第 12 章　多媒体编程

前面章节介绍了有关 Visual Basic 6.0 的程序设计基础和使用 Visual Basic 6.0 进行应用程序设计的基本工具，下面将就 Visual Basic 6.0 的具体应用进行详细介绍。Visual Basic 6.0 功能强大，可以进行数据库应用程序、多媒体应用程序、网络应用程序等多类系统的开发。本章将首先对 Visual Basic 6.0 提供的多媒体编程功能进行详细介绍。

12.1　使用多媒体控件编程

Visual Basic 6.0 中，多媒体程序设计主要是通过 MultiMedia MCI 控件和 WindowsMediaPlayer 控件来实现的，本节将主要针对 MCI 控件的一些常用属性、事件和方法进行介绍。此外，本节还将简单介绍使用 WindowsMediaPlayer 控件进行多媒体编程的过程。

12.1.1　添加 Multimedia MCI 控件

Multimedia MCI 控件管理媒体控制接口（MCI）设备上的多媒体文件的记录与回放。从概念上说，这种控件就是一组按钮，用于向诸如声卡、MIDI 序列发生器、CD-ROM 驱动器、视频 CD 播放器和视频磁带记录器及播放器等设备发出 MCI 命令。此外，MCI 控件还支持 Windows（*.avi）视频文件的回放。

与其他 ActiveX 控件相同，Multimedia MCI 控件也不在 Visual Basic 6.0 的标准控件工具栏上。因此，在使用该控件前，首先需要添加该控件到控件栏上。

要将 Multimedia MCI 控件添加到窗体上，可通过选择“工程”→“部件”命令，在“部件”对话框的“控件”选项卡中选择“Microsoft Multimedia Control 6.0”选项后，单击“应用”按钮，即可在工具箱中添加 Multimedia 控件按钮，如图 12-1 所示。

添加完成后，即可以在控件工具栏上看到该控件的图标，如图 12-2 所示。

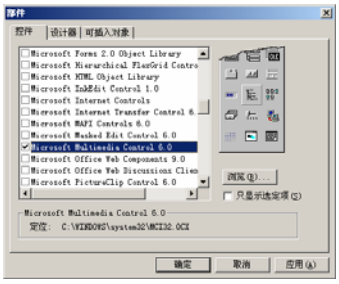

图 12-1　添加 Multimedia 控件

图 12-2　MCI 控件

接下来，就可以像使用普通标准控件一样使用该 MCI 控件了。在新建窗体中，将该控件拖动到窗体上，其形状如图 12-3 所示。

图 12-3　Multimedia MCI 控件

12.1.2　Multimedia MCI 控件的属性

Multimedia MCI 控件用于判断当前是否有文件在播放、播放的模式等，主要通过该控件的属性设置来完成。Multimedia MCI 控件的主要属性如表 12-1 所示。

表 12-1　Multimedia MCI 控件的主要属性

属性名	属性值	说明
AutoEnable	True 或 False	能否自动检测功能按钮的状态
按钮的 Enable	True 或 False	某按钮是否有效
按钮的 Visible	True 或 False	某按钮是否可见
Can 按钮名	True 或 False	监测媒体设备的 Play、Eject 等功能
Command	MCI 命令	指定将要执行的 MCI 命令
DeviceID	长整数	指定当前打开 MCI 设备的设备 ID
DeviceType	设备类别代号	指定要打开的 MCI 设备类型
FileName	文件名	设置媒体设备打开或存储的文件名
Frames	长整数	设置媒体设备进退的帧数
From、TO	长整数	为 Play 或 Record 命令规定起始点
HWndDisplay	长整数	指定电影播放窗口
Length	长整数	返回所使用的多媒体文件长度
Mode	524（未打开） 525（停止） 526（正在播放） 527（正在记录） 528（正在搜索） 529（暂停） 530（准备好）	返回打开的 MCI 设备的当前状态

可以看出，这些属性控制了对音频视频文件的播放、暂停等功能，也设定了播放文件的

类型、名称等。此外，Multimedia MCI 控件还有一个较为重要的属性——DeviceType 属性，该属性用于设定播放的文件类型。常见的多媒体设备类别如表 12-2 所示。

表 12-2　常见多媒体设备类别

设备类型	DeviceType 属性	文件类型	说明
CD audio	Cdaudio		音频 CD 播放器
Digital Audio Tape	Dat		数字音频磁带播放器
Digital Video	DigitalVideo		窗口中的数字视频
Other	Other		未定义 MCI 设备
Overlay	Overlay		视频重叠设备
Scanner	Scanner		图像扫描仪
Sequencer	Sequencer	.mid	音响设备数字接口（MIDI）序列发生器
Vcr	VCR		可程序控制录像机
AVI	AVIVideo	.avi	视频文件
Videodisc	Videodisc		激光视盘播放器
Waveaudio	Waveaudio	.wav	播放数字波形文件的音频设备

12.1.3　Multimedia MCI 控件的命令

MCI 控件包含了许多控制按钮，如播放、暂停、停止等。这些操作都是通过 MCI 命令来实现的，MCI 的常用命令如表 12-3 所示。

表 12-3　MCI 常用命令

命令	MCI 命令	说明
Open	MCI_OPEN	打开 MCI 设备
Close	MCI_CLOSE	关闭 MCI 设备
Play	MCI_PLAY	用 MCI 设备进行播放
Pause	MCI_PAUSE	暂停播放或录制
Stop	MCI_STOP	停止 MCI 设备
Back	MCI_STEP	单步回倒
Step	MCI_STEP	步进
Prev	MCI_SEEK	使用 Seek 命令跳到当前曲目的起始位置或上一个曲目
Next	MCI_SEEK	使用 Seek 命令跳到下一曲目的起始位置
Seek	MCI_SEEK	向后或向前查找一个位置
Record	MCI_RECORD	录制 MCI 设备的输入
Eject	MCI_SET	从 CD 驱动器中弹出媒体
Save	MCI_SAVE	保存打开的文件

通常，MCI 控件就是通过这些命令和属性来对文件的播放和类型进行控制的。这些属性和命令将在后面小节的实例中具体使用到。

12.1.4　多媒体编程步骤

在 Visual Basic 6.0 中，如果通过 MCI 等多媒体控件进行多媒体编程，一般需要通过如下 5 个步骤来实现。

（1）为窗体添加 MCI 多媒体控件，并用 Multimedia MCI 控件的 DeviceType 属性指定多媒体设备的类别。

（2）如果程序设计中涉及到媒体文件，需用 Filename 属性指定文件。

（3）进行具体播放控制前，需用 Command 属性的 Open 值打开媒体设备，即使用 Open 命令打开设备。

（4）具体播放控制中，用 Command 属性的其他值控制媒体设备，并对特殊键（如 Pause 暂停键）进行具体设计。

（5）完成播放后，需用 Command 属性的 Close 值关闭媒体设备。

12.1.5　使用 MCI 控件制作音频播放器

为了更具体地介绍 MCI 控件是如何实现 Visual Basic 6.0 中的多媒体编程的，下面给出一个音频播放器的实例。

该实例利用 Multimedia MCI 控件的多媒体功能制作一个简单的音频播放器。当单击“打开”按钮后，从“打开文件”对话框中选择要播放的文件，然后利用 Multimedia MCI 控件进行播放。根据如上多媒体编程的一般步骤，该实例的实现步骤如下。

（1）界面设计。新建一个“标准 EXE”工程后，在新建窗体上添加一个 Multimedia MCI 控件 MMControl1、一个 Slider 控件 Slider1、一个 Toolbar 控件 Toolbar1、一个 CommonDialog 控件 CommonDialog1，如图 12-4 所示。

提示：Multimedia MCI 控件需在“部件”对话框的“控件”选项卡中应用“Microsoft Multimedia Control 6.0”选项后才能使用，Slider 控件、Toolbar 控件都需应用“Microsoft Windows Common Control 6.0”后才能使用，CommonDialog 控件需应用“Microsoft Common Dialog Control 6.0”选项后才能使用。

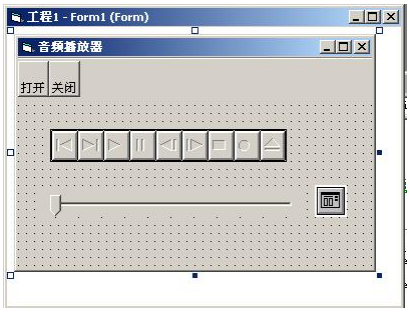

图 12-4　音频播放器窗体设计

（2）编写代码。此处涉及的代码较多，将分别进行介绍。首先在在窗体 Form1 的 Load 事件中，输入如下代码。

```
Private Sub Form_Load()
   With MMControl1
   .Notify = True                  '将 Notify 属性设置为 True ，以便在 Done 事件中处理错误信息
    .Wait = False                  '将 Wait 属性设置为 False ，采用非阻塞方式传递 MCI 命令
```

```
    End With
End Sub
```

上述代码用于给窗体上的控件进行初始化设置。可以看出，这是通过 MCI 控件的属性进行的设置，设置其 Notify 属性为 True 值，Wait 属性为 False 值。接下来在 Multimedia MCI 控件 MMControl1 中的 Done 事件中，输入如下代码。

```
Private Sub MMControl1_Done(NotifyCode As Integer)
    With MMControl1
        If .Error <> 0 Then
        MsgBox "Error #" & Error & "" & .ErrorMessage                    ‘错误处理
        End If
    End With
End Sub
```

上述代码在 MMControl1_Done 事件中显示错误信息，必要时也可以加入其他代码。同时，在 Multimedia MCI 控件 MMControl1 中的 StatusUpdate 事件中，输入如下代码。

```
Private Sub MMControl1_StatusUpdate()
    With MMControl1
        If .DeviceID <> 0 Then
            If Slider1.Value <> .Position Then Slider1.Value = .Position      ‘设定滑竿位置
        End If
    End With
End Sub
```

上述代码在 MMControl1_StatusUpdate 事件过程中，设置 Slider1 控件的滑杆位置，然后在工具栏 ToolBar 控件中，输入如下代码。

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    On Error Resume Next
    Select Case Button.Index
    Case 1                                                          ‘单击“打开”按钮"
        With CommonDialog1
            .CancelError = True
            .ShowOpen                                               ‘显示文件对话框
            If .FileName <> "" Then
                MMControl1.FileName = .FileName                     ‘获取打开文件的文件名
                MMControl1.Command = "Open"                         ‘打开文件
                Slider1.Max = MMControl1.Length                     ‘设定滑竿最大位置
                Slider1.SmallChange = 1
                Slider1.LargeChange = Slider1.Max / 5               ‘设定滑竿每次的移动量
            End If
              Form_Resize
            End With
    Case 2                                                          ‘单击“关闭”按钮
        If MMControl1.Mode = mciModeNotOpen Then                    ‘当设备没有打开时
            MMControl1.Command = "stop"                             ‘停止播放
            MMControl1.Command = "close"                            ‘关闭播放器
            Form_Resize
        End If
    End Select
End Sub
```

上述代码根据用户选择的按钮，分别进行不同的操作。单击“打开”按钮，则打开文件选择对话框，提供给文件目录以供用户选择目标文件进行播放，同时对进度条 Slider 控件的

属性进行设置。单击“关闭”按钮，关闭播放器。最后，在窗体 Form1 的 Unload 事件中，输入以下代码用于在关闭 MCI 设备前，显式使用 stop 停止 MCI 设备。

```
Private Sub Form_Unload(Cancel As Integer)
    Form1.MMControl1.Command = "stop"
    Form1.MMControl1.Command = "close"
```

（3）执行程序。上述代码编写完成后，执行程序，执行效果如图 12-5 所示。单击“打开”按钮，选择音频文件，就可以播放音频文件了。

至此，一个具有播放音频功能的播放器就完成了。可以看到，使用 MCI 控件实现多媒体编程是非常方便的，只需调用 MCI 控件的对应属性和命令即可。

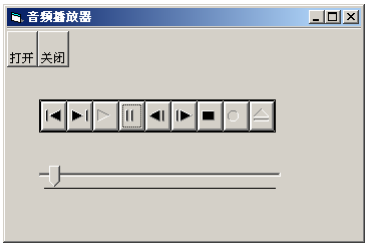

图 12-5　音频播放器

12.1.6　使用 WindowsMediaPlayer 控件

WindowsMediaPlayer 控件用于播放音频和视频多媒体文件。相比于 MMControl 控件，WindowsMediaPlayer 控件能够识别更多的多媒体文件格式，如 MIDI、WAV、SND、AU、AIF、WMA、MP3、AVI、WMA、MPEG、MLV、ASF 及 WM 文件格式。WindowsMediaPlayer 控件使用非常方便，在程序设计中只需很少的代码，即可完成多媒体文件的播放等相关操作，所以该控件的使用频率越来越高。在某种程度上可以说，WindowsMediaPlayer 控件将 Windows 系统自带的媒体播放器嵌入了用户的应用程序。该控件的主要属性如表 12-4 所示。

表 12-4　WindowsMediaPlayer 控件主要属性

属性	功能
URL	用于指定播放多媒体文件的位置
uiMode	用于设置播放器的界面，可为 Full、Mini、None、Invisable
playState	播放状态，1=停止；2=暂停；3=播放；6=正在缓冲；9=正在连接；10=准备播放
fullScreen	设置是否全屏显示
currentPositionString	当前曲目的进度，字符串格式，如“00：13”
settings	播放器基本设置。通过该属性下的其他属性对播放器的基本设置进行相应设置
currentMedia	当前媒体属性。通过该属性获取当前媒体的基本信息，如媒体长度、原始地址等信息

WindowsMediaPlay 控件的方法用于播放器的基本控制，具体方法如表 12-5 所示。

表 12-5　WindowsMediaPlay 控件基本方法

基本方法	功能
Play	播放媒体
Stop	停止播放
Pause	暂停播放
fastForward	快进

续表

基本方法	功能
fastReverse	快退
Next	下一曲
Previous	上一曲

例如，用 WindowsMediaPlay 控件完成一个简单的多媒体播放器。该程序可以完成音频及视频文件的播放，其实现步骤如下。

（1）新建一个“标准 EXE”工程。

（2）添加 WindowsMediaPlayer 控件。选择“工程”→“部件”命令，打开“部件”对话框，在“控件”选项卡中选择“Windows Media Player”选项，将该控件加载到工具箱中，如图 12-6 所示。加载后在控件工具栏上的位置如图 12-7 所示。

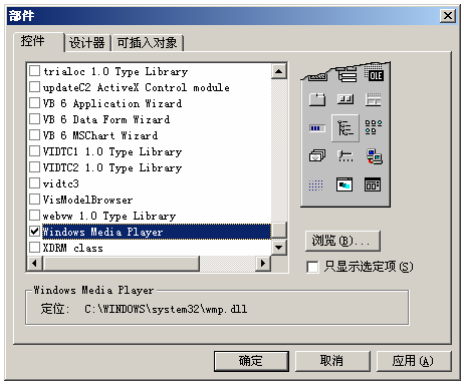

图 12-6　添加到工具栏

图 12-7　Windows Media Player 控件

（3）添加控件并设置属性。向窗体添加一个 WindowsMediaPlay 控件、一个标签控件、两个按钮控件及一个 Common Dialog 控件，窗体和控件具体属性设置如表 12-6 所示。

表 12-6　窗体及控件的属性设置

控件	属性名	设置状态
Form	Caption	WindowsMediaPlayer 控件
Command1	Caption	打开
Command2	Caption	关闭

上述属性设置完成后，该多媒体播放器的窗体设计如图 12-8 所示。

（4）编写程序代码。该实例实现当用户打开一个音频或视频文件时自动播放该文件，因此在“打开”按钮中输入如下事件代码。

```
Private Sub Command1_Click()
CommonDialog1.ShowOpen                                    '打开“打开”对话框
WindowsMediaPlayer1.URL = CommonDialog1.FileName          '获取打开文件并开始播放
Label1.Caption = WindowsMediaPlayer1.currentMedia.sourceURL '获取当前播放的文件路径并显示
End Sub
```

此外，在“关闭”按钮中输入 End 代码，退出程序即可。

（5）运行程序。运行该实例后，单击“打开”按钮后选择打开本地计算机中的一个多媒体文件后，播放器开始播放该文件，如图 12-9 所示。

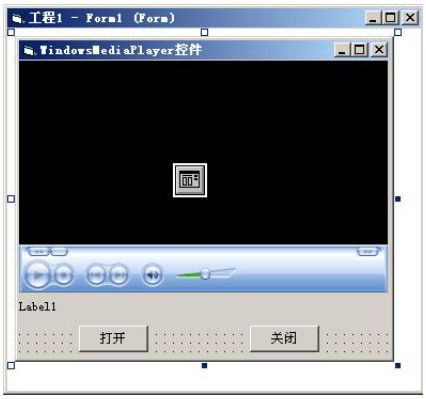

图 12-8　窗体设计

图 12-9　运行多媒体播放器

至此，使用 Visual Basic 6.0 提供的多媒体控件实现多媒体编程就完成了，下节将对 Visual Basic 的另外一种多媒体编程技术——API 实现多媒体编程进行简要介绍。

12.2　API 概述

Visual Basic 6.0 在多媒体编程方面功能强大，而 Visual Basic 6.0 对多媒体的操作都可以通过 Windows API 函数来实现。因此，本节将重点介绍 Windows API 函数。

12.2.1　Windows API

Windows API（Application Programming Interface, 应用程序编程接口）是 Microsoft 系统预先定义的一套 Windows 函数，用于控制系统中各个部件的外观和行为。用户的每个动作都会引发一个或几个 API 函数的运行，用于告诉 Windows 系统需要发生的事件。

可以看出，Windows API 也就是一系列的底层函数，是系统提供给用户进入操作系统核心，进行高级编程的途径。通过在 Visual Basic 应用程序中声明外部过程就能够访问 Windows API（以及其他的外部 DLL）。在声明了过程之后，调用它的方法与调用 Visual Basic 自己的过程相同。

一般的程序设计语言只能提供一种自动的而且更容易访问 API 的方法。Windows API 函数由许多动态链接库（DLL）组成。在 32 位 Windows 系统中，核心的 API DLL 如下。

- gdi32.dll：图形显示界面的 API。
- kernel32.dll：处理低级任务（如内存和任务管理）的 API。
- user32.d11：处理窗口和消息（能把其中一些当作事件访问）的 API。
- Winmm.dll：处理多媒体任务（如波形音频、MIDI 音乐和数字影像等）的 API。多

媒体编程中主要使用到的 API 函数就在这个链接库中。

使用 Visual Basic 语言编写的每行代码都会被转换为 API 函数，然后传递给 Windows。例如，语句 Form1.Print Visual Basic 将会以一定的参数调用 TextOut 这个 API 函数来实现在窗体上输出 Visual Basic 字符串。同样，当单击窗体上的一个按钮时，Windows 系统会发送一个消息给窗体（这对用户来说是透明的），Visual Basic 获取这个调用并经过分析后生成一个特定事件，如按钮单击事件 Button_Click。

至于 Microsoft 究竟提供了哪些 API 函数，以及其对应功能，这里不再赘述，感兴趣的读者可参阅 API 函数的相关资料。

12.2.2 查看 API

如前所述，Windows API 函数的数量众多，读者不可能详细了解每个 API 函数的具体功能。在 Visual Basic 6.0 中，提供了一个查看 API 的工具——API Viewer，用于查看 API 函数和常量。使用 API Viewer 查看 API 函数的方法如下。

（1）启动 API Viewer。选择“开始”菜单中 Visual Basic 程序组中的“API 文本浏览器”命令来启动 API Viewer，如图 12-10 所示。API Viewer 的主界面如图 12-11 所示。

图 12-10　启动 API Viewer

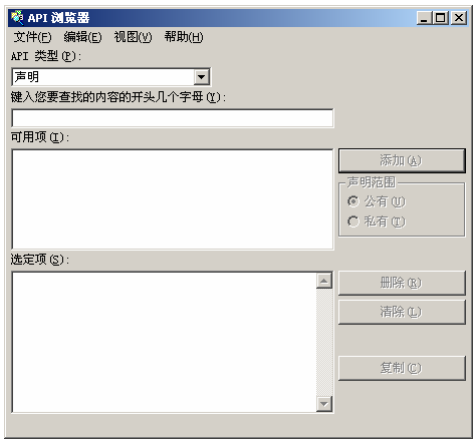

图 12-11　API Viewer 主界面

（2）打开 API 文件。启动完成后，选择“文件”→“加载文本文件”命令，在“选择一个文本 API 文件”对话框中选择一个要打开的 API 文件，如打开 WIN32API 文件，如图 12-12 所示。

（3）查找 API 函数。打开存储 API 的文件后，就可以在文本框中输入要搜索的 API 函数、常量或类型的开头几个字母，将会在“可用项”列表框中显示查找到的相应内容。例如，在文本框中输入 Open 关键字后，“可用项”列表框中将会列出以 Open 开头的所有函数名，如图 12-13 所示。

（4）复制 API 函数声明。找到所需的 API 函数后，在列表框中选中函数后单击“添加”按钮，在“选定项”文本框中就显示出该函数的 Visual Basic 声明，单击“复制”按钮即可将其复制到剪贴板中，随后可粘贴到用户工程代码中，如图 12-14 所示。

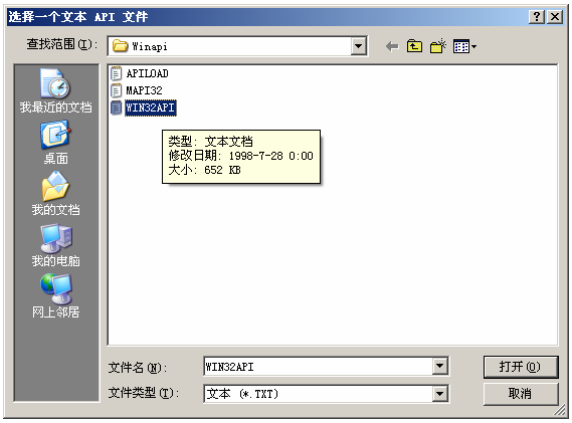

图 12-12　加载文本 API 文件

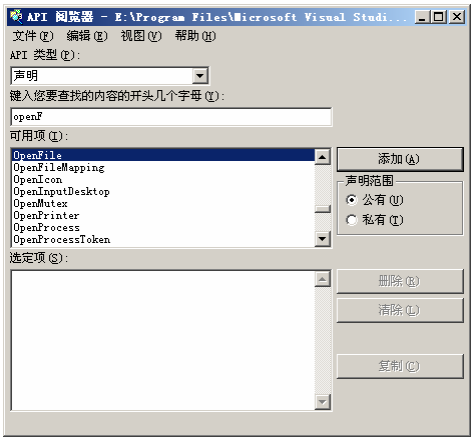

图 12-13　查找 API 函数

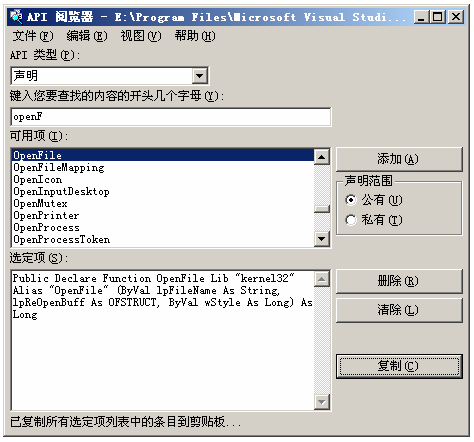

图 12-14　复制 API 函数声明

通过 API 函数 OpenFile 的声明，可以看出，API 函数的声明是通过关键字 Declared 定义的一个公有函数。例如，为 OpenFile 函数的声明如下。

```
Public Declare Function OpenFile Lib "kernel32" Alias "OpenFile" (ByVal lpFileName As String, lpReOpenBuff As OFSTRUCT, ByVal wStyle As Long) As Long
```

其中的参数说明如下。

- Declare：关键字，表明要声明一个外部过程。
- Function：表明声明的是一个外部函数。如果为 Sub，则为外部过程。
- OpenFile：要声明的外部过程的函数名。
- Lib：关键字，表明函数位于下面的 DLL 库中。
- "kernel32"：DLL 的库名，必须用引号括起来。
- Alias：关键字，表明可以用紧跟其后的别名“OpenFile”来调用此函数。
- OFSTRUCT：为参数 lpReOpenBuff 的数据类型。

12.2.3 使用 API

API 函数在使用前，必须先复制该函数的声明到工程中，可以将该声明复制到剪贴板中，粘贴到工程代码中声明 API 函数，也可以直接键入 API 函数声明。

下面通过一个具体实例来介绍在 Visual Basic 6.0 中如何使用 API。例如，利用 API 制作一个总在最前的窗口，其实现步骤如下。

（1）新建“标准 EXE”工程，该工程包含 3 个窗体和一个标准模块，窗体的名称分别为 Form1、Form2 和 Form3，标准模块的名称为 Module1。

说明：标准模块在工程中的主要功能是包含一些全局级变量和函数的声明以及定义，以供在工程的其他窗体模块中使用。在 Visual Basic 6.0 的工程中添加标准模块的方法为：选择“工程”→“添加模块”命令，即可打开“添加模块”对话框，如图 12-15 所示。

在图 12-5 中单击“打开”按钮即可将一个标准模块添加到工程中了。此时再查看 Visual Basic 6.0 的工程资源管理器，就可以看到新增加的标准模块 Module1，如图 12-16 所示。

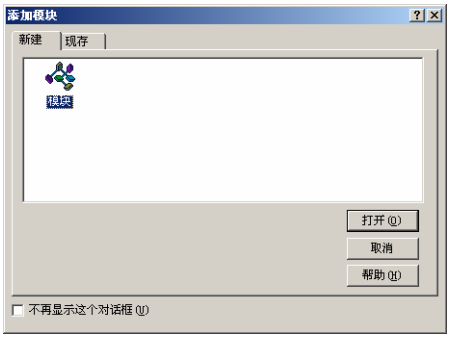

图 12-15 “添加模块”对话框

图 12-16 添加了标准模块的工程资源管理器

（2）在 API Viewer 中查找 API 函数的声明。在 API 阅览器中查找 SetWindowPos 函数，单击“添加”按钮，在“选定项”文本框中就会显示出该函数的声明文本，然后单击“复制”按钮，如图 12-17 所示。

（3）复制 API 函数到标准模块中。打开 Module1.bas 并将上述函数声明文本粘贴到其中，如图 12-18 所示。

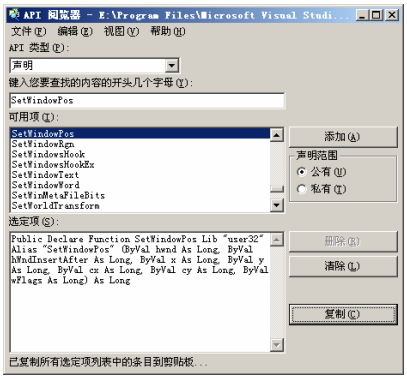

图 12-17 查找 API 函数并复制

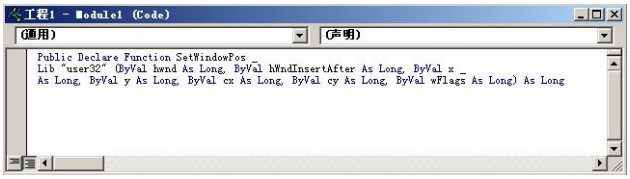

图 12-18 复制 API 函数到标准模块中

（4）使用 API 函数。打开窗体 Form3 的代码编辑框，在其中输入使用该 API 函数的事件代码。需要注意的是，由于该实例要求窗体 Form3 启动时显示在最前，因此驱动事件应为启动 Form3 窗体时的 Load 装载事件，事件代码如下。

```
Const SWP_NOMOVE = 2                                    '不更新窗口位置
Const SWP_NOSIZE = 1                                    '不更新窗口大小
Const HWND_TOPMOST = -1                                 '窗口放在所有窗口顶部
Const HWND_NOTOPMOST = -2                               '窗口不能放在所有窗口顶部
Private Sub Form_Load()                                 '装载事件
Form1.Show                                              '依次显示 3 个窗体
Form2.Show
Form3.Show
SetWindowPos Form3.hwnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE_ + SWP_NOSIZE
                                                        '设置第 3 个窗体显示在最前
End Sub
```

可以看出，上述代码将 API 函数 SetWindowsPos 的 3 个参数都以常量的方式进行了定义，在 Form3 的 Load 事件中则依次显示 3 个窗体。

（5）运行上述实例，执行结果如图 12-19 所示。

可以看到，代码中设定的是 Form1、Form2 和 Form3 这 3 个窗体依次显示，由于使用了 SetWindowsPos 函数设定 Form3 始终显示在最前，即使运行后焦点不在 Form3 上，其依然不会被其他窗体覆盖而显示在最前端。

图 12-19　API 应用

12.2.4　注意事项

由上述实例可知，使用 Visual Basic 调用 API 可实现许多独特的功能。但是，作为一种高效编程环境，Visual Basic 封装了部分 Windows API 函数，牺牲了一些 API 的功能。调用 API 时稍有不慎就可能导致 API 编程错误，出现难以捕获或间歇性的错误，甚至导致程序崩溃。要避免 API 编程错误，提高 Visual Basic 调用 API 的安全性，应重点注意如下事项。

- 指定“Option Explicit”：编程前最好将 Visual Basic 编程环境中的“Require Variable Declaration（要求变量申明）项选中。如果该项未被指定，任何简单的录入错误都可能会产生一个 Variant 变量。
- 注意 Visual Basic 整数和 Win32 整数的区别：在 Visual Basic 环境下，涉及到的所有 Integer（整型数）都是 16 位，而一旦涉及 C/C++的 Win32 文档时，则是 32 位。
- 注意检查参数类型：API 错误中，除了因遗漏 ByVal 关键字导致的错误外，大约有一半是因为声明中有不正确的参数类型。在 Win32 环境下，无论是 8 位、16 位，还是 32 位数值变量都是以 32 位传递，如果同时使用，则很难发现其中错误。
- 切勿忽视 ByVal，确保函数声明的完整性：ByVal 用于“按值”调用。参数传递时，不是将指向 DLL 的指针传递给参数变量本身，而是将传递参数值的一份拷贝传递给 DLL。
- 重新检查函数名：在 Win32 环境下，API 函数的名称要求区分大小写。在一个 DLL

函数里找不到声明的函数时，有必要检查一下函数名。

- 预先初始化字符串，以免造成冲突：如果 API 函数需要一个指向缓冲区的指针，以便从中载入数据，而此时传递的是字符串变量，应该先初始化字符串长度。
- 跟踪检查参数、返回类型和返回值：利用立即模式和单步调试功能确保函数声明的类型明确，通过跟踪和检查参数的来源及类型，可以排除参数的错误传递。

12.3 API 多媒体编程

上节介绍了 Windows 提供的 API 函数的基础知识，本节将介绍使用其提供的多媒体函数来实现多媒体编程。

12.3.1 常用 API 多媒体函数

在开发复杂的多媒体应用程序时，需要使用到高级的 MCI 命令，如用 sysinfo 命令得到设备的安装名称。这时，使用 Windows 系统中 Winmm.dll 动态链接库提供的 100 多个处理多媒体的 API 函数来查询和控制 MCI 设备就会非常方便。

通常，适合 Visual Basic 使用的 API 函数主要有 mciExecute()、mciSendCommand()、mciSendString()和 mciGetErrorString()等，下面将简单介绍这几个函数。

（1）mciSendString 函数：其功能是传送指令字符串给 MCI。可以通过 Visual Basic 6.0 提供的 API Viewer 查看到该函数的声明格式如下。

```
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
```

（2）mciExecute 函数：其功能和 mciSendString 一样，不同的是当发生错误时，mciExecute 会弹出对话框显示错误信息。该函数的声明格式如下。

```
Declare Function mciExecute Lib "winmm.dll" Alias "mciExecute" (ByVal lpstrCommand As String) As Long
```

（3）mciGetErrorString 函数：其功能是将 MCI 错误代码转换为字符串。该函数的声明格式如下。

```
Declare Function mciGetErrorString Lib "winmm.dll" Alias "mciGetErrorStringA" (ByVal dwError As Long, ByVal lpstrBuffer As String, ByVal uLength As Long) As Long
```

这几个函数在具体的多媒体应用程序设计中有其各自的用处，也是 Visual Basic 6.0 中常用的支持多媒体编程的 API 函数。

12.3.2 使用 API 函数制作播放器

下面通过一个制作播放器的实例来介绍在 Visual Basic 6.0 中如何使用 API 函数来进行多

媒体编程。与通过 MCI 控件制作音频播放器类似，该播放器同样也支持各种音频文件的播放，也需要包含控制播放的功能，其实现步骤如下。

（1）设计窗体。该窗体包含 6 个按钮，为简便起见，此处以工具栏 ToolBar 控件来设计这几个按钮，如图 12-20 所示。

（2）添加 API 函数声明。此处用到的是 mciExecute()函数，因此需添加该函数的声明到 Module 标准模块中。在 API Viewer 中找到该函数，如图 12-21 所示。

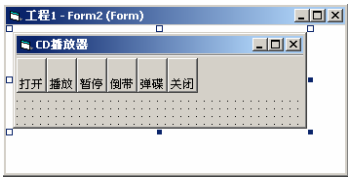

图 12-20　CD 播放器窗体设计

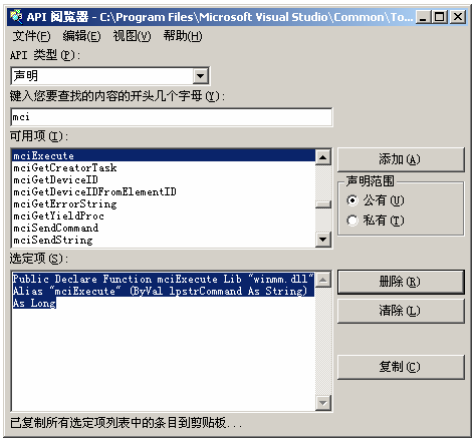

图 12-21　查找 API 函数

复制该函数的声明文本后，打开该实例工程的 APIModule1.bas 标准模块，选择“编辑”→“粘贴”命令后，该函数声明就添加到了模块中，如图 12-22 所示。

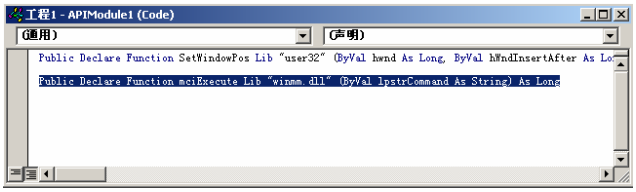

图 12-22　添加函数声明

（3）编写事件代码。该实例中，定义了一个通用函数 PlayCD，并在窗体 Form1 的 Load 事件中，将几个按钮进行了初始化，但主要代码集中在工具栏 ToolBar 上，代码如下。

```
Private Sub PlayCD()
    mciExecute "Play cd"                                                    '开始播放
    Toolbar1.Buttons(2).Enabled = False: Toolbar1.Buttons(3).Enabled = True'设定按钮是否可用
    Toolbar1.Buttons(4).Enabled = False: Toolbar1.Buttons(5).Enabled = False
End Sub
Private Sub Form_Load()                                                     '初始化各按钮状态
    Toolbar1.Buttons(1).Enabled = True: Toolbar1.Buttons(2).Enabled = False
    Toolbar1.Buttons(3).Enabled = False: Toolbar1.Buttons(4).Enabled = False
    Toolbar1.Buttons(5).Enabled = False: Toolbar1.Buttons(6).Enabled = True
End Sub
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Index
    Case 1                                                                  '单击“打开”按钮
        mciExecute "Open Cdaudio Alias cd"                                  '打开媒体设备
```

```
        Toolbar1.Buttons(1).Enabled = False: Toolbar1.Buttons(2).Enabled = True
        Toolbar1.Buttons(3).Enabled = False: Toolbar1.Buttons(4).Enabled = False
        Toolbar1.Buttons(5).Enabled = False
        PlayCD                                                    '调用 PlayCD 过程开始播放
    Case 2                                                        '单击“播放”按钮
        PlayCD                                                    '调用 PlayCD 过程开始播放
    Case "3                                                       '单击“暂停”按钮"
        mciExecute "Stop cd"                                      '暂停播放
        Toolbar1.Buttons(2).Enabled = True: Toolbar1.Buttons(3).Enabled = False
        Toolbar1.Buttons(4).Enabled = True: Toolbar1.Buttons(5).Enabled = True
    Case 4                                                        '单击“倒带”按钮
        mciExecute "Seek cd To Start"                             '移动到起始位置
        Toolbar1.Buttons(1).Enabled = False: Toolbar1.Buttons(2).Enabled = True
        Toolbar1.Buttons(3).Enabled = False: Toolbar1.Buttons(4).Enabled = False
        Toolbar1.Buttons(5).Enabled = True:
    Case 5                          '单击“弹碟”按钮，该按钮的标题在“弹碟”和“装碟”之间转化
        If Toolbar1.Buttons(5).Caption = "弹碟" Then
        mciExecute "Set cd Door open"
        Toolbar1.Buttons(5).Caption = "装碟"
    Else
        mciExecute "Set cd Door closed"
        Toolbar1.Buttons(5).Caption = "弹碟"
    End If
    Toolbar1.Buttons(1).Enabled = False: Toolbar1.Buttons(2).Enabled = True
    Toolbar1.Buttons(3).Enabled = False: Toolbar1.Buttons(4).Enabled = False
    Case 6                                                        '单击“关闭”按钮
      mciExecute "Close cdp"                                      '关闭媒体设备
    End Select
End Sub
```

（4）运行程序。完成编码后，运行上述实例，结果如图 12-23 和图 12-24 所示。

图 12-23　播放前

图 12-24　正在进行播放

此外，使用 sndPlayComand 函数可以直接播放音频文件或系统声音。其包含两个参数，参数 lpszSoundName 用于指定播放的音频文件或系统声音，参数 uFlags 用于设定播放状态。参数 uFlags 的设置值如表 12-7 所示。

表 12-7　sndPlayComand 函数参数 uFlags 的设置值

设置值	说明
&H00	同步播放
&H01	非同步播放
&H02	即使没有指定文件也不播放预设声音
&H8	重复播放
&H16	不停止其他正在播放的声音

12.4　综合实例

电影（包括声音和图像）是应用最为广泛的媒体信息之一，也是多媒体关键技术之一。在多媒体系统中，AVI（Audio Video Interface，音频视频接口）文件是存储电影（包括声音和图像）的标准格式。AVI 文件一般是通过捕获实时视频信号得来的，也可以通过扫描仪获取图像或者使用动画制作软件得到。于是，屏幕窗口上的音频视频操作，就变成了对 AVI 文件的处理。

播放 AVI 文件（即影片）的方法有许多种，而其中以利用 Visual Basic 6.0 所提供的 MCI 控件的实现方法最为简单方便。下面以一个具体实例来说明如何制作播放 AVI 文件，其实现步骤如下。

（1)新建“标准 EXE”工程，在新建窗体上添加 1 个 TextBox 文本框控件、3 个 CommandButton 按钮控件和 1 个 MMControl 控件。

（2）设置窗体和控件属性。通过窗体的属性设置和控件的属性设置设计窗体界面，如图 12-25 所示。

图 12-25　窗体设计

可以看出，此处为该界面设置了窗体和控件的属性。其窗体和控件的属性设置如表 12-8 所示。

表 12-8　控件属性设置

控件	属性	设置值
Form1	Caption	Form1.Caption="AVI 播放器"
Command1	Caption	Command1.Caption="播放"
Command2	Caption	Command2.Caption="暂停"
Command3	Caption	Command3.Caption="退出"
Text1	MultiLine	True
	Enable	False
MMControl	Visible	False

可以看出，上述窗体设计中，由于需要将 TextBox 作为播放窗口，因此设置其 MultiLine 属性为 True，Enable 属性为 False，这样它就不能接收用户输入。此外，将 MMControl 控件的 Visible 属性设为 False，使它运行时不可见，使运行界面变得美观。

（3）编写事件代码。该实例要求播放 AVI 文件，因此同样需要通过 MMControl 控件进行多媒体编程的一般步骤，同时还需设置其 DeviceType 值为 AVIVedio，其事件代码如下。

```
Option Explicit
Dim PauseTimes As Integer                          '记录单击“暂停”按钮的次数
Private Sub Form_Load()                            '初始化
PauseTimes = 0
MMControl1.DeviceType = "AVIVideo"                 '指定 Mci 设备类型
MMControl1.FileName = App.Path & "\1.avi"          '设定播放的文件，你可以自行设定
```

```
MMControl1.Command = "Open"                          '执行打开命令
MMControl1.hWndDisplay = Text1.hWnd                  '在文本框上播放
CmdPause.Enabled = False
End Sub
Private Sub Command1_Click()                         '播放按钮
Command2.Caption = "暂  停(&U)"
PauseTimes = 0                                       '变量重新初始化
Command2.Enabled = True                              '暂停按钮可以使用
MMControl1.Notify = True
MMControl1.Command = "Play"                          '播放
End Sub
Private Sub Command2_Click()                         '暂停按钮
PauseTimes = PauseTimes + 1
If PauseTimes Mod 2 = 1 Then                         '奇数，改变按钮的 Caption 属性
    Command2.Caption = "继  续(&C)"
Else
    Command2.Caption = "暂  停(&U)"
End If
MMControl1.Command = "Pause"                         '执行 Pause 命令
End Sub
Private Sub MMControl1_Done(NotifyCode As Integer)
If NotifyCode = 1 Then                               '播放正常完毕
    MMControl1.To = 0
    MMControl1.Command = "Seek"                      '移至开头
    Command2.Enabled = False
End If
End Sub
Private Sub Command3_Click()                         '退出按钮
MMControl1.Command = "Close"                         '先发出关闭命令，再关闭程序
Unload Me
End Sub
```

（4）运行程序。完成上述编码后，运行该程序。需要注意的是，该程序的运行需要在当前路径下存在 1.avi 文件，否则将无法播放，该文件在光盘源码的当前文件夹中。运行该程序后，单击“播放”按钮后，其结果如图 12-26 所示。单击“暂停”按钮后则暂停播放，单击“退出”按钮后则程序结束，这就实现了 AVI 播放器的功能。

图 12-26　运行 AVI 播放器

12.5　小结

Visual Basic 6.0 对多媒体编程的支持主要体现在对音频视频的处理上，本章主要介绍了两种多媒体编程的方法。Visual Basic 6.0 提供了一个 ActiveX 控件用于支持多媒体编程，这就是 MCI 控件。本章详细介绍了 MCI 控件的属性和命令，并对如何使用该控件进行多媒体编程进行了详细讲解。此外，使用 Windows API 函数处理多媒体也是编程的常用手段，本章同样详细介绍了几个常用的 API 函数及其使用方法。最后，通过具体的实例，依次介绍了音频播放器和 AVI 播放器的实现过程。本章涉及到了许多 API 函数，如需详细了解，可参阅 API 函数的相关资料。