添加控件

在窗体上添加六个 CommandButton 控件和四个 OptionButton 控件，它们的属性设置如表 9-9 所示。

表 9-9　控件的属性及其设置

控　件	属　性	设　置
CommandButton 控件	(Name)	Open
	Caption	打开
CommandButton 控件	(Name)	Play
	Caption	播放
CommandButton 控件	(Name)	stop1
	Caption	停止/暂停
CommandButton 控件	(Name)	Eject
	Caption	弹碟
CommandButton 控件	(Name)	Close1
	Caption	装入光碟
CommandButton 控件	(Name)	Exit
	Caption	退出
OptionButton 控件	(Name)	Option1
	Caption	左声道输出
	Value	False
OptionButton 控件	(Name)	Option2
	Caption	右声道输出
	Value	False
OptionButton 控件	(Name)	Option3
	Caption	立体声输出
	Value	True
OptionButton 控件	(Name)	Option4
	Caption	静音
	Value	False

添加控件后的窗体如图 9-14 所示。

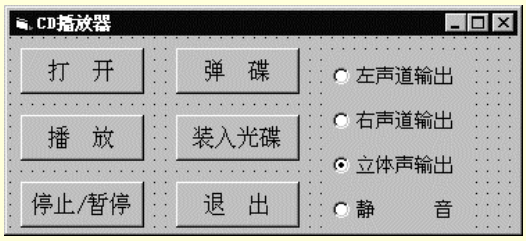


图 9-14　添加控件后的窗体

3．程序的初始化

由于在未打开 CDAudio 设备之前不能够对它进行播放等操作，所以要在窗体的 Form_Load()过程中添加有关控件的初始化代码如下：

```
Private Sub Form_Load()
play.Enabled = False
stop1.Enabled = False
eject.Enabled = False
close1.Enabled = False
option1.Enabled = False
option2.Enabled = False
option3.Enabled = False
option4.Enabled = False
'初始化按钮的状态
End Sub
```

在程序刚刚开始运行时，由于还没有打开相应的 CDAudio 设备，还不能够对它执行操作，所以“播放”、“弹碟”、“停止/暂停”和“装入光碟”按钮处于无效的状态，同理，四个 OptionButton 控件也应该处于无效的状态。

4. 为 CommandButton 控件添加代码

下面以“打开”按钮为例来添加代码，在设计阶段双击“打开“按钮，在 open_Click()事件中添加下列代码：

```
Private Sub open_Click()
mciExecute "open cdaudio alias cd"
'打开 CDAudio 设备
play.Enabled = True
stop1.Enabled = True
eject.Enabled = True
close1.Enabled = False
Option1.Enabled = True
Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True
'设置按钮的状态
End Sub
```

程序首先通过 mciExecute "open cdaudio alias cd"这一条语句来打开 CDAudio 设备，然后可以对各个按钮的有效状态进行相应的设置，如“播放”按钮应处于有效的状态，而“装入光碟”按钮则应该处于无效的状态。

其余几个 CommandButton 控件的代码就不一一的解释了，具体的代码请参见后面所服的源程序。

5. 处理 OptionButton 控件的事件

在打开 CDAudio 设备后，播放 CD 音乐之前，可以通过四个 OptionButton 控件来选择声音的输出形式，如“左声道输出”和“立体声输出”等，代码如下：

```
Private Sub play_Click()
If Option1(0).Value Then
    mciExecute "set cd audio right off"
    '关闭右声道
    mciExecute "set cd audio left on"
    '打开左声道
ElseIf Option2(1).Value Then
    mciExecute "set cd audio left off"
    '关闭左声道
    mciExecute "set cd audio right on"
    '打开右声道
ElseIf Option3(2).Value Then
    mciExecute "set cd audio all on"
    mciExecute "set cd audio left on"
    mciExecute "set cd audio right on"
    '立体声播放 CD
ElseIf Option4(0).Value Then
    mciExecute "set cd audio all off"
    '静音
End If
mciExecute "play cd"
'播放 CD
End Sub
```

在播放 CD 音乐之前，即 mciExecute "play cd"语句执行之前，通过一个条件判断语句来确定用户对声音输出形式的选择，比如选择“右声道输出”，那么在播放 CD 音乐之前程序会执行下面的几条语句：

```
mciExecute "set cd audio left off"
'关闭左声道
mciExecute "set cd audio right on"
'打开右声道
```

即首先关闭左声道，打开右声道，然后再执行相应的播放等操作。

6. 添加剩余的代码并运行

在剩余的控件中添加能够实现相应功能的代码，如在“退出”按钮中添加下列代码：

```
Private Sub exit_Click()
mciExecute "close cd"
'关闭 CD 设备
End
'结束运行
End Sub
```

其余几个控件的详细代码见附后的程序源代码。

存储文件，运行程序，初始画面如图 9-15 所示。

图 9-15 程序运行初始画面

在程序运行时，单击“打开”按钮，这时“播放”按钮变为有效的状态，单击“播放”按钮，从音箱的右声道中就可以听到 CD 音乐的声音，结果如图 9-16 所示。

图 9-16 运行结果

附源程序如下：

程序清单

```
Private Declare Function mciExecute Lib "winmm.dll" (ByVal lpstrCommand As String) As Long
'声明函数
Private Sub close1_Click()
mciExecute "set cd door closed"
'合上光驱
close1.Enabled = False
eject.Enabled = True
End Sub
```

```
Private Sub eject_Click()
mciExecute "set cd door open"
'弹出光碟
eject.Enabled = False
close1.Enabled = True
'设置按钮的状态
End Sub

Private Sub exit_Click()
mciExecute "close cd"
'关闭 CD 设备
End
'结束运行
End Sub

Private Sub Form_Load()
play.Enabled = False
stop1.Enabled = False
eject.Enabled = False
close1.Enabled = False
Option1.Enabled = False
Option2.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
'初始化按钮的状态
End Sub

Private Sub open_Click()
mciExecute "open cdaudio alias cd"
'打开 CD 设备
play.Enabled = True
stop1.Enabled = True
eject.Enabled = True
close1.Enabled = False
Option1.Enabled = True
Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True
'设置按钮的状态
End Sub

Private Sub Option1_Click(Index As Integer)
mciExecute "set cd audio right off"
'关闭右声道
mciExecute "set cd audio left on"
'打开左声道
```

```
End Sub
Private Sub Option2_Click(Index As Integer)
mciExecute "set cd audio left off"
'关闭左声道
mciExecute "set cd audio right on"
'打开右声道
End Sub

Private Sub Option3_Click(Index As Integer)
mciExecute "set cd audio all on"
mciExecute "set cd audio left on"
mciExecute "set cd audio right on"
'立体声播放 CD
End Sub

Private Sub Option4_Click(Index As Integer)
mciExecute "set cd audio all off"
'静音
End Sub

Private Sub play_Click()
mciExecute "play cd"
'播放 CD
Option1.Enabled = False
Option2.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
'设置按钮的状态
End Sub

Private Sub stop1_Click()
mciExecute "stop cd"
'停止播放
Option1(0).Enabled = True
Option2(1).Enabled = True
Option3(2).Enabled = True
Option4(0).Enabled = True
'设置按钮的状态
End Sub
```

9.3.2 制作视频播放器

前面在第六章已经介绍过利用 MMControl 控件制作一个 AVI 视频播放器，实现起来十分的方便。
下面将要介绍的利用 API 函数制作视频播放器，只需要了解一个 API 函数——mciSendString()函数即可。

注意：

> ✍ **通过上一节所讲的内容以及本节所讲的内容的学习，读者应该能总结各种制作多媒体的方法及其特点。**

下面的示例综合上面提到的两个 API 函数--mciExecute()和 mciSendString()，它的具体步骤如下：

1．建立项目

首先启动一个新的项目，在窗体的声明段中加入对函数 mciExecute()和 mciSendString()的声明；

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" (ByVal lpstrCommand As String, ByVal lpstrReturnString As String, ByVal uReturnLength As Long, ByVal hwndCallback As Long) As Long
Private Declare Function mciExecute Lib "winmm.dll" (ByVal lpstrCommand As String) As Long
```

然后，在工具栏上添加一个 CommonDialog 控件，添加控件后工具栏如图 9-17 所示。

图 9-17　添加控件后的工具栏

CommonDialog 控件的功能是显示一个“打开文件”的对话框，从中可以选择我们所需要的文件，同时还可以返回文件的路径和文件名。

按照不同功能的需要，在窗体上放置四个 CommandButton 控件、四个 Label 控件和一个 PictureBox 控件，它们的属性设置如表 9-10 所示。

表 9-10　控件的属性设置

控　件	属　性	设　置
CommandButton	(Name)	Open
	Caption	打开
CommandButton	(Name)	Play
	Caption	播放
CommandButton	(Name)	stop1
	Caption	停止/暂停
CommandButton	(Name)	Exit
	Caption	退出
Label	(Name)	Label 1
	Caption	
Label	(Name)	Label 2
	Caption	
Label	(Name)	Labe3
	Caption	设备名称：
Label	(Name)	Label 4
	Caption	播放文件：
PictureBox	(Name)	Picture1
	Picture	(None)
CommonDialog	(Name)	CommonDialog1
	Filename	*.AVI
	Filter	*.AVI
	Initdir	d:\vbstudio

其中四个 CommandButton 控件的功能是实现相应的“播放”和“暂停”等动作，四个 Label 控件的功能是显示播放的视频文件的信息，而 PictureBox 控件的作用是为播放视频文件提供一个容器。

添加控件后窗体如图 9-18 所示。

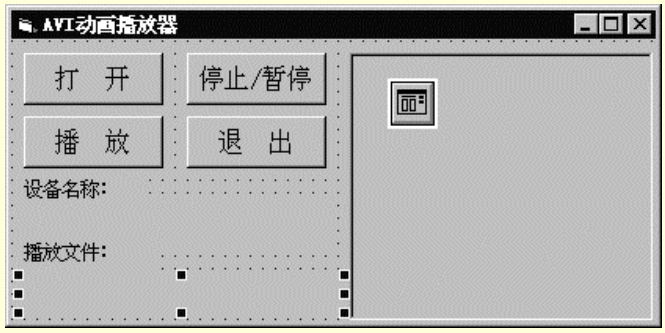


图 9-18 添加控件后的窗体

2．程序的初始化

在窗体的 Form_Load()事件中加入程序的初始化代码：

```
Private Sub Form_Load()
flag = False
'设置标志
play.Enabled = False
stop1.Enabled = False
'初始化按钮的状态
End Sub
```

程序首先通过 flag = False 这条语句来设置标志变量 flag，通过这个变量在以后的运行过程中可以判断程序是否打开了视频播放设备，以便进行相应的处理；

而接下来的两条语句：play.Enabled = False 和 stop1.Enabled = False 则是用来初始化按钮的状态的。

3．给“播放”按钮添加代码

在设计阶段双击“播放”按钮，在它的 open_Click()事件中添加下列代码：

```
Private Sub open_Click()
Dim returnstr As String * 128
'定义储存返回字符串的变量
Dim str1 As String * 128
CommonDialog1.Action = 1
'显示“打开文件”的对话框
str1 = mciSendString("open " & CommonDialog1.FileName & " alias avi style child parent" + Str$(Form1.hWnd), returnstr, 128, 0)
'打开视频播放设备
mciExecute ("set avi seek exactly on")
'设置搜索模式
mciExecute ("window avi handle" + Str$(Picture1.hWnd))
'设置视频播放的窗口
play.Enabled = True
stop1.Enabled = True
'设置按钮的状态
str1 = mciSendString("info avi product", returnstr, 128, 0)
'取得播放视频的设备名称
Label1.Caption = returnstr
'显示播放视频的设备名称
Label2.Caption = CommonDialog1.FileName
'显示文件的路径和文件名
mciExecute ("cue avi to 0")
'显示视频的静桢画面
```

```
flag = True
‘ 设置标志
End Sub
```

程序说明：

程序运行的过程中单击“播放”按钮时，激活了 open_Click()**事件，程序就会通过** CommonDialog1.Action = 1 **这条语句来显示一个“打开文件”的对话框，如图** 9-19 **所示。**

图 9-19 **“打开”对话框**

当选择一个有效的视频文件后，程序就会通过如下的语句打开一个视频播放设备，视频设备的别名为“ avi **”，并且视频将会在窗体的一个子窗口播放：**

```
str1 = mciSendString("open " & CommonDialog1.FileName & " alias avi style child parent" + Str$(Form1.hWnd), returnstr, 128, 0)
```

后面的语句 mciExecute ("window avi handle" + Str$(Picture1.hWnd))**用来设置具体的播放视频的容器为** PictureBox **控件，然后** mciExecute ("cue avi to 0")**所实现的功能是显示视频的静帧画面，源代码由于比较容易理解，在此就不多加介绍了。**

4. 运行

提示：

另外几个 CommandButton 控件的代码读者可以参考前面“制作 CD 播放器”的示例自己添加。

完整的代码见后面所附的程序源代码。添加完代码，请存储文件，按 F5 **键运行程序，结果如图** 9-20 **所示。**

图 9-20 **程序运行的初始画面**

单击“打开”按钮后，就会弹出一个“打开”的对话框，在其中选择一个有效的视频文件后，结果如图 9-21 **示。**

图 9-21 **选择了一个有效的视频文件**

在窗体上会出现视频文件的一些基本的信息，如设备名称和文件的路径和文件名，单击“播放”按钮后，

结果如图 9-22 示。

图 9-22 运行结果

附程序源代码：

程序清单

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" (ByVal lpstrCommand As String, ByVal lpstrReturnString As String, ByVal uReturnLength As Long, ByVal hwndCallback As Long) As Long
Private Declare Function mciExecute Lib "winmm.dll" (ByVal lpstrCommand As String) As Long
'声明函数

Dim flag As Boolean
Private Sub exit_Click()
If flag Then
    mciExecute "close avi"
    '关闭视频设备
End If
End
'结束运行
End Sub

Private Sub Form_Load()
flag = False
'设置标志
play.Enabled = False
stop1.Enabled = False
'初始化按钮的状态
End Sub

Private Sub open_Click()
Dim returnstr As String * 128
'定义储存返回字符串的变量
Dim str1 As String * 128
CommonDialog1.Action = 1
'显示“打开文件”的对话框
str1 = mciSendString("open " & CommonDialog1.FileName & " alias avi style child parent" + Str$(Form1.hWnd), returnstr, 128, 0)
'打开视频播放设备
mciExecute ("set avi seek exactly on")
'设置搜索模式
```

```
mciExecute ("window avi handle" + Str$(Picture1.hWnd))
'设置视频播放的窗口
play.Enabled = True
stop1.Enabled = True
'设置按钮的状态
str1 = mciSendString("info avi product", returnstr, 128, 0)
'取得播放视频的设备名称
Label1.Caption = returnstr
'显示播放视频的设备名称
Label2.Caption = CommonDialog1.FileName
'显示文件的路径和文件名
mciExecute ("cue avi to 0")
'显示视频的静桢画面
flag = True
'设置标志
End Sub

Private Sub play_Click()
mciExecute "play avi"
'播放 AVI 视频
End Sub

Private Sub stop1_Click()
mciExecute "pause avi"
'暂停播放
End Sub
```

9.4 小 结

本章着重介绍了在 Visual Basic 中对 Windows 的 API 函数进行调用的方法。重点是介绍 API 函数对图像、声音以及视频的处理方法，虽然不必调用 Windows 的 API 函数也可以实现一些多媒体程序，但是还是有一些功能单凭 Visual Basic 的内部控件是办不到的，这就需要调用 API 函数来实现。所以说，API 函数实际上是对 Visual Basic 内部功能的一种完善。

最后需要说明的是，API 函数的领域实际是比较广的，它可以应用于各种 Windows 程序开发软件中，比如 Visual C++、Borland Dephi 等。本章所介绍的 API 知识仅局限于 Visual Basic 中最常用的一些领域，其实 API 的使用还是非常值得研究的，在这方面有兴趣的读者可以参考有关 API 函数专用开发手册。

自此，本书全部讲解完毕，以一句古老的格言结束本书，和读者共勉：

“书山有路勤为径，学海无涯苦作舟。”